

## **Oracle Fusion Middleware**

Developer's Guide for Oracle TopLink

11g Release 1 (11.1.1)

**B32441-03**

May 2009

Oracle Fusion Middleware Developer's Guide for Oracle TopLink, 11g Release 1 (11.1.1)

B32441-03

Copyright © 1997, 2009, Oracle and/or its affiliates. All rights reserved.

Primary Author: Liza Rekadze

Contributing Author: Rick Sapir

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.





---

---

# Contents

<b>Preface</b> .....	lix
Audience .....	lix
Documentation Accessibility .....	lix
Related Documentation .....	lx
Conventions .....	lx
<b>Part I TopLink Application Development Overview</b>	
<b>1 Introduction to TopLink</b>	
1.1 What Is TopLink? .....	1-1
1.2 What Is the Object-Persistence Impedance Mismatch.....	1-3
1.3 TopLink Key Features .....	1-4
1.4 TopLink Application Architectures .....	1-5
<b>2 Introduction to TopLink Application Development</b>	
2.1 Introduction to TopLink Application Development .....	2-1
2.1.1 Typical Development Stages .....	2-2
2.1.2 Oracle Development Support .....	2-3
2.2 Designing Your Application with TopLink .....	2-3
2.2.1 How to Use TopLink in Your Application Design .....	2-4
2.2.1.1 Relational Database Usage .....	2-4
2.2.1.2 Object-Relational Data Type Database Usage .....	2-4
2.2.1.3 Oracle XML Database (XDB) Usage.....	2-4
2.2.1.4 Enterprise Information System (EIS) Usage .....	2-4
2.2.1.5 XML Usage .....	2-4
2.2.2 Target Platforms.....	2-5
2.3 Selecting an Architecture with TopLink.....	2-5
2.3.1 Tiers .....	2-5
2.3.1.1 Three Tier.....	2-6
2.3.1.1.1 Java EE or Non-Java EE .....	2-6
2.3.1.1.2 Client .....	2-6
2.3.1.2 Two Tier .....	2-6
2.3.2 Service Layer .....	2-7
2.3.2.1 EJB Session Beans .....	2-7
2.3.2.1.1 Stateful .....	2-7

2.3.2.1.2	Stateless.....	2-7
2.3.2.2	EJB Entity Beans.....	2-7
2.3.2.2.1	Container-Managed Persistence (CMP).....	2-8
2.3.2.2.2	Bean-Managed Persistence (BMP).....	2-8
2.3.2.3	JPA Entities.....	2-8
2.3.2.4	Plain Old Java Objects (POJO).....	2-8
2.3.3	Data Access.....	2-8
2.3.3.1	Data Type.....	2-9
2.3.3.2	Multiple Data Sources.....	2-9
2.3.3.3	Isolating Data Access.....	2-9
2.3.3.4	Historical Data Access.....	2-9
2.3.4	Caching.....	2-9
2.3.4.1	Cache Type.....	2-10
2.3.4.2	Refreshing.....	2-10
2.3.4.3	Cache Coordination.....	2-10
2.3.4.3.1	Protocol.....	2-10
2.3.4.3.2	Synchronization.....	2-10
2.3.5	Locking.....	2-11
2.3.5.1	Optimistic Locking.....	2-11
2.3.5.2	Pessimistic Locking.....	2-11
2.4	Building and Using the Persistence Layer.....	2-11
2.4.1	Implementation Options.....	2-11
2.4.1.1	Using EclipseLink JPA Metatdata, Annotations, and XML.....	2-12
2.4.1.2	Using TopLink Metatdata XML.....	2-12
2.4.1.3	Using TopLink Metadata Java API.....	2-12
2.4.1.4	Using Method and Direct Field Access.....	2-12
2.4.1.5	Using Weaving.....	2-13
2.4.2	Persistent Class Requirements.....	2-13
2.4.3	Persistence Layer Components.....	2-14
2.4.3.1	Mapping Metadata.....	2-14
2.4.3.2	Session.....	2-14
2.4.3.3	Cache.....	2-15
2.4.3.4	Queries and Expressions.....	2-15
2.4.3.5	Transactions.....	2-15
2.4.4	How to Use the Persistence Layer.....	2-16
2.5	Deploying the Application.....	2-16
2.5.1	About Deployments.....	2-17
2.5.2	How to Use TopLink in a Java EE Application.....	2-17
2.6	Optimizing and Customizing the Application.....	2-17
2.7	Troubleshooting the Application.....	2-17
2.8	Persisting Objects.....	2-18
2.8.1	Application Object Model.....	2-18
2.8.2	Data Storage Schema.....	2-18
2.8.3	Primary Keys and Object Identity.....	2-19
2.8.4	Mappings.....	2-19
2.8.5	Foreign Keys and Object Relationships.....	2-19
2.8.6	Inheritance.....	2-19

2.8.7	Concurrency .....	2-20
2.8.8	Caching.....	2-20
2.8.9	Nonintrusive Persistence.....	2-20
2.8.10	Indirection.....	2-21
2.8.11	Mutability.....	2-21
2.9	Working with TopLink Metadata.....	2-22
2.9.1	Advantages of the TopLink Metadata Architecture.....	2-23
2.9.2	Creating Project Metadata .....	2-24
2.9.2.1	Descriptors and Mappings.....	2-24
2.9.2.1.1	Amending Descriptors .....	2-24
2.9.2.2	Data Source Login Information.....	2-25
2.9.3	Creating Session Metadata .....	2-25
2.9.4	Deploying Metadata.....	2-25
2.10	Using Weaving.....	2-25
2.10.1	Configuring Dynamic Weaving Using EclipseLink Agent .....	2-26
2.10.1.1	To Configure Dynamic Weaving Using EclipseLink Agent .....	2-26
2.10.2	Configuring Static Weaving.....	2-26
2.10.3	Disabling Weaving Using TopLink Persistence Unit Properties.....	2-26
2.10.4	Packaging a POJO Application for Weaving.....	2-27
2.10.4.1	To Package a POJO Application for Weaving.....	2-27
2.10.5	What You May Need to Know About Weaving and POJO Classes.....	2-27
2.10.6	What You May Need to Know About Weaving and Java EE Application Servers .....	2-28
2.11	Considering Three-Tier Architecture.....	2-28
2.11.1	Example Implementations.....	2-29
2.11.2	Advantages and Disadvantages .....	2-29
2.11.3	Variation Using Remote Sessions.....	2-29
2.11.4	Technical Challenges.....	2-29
2.12	Considering Two-Tier Architecture .....	2-30
2.12.1	Example Implementations.....	2-31
2.12.2	Advantages and Disadvantages .....	2-31
2.12.3	Technical Challenges.....	2-31
2.13	Considering EJB Session Bean Facade Architecture .....	2-31
2.13.1	Example Implementation .....	2-32
2.13.2	Advantages and Disadvantages .....	2-32
2.13.3	What Are Session Beans.....	2-33
2.13.4	Technical Challenges.....	2-33
2.13.5	What Is a Unit of Work Merge.....	2-34
2.14	Considering EJB Entity Beans with CMP Architecture.....	2-34
2.14.1	Example Implementation .....	2-35
2.14.2	Advantages and Disadvantages .....	2-35
2.14.3	Technical Challenges.....	2-36
2.14.3.1	External JDBC Pools.....	2-36
2.14.3.2	JTA/JTS Integration .....	2-36
2.14.3.3	Cache Coordination.....	2-36
2.14.3.4	Maintaining Bidirectional Relationships.....	2-36
2.14.3.5	Managing Dependent Objects .....	2-37
2.14.3.6	Managing Collections of EJBObject Objects .....	2-37

2.15	Considering EJB Entity Beans with BMP Architecture .....	2-38
2.15.1	Example Implementations.....	2-39
2.15.2	Advantages and Disadvantages .....	2-39
2.15.3	Technical Challenges.....	2-40
2.15.3.1	External JDBC Pools .....	2-40
2.15.3.2	JTA/JTS Integration .....	2-40
2.15.3.3	Cache Coordination.....	2-40
2.16	Considering JPA Entity Architecture.....	2-40
2.16.1	Example Implementations.....	2-42
2.16.2	Advantages and Disadvantages .....	2-42
2.17	Considering Web Services Architecture.....	2-42
2.17.1	Example Implementations.....	2-43
2.17.2	Advantages and Disadvantages .....	2-43
2.17.3	Technical Challenges.....	2-43
2.18	Considering EclipseLink Service Data Objects (SDO) Architecture.....	2-43

## **Part II TopLink Development Tools Overview**

### **3 Introduction to TopLink Development Tools**

3.1	Development Environment .....	3-2
3.2	TopLink Run-Time Environment .....	3-3

### **4 Using Oracle JDeveloper TopLink Editor**

4.1	Introduction to Oracle JDeveloper TopLink Editor .....	4-1
4.2	Configuring the Oracle JDeveloper TopLink Editor .....	4-1
4.3	Using the Oracle JDeveloper TopLink Editor.....	4-1
4.3.1	TopLink Project Elements in the Application Navigator.....	4-2
4.3.2	TopLink Editor Tabs in the Editor Window .....	4-2
4.3.3	TopLink Project Elements in the Structure Window .....	4-3

### **5 Using TopLink Workbench**

5.1	Introduction to TopLink Workbench.....	5-1
5.2	Configuring the TopLink Workbench Environment.....	5-2
5.2.1	How to Configure the Language Preference .....	5-3
5.3	Using TopLink Workbench .....	5-3
5.3.1	How to Use Menus .....	5-5
5.3.1.1	Using Menu Bar Menus.....	5-5
5.3.1.2	Using Context Menus .....	5-6
5.3.2	How to Use Toolbars.....	5-6
5.3.2.1	Using Standard Toolbar.....	5-6
5.3.2.2	Using Context Toolbar .....	5-7
5.3.3	How to Use the Navigator.....	5-9
5.3.4	How to Use the Editor.....	5-11
5.3.5	How to Use the Problems Window.....	5-11
5.3.6	How to Use the Online Help.....	5-12
5.4	Using TopLink Workbench Preferences.....	5-12



5.4.1	How to Use General Preferences .....	5-13
5.4.2	How to Use Help Preferences .....	5-14
5.4.3	How to Use Mappings Preferences.....	5-15
5.4.4	How to Use Class Preferences .....	5-16
5.4.5	How to Use EJB Preferences.....	5-17
5.4.6	How to Use Database Preferences.....	5-18
5.4.7	How to Use Sessions Configuration Preferences .....	5-19
5.4.8	How to Use New Names Preferences.....	5-19
5.4.9	How to Use Session Platform Preferences .....	5-20
5.4.10	How to Use Platforms Preferences.....	5-21
5.5	Using Databases .....	5-22
5.5.1	How to Use Database Tables in the Navigator Window .....	5-23
5.5.1.1	Logging In and Out of a Database .....	5-23
5.5.1.2	Creating New Tables.....	5-23
5.5.1.3	Importing Tables from a Database.....	5-24
5.5.1.4	Removing Tables .....	5-26
5.5.1.5	Renaming Tables .....	5-26
5.5.1.6	Refreshing Tables from the Database .....	5-26
5.5.2	How to Use Database Tables in the Editor Window .....	5-27
5.5.2.1	Working with Column Properties.....	5-27
5.5.2.2	Setting a Primary Key for Database Tables .....	5-28
5.5.2.3	Creating Table References.....	5-29
5.5.2.4	Creating Field Associations .....	5-30
5.5.3	How to Generate Data from Database Tables .....	5-31
5.5.3.1	Generating SQL Creation Scripts .....	5-32
5.5.3.2	Generating Classes and Descriptors from Database Tables.....	5-32
5.5.3.3	Generating EJB Entity Beans and Descriptors from Database Tables .....	5-34
5.5.3.4	Generating Tables on the Database.....	5-35
5.6	Using XML Schemas.....	5-36
5.6.1	How to Use XML Schemas in the Navigator .....	5-36
5.6.2	How to Use an XML Schema Structure.....	5-37
5.6.3	How to Import an XML Schema.....	5-38
5.6.4	How to Configure an XML Schema Reference.....	5-40
5.6.4.1	How to Configure an XML Schema Reference Using TopLink Workbench ....	5-40
5.6.4.2	How to Configure an XML Schema Reference Using Java .....	5-41
5.6.5	How to Configure XML Schema Namespace.....	5-42
5.6.5.1	How to Configure XML Schema Namespace Using TopLink Workbench .....	5-43
5.6.5.2	How to Configure XML Schema Namespace Using Java.....	5-44
5.7	Using Classes .....	5-44
5.7.1	How to Create Classes .....	5-44
5.7.1.1	How to Create Classes Using TopLink Workbench.....	5-44
5.7.2	How to Configure Classes.....	5-45
5.7.2.1	Configuring Class Information.....	5-46
5.7.2.1.1	Using TopLink Workbench.....	5-46
5.7.2.2	Configuring Class Modifiers.....	5-46
5.7.2.2.1	Using TopLink Workbench.....	5-46
5.7.2.3	Configuring Class Interfaces.....	5-47

5.7.2.3.1	Using TopLink Workbench.....	5-47
5.7.2.4	Adding Attributes .....	5-48
5.7.2.4.1	Using TopLink Workbench.....	5-48
5.7.2.5	Configuring Attribute Modifiers.....	5-48
5.7.2.5.1	Using TopLink Workbench.....	5-48
5.7.2.6	Configuring Attribute Type Declaration .....	5-49
5.7.2.6.1	Using TopLink Workbench.....	5-49
5.7.2.7	Configuring Attribute Accessing Methods.....	5-51
5.7.2.7.1	Using TopLink Workbench.....	5-51
5.7.2.8	Adding Methods.....	5-52
5.7.2.8.1	Using TopLink Workbench.....	5-52
5.7.2.9	Configuring Method Modifiers .....	5-53
5.7.2.9.1	Using TopLink Workbench.....	5-53
5.7.2.10	Configuring Method Return Type .....	5-54
5.7.2.10.1	Using TopLink Workbench.....	5-54
5.7.2.11	Configuring Method Parameters .....	5-54
5.7.2.11.1	Using TopLink Workbench.....	5-54
5.7.3	How to Import and Update Classes.....	5-55
5.7.3.1	Importing and Updating Classes Using TopLink Workbench.....	5-55
5.7.4	How to Manage Nondesoriptor Classes.....	5-57
5.7.5	How to Rename Packages .....	5-57
5.7.5.1	Renaming Packages Using TopLink Workbench .....	5-58
5.8	Integrating TopLink Workbench with Apache Ant.....	5-58
5.8.1	How to Configure Ant to Use TopLink Workbench Tasks .....	5-59
5.8.1.1	Creating Library Dependencies.....	5-59
5.8.1.2	Declaring TopLink Workbench Tasks .....	5-59
5.8.2	What You May Need to Know About TopLink Workbench Ant Task API.....	5-60
5.8.3	How to Create TopLink Workbench Ant Tasks.....	5-60
5.8.4	How to Create the mappings.validate Task .....	5-62
5.8.4.1	Using Parameters.....	5-62
5.8.4.2	Specifying Parameters Specified as Nested Elements.....	5-62
5.8.4.3	Examples .....	5-62
5.8.5	How to Create the session.validate Task.....	5-63
5.8.5.1	Using Parameters.....	5-63
5.8.5.2	Specifying Parameters Specified as Nested Elements.....	5-63
5.8.5.3	Examples .....	5-63
5.8.6	How to Create the mappings.export Task .....	5-64
5.8.6.1	Using Parameters.....	5-64
5.8.6.2	Specifying Parameters Specified as Nested Elements.....	5-64
5.8.6.3	Examples .....	5-64
5.8.7	How to Create the classpath Task .....	5-65
5.8.7.1	Using Parameters.....	5-65
5.8.7.2	Specifying Parameters Specified as Nested Elements.....	5-65
5.8.7.3	Examples .....	5-65
5.8.8	How to Create the ignoreerror Task .....	5-66
5.8.8.1	Using Parameters.....	5-66
5.8.8.2	Specifying Parameters Specified as Nested Elements.....	5-66

5.8.8.3	Examples .....	5-66
5.8.9	How to Create the ignoreerrorset Task .....	5-66
5.8.9.1	Using Parameters.....	5-66
5.8.9.2	Specifying Parameters Specified as Nested Elements.....	5-66
5.8.9.3	Examples .....	5-67
5.8.10	How to Create the loginspec Task.....	5-67
5.8.10.1	Using Parameters.....	5-67
5.8.10.2	Specifying Parameters Specified as Nested Elements.....	5-68
5.8.10.3	Examples .....	5-68

## 6 Using the Schema Manager

6.1	Introduction to the Schema Manager.....	6-1
6.1.1	How to Use Schema Manager Java and Database Type Conversion.....	6-3
6.1.2	How to Use Sequencing .....	6-3
6.2	Creating a Table Creator .....	6-4
6.2.1	How to Use TopLink Workbench During Development.....	6-4
6.2.2	How to Use the Default Table Generator at Run Time .....	6-4
6.2.3	How to Use Java to Create a Table Creator .....	6-5
6.2.3.1	Creating a TableCreator Class .....	6-5
6.2.3.2	Creating a TableDefinition Class.....	6-5
6.2.3.3	Adding Fields to a TableDefinition.....	6-5
6.2.3.4	Defining Sybase and Microsoft SQL Server Native Sequencing .....	6-6
6.3	Creating Tables with a Table Creator .....	6-6
6.4	Creating Database Tables Automatically .....	6-6
6.4.1	Creating Database Tables Automatically in JPA Projects.....	6-6
6.4.2	Creating Database Tables Automatically in EJB CMP Projects .....	6-7

## 7 Using an Integrated Development Environment

7.1	Configuring TopLink for Oracle JDeveloper .....	7-1
7.1.1	How to Use TopLink Mappings .....	7-1
7.2	Configuring TopLink Workbench with Source Control Management Software .....	7-3
7.2.1	How to Use a Source Control Management System.....	7-3
7.2.2	How to Merge Files .....	7-4
7.2.2.1	Merging Project Files.....	7-4
7.2.2.2	Merging Table, Descriptor, and Class Files .....	7-5
7.2.3	How to Share Project Objects.....	7-6
7.2.4	How to Manage the ejb-jar.xml File .....	7-6
7.2.5	How to Work with Locked Files.....	7-6

## Part III TopLink Application Deployment

### 8 Integrating TopLink with an Application Server

8.1	Introduction to the Application Server Support .....	8-1
8.2	Integrating TopLink with an Application Server.....	8-2
8.2.1	What Are the Software Requirements .....	8-2
8.2.2	How to Configure the XML Parser Platform.....	8-3

8.2.2.1	Configuring XML Parser Platform.....	8-3
8.2.2.2	Creating an XML Parser Platform.....	8-4
8.2.2.3	XML Parser Limitations.....	8-4
8.2.3	How to Set Security Permissions.....	8-4
8.2.4	How to Migrate the Persistence Manager.....	8-4
8.2.5	How to Integrate Clustering.....	8-5
8.3	Integrating TopLink with Oracle WebLogic Server.....	8-5
8.3.1	How to Configure Classpath.....	8-5
8.3.2	How to Integrate JTA.....	8-6
8.3.3	How to Integrate JMX.....	8-6
8.3.4	How to Integrate the Security Manager.....	8-6
8.4	Integrating TopLink with Oracle Containers for Java EE (OC4J).....	8-7
8.4.1	How to Integrate CMP.....	8-7
8.4.2	How to Migrate OC4J Orion CMP Persistence to OC4J TopLink Persistence.....	8-7
8.4.2.1	What You May Need to Know About Migrating OC4J Orion Persistence to OC4J TopLink Persistence	8-8
8.4.2.2	Using the TopLink Migration Tool from TopLink Workbench.....	8-11
8.4.2.3	Using the TopLink Migration Tool from the Command Line.....	8-12
8.4.2.4	Performing Post-Migration Changes.....	8-14
8.4.2.4.1	EJB 2.1 Persistence Manager Customization.....	8-15
8.4.2.4.2	Session Event Listener.....	8-15
8.4.2.5	Troubleshooting Your Migration.....	8-15
8.4.3	How to Integrate JTA.....	8-16
8.4.4	How to Integrate with Oracle Application Server Manageability and Diagnosability.....	8-16
8.5	Integrating TopLink with IBM WebSphere Application Server.....	8-16
8.5.1	How to Configure Classpath.....	8-17
8.5.1.1	Configuring Classpath for IBM WebSphere Application Server 6.1 and Later	8-17
8.5.2	How to Configure Class Loader Order.....	8-17
8.5.3	How to Integrate JTA.....	8-17
8.5.4	How to Configure Clustering on IBM WebSphere Application Server.....	8-17
8.6	Integrating TopLink with Sun Application Server.....	8-17
8.6.1	How to Configure Classpath.....	8-18
8.6.2	How to Integrate JTA.....	8-18
8.7	Integrating TopLink with JBoss Application Server.....	8-18
8.7.1	How to Configure Classpath.....	8-18
8.7.2	How to Integrate JTA.....	8-18
8.7.3	How to Configure JPA Application Deployment to JBoss 4.2 Application Server.	8-18
8.8	Defining Security Permissions.....	8-19
8.8.1	How to Define Permissions Required by TopLink Features.....	8-19
8.8.1.1	Defining System Properties.....	8-20
8.8.1.2	Loading project.xml or sessions.xml Files.....	8-20
8.8.1.3	Defining Cache Coordination.....	8-20
8.8.1.4	Accessing a Data Source by Port.....	8-20
8.8.1.5	Logging with java.util.logging.....	8-21
8.8.1.6	Granting Permissions for Java EE Application Deployment.....	8-21
8.8.2	How to Define Permissions Required when doPrivileged Is Disabled.....	8-21
8.8.3	How to Disable doPrivileged Operation.....	8-21

8.9	Configuring Miscellaneous EJB CMP Options .....	8-21
8.9.1	How to Configure EJB CMP Setter Parameter Type Checking.....	8-22
8.9.2	How to Configure EJB CMP Unknown Primary Key Class Support.....	8-22
8.9.3	How to Configure EJB CMP Single-Object Finder Return Type Checking.....	8-22

## 9 Creating TopLink Files for Deployment

9.1	Introduction to the TopLink Deployment File Creation .....	9-1
9.1.1	project.xml File .....	9-2
9.1.1.1	XSD File Format .....	9-2
9.1.1.2	POJO Applications and Project Metadata .....	9-2
9.1.1.3	JPA Applications and Project Metadata .....	9-2
9.1.1.4	CMP Applications and Project Metadata .....	9-3
9.1.1.5	Creating the project.xml File with Oracle JDeveloper .....	9-3
9.1.1.6	Creating the project.xml File with TopLink Workbench .....	9-3
9.1.1.7	Creating project.xml Programatically .....	9-3
9.1.2	sessions.xml File .....	9-4
9.1.2.1	XSD File Format .....	9-4
9.1.2.2	POJO Applications and Session Metadata .....	9-4
9.1.2.3	JPA Applications and Session Metadata .....	9-4
9.1.2.4	CMP Applications and Session Metadata .....	9-5
9.1.3	ejb-jar.xml File .....	9-5
9.1.4	<i>JAVA-EE-CONTAINER</i> -ejb-jar.xml File .....	9-6
9.1.4.1	Oracle WebLogic Server and the weblogic-ejb-jar.xml File .....	9-6
9.1.4.2	OC4J and the orion-ejb-jar.xml File .....	9-6
9.1.5	toplink-ejb-jar.xml File .....	9-7
9.1.5.1	OC4J and the toplink-ejb-jar.xml File .....	9-7
9.2	Creating Deployment Files for Java Applications .....	9-7
9.3	Creating Deployment Files for JavaServer Pages and Servlet Applications .....	9-7
9.4	Creating Deployment Files for Session Bean Applications .....	9-7
9.4.1	How to Create Deployment Files for EJB 1. <i>n</i> and 2. <i>n</i> Session Bean Applications .....	9-8
9.4.2	How to Create Deployment Files for EJB 3.0 Session Bean Applications .....	9-8
9.5	Creating Deployment Files for JPA Applications .....	9-8
9.6	Creating Deployment Files for CMP Applications .....	9-8
9.7	Creating Deployment Files for BMP Applications .....	9-9
9.8	Configuring the weblogic-ejb-jar.xml File for Oracle WebLogic Server .....	9-9
9.8.1	What You May Need to Know About Unsupported weblogic-ejb-jar.xml File Tags .....	9-9
9.9	Configuring the orion-ejb-jar.xml File for OC4J .....	9-10
9.9.1	How to Configure persistence-manager Entries .....	9-10
9.9.1.1	Configuring pm-properties .....	9-10
9.9.1.2	Configuring cache-synchronization Properties .....	9-12
9.9.1.3	Configuring default-mapping Properties .....	9-12

## 10 Packaging a TopLink Application

10.1	Packaging Java Applications .....	10-1
10.2	Packaging JavaServer Pages and Servlet Applications .....	10-2
10.2.1	How to Create the TopLink Domain JAR .....	10-3

10.3	Packaging Session Bean Applications.....	10-3
10.3.1	How to Package an EJB 1. <i>n</i> and 2. <i>n</i> Session Bean Application.....	10-3
10.3.2	How to Package an EJB 3.0 Session Bean Application.....	10-4
10.3.3	How to Create the TopLink Domain JAR.....	10-4
10.3.4	How to Create the EJB JAR.....	10-5
10.4	Packaging JPA Applications.....	10-5
10.5	Packaging a POJO Application for Weaving.....	10-5
10.6	Packaging CMP Applications.....	10-5
10.6.1	How to Create the EJB JAR.....	10-6
10.7	Packaging BMP Applications.....	10-6
10.7.1	How to Create the TopLink Domain JAR.....	10-7
10.7.2	How to Create EJB JAR File.....	10-8
10.8	Packaging with TopLink Metadata File Resource Paths.....	10-8

## 11 Deploying a TopLink Application

11.1	Deploying Java Applications.....	11-1
11.2	Deploying JavaServer Pages and Servlets.....	11-1
11.3	Deploying Session Bean Applications.....	11-2
11.4	Deploying JPA Applications.....	11-2
11.5	Deploying CMP Applications.....	11-2
11.5.1	How to Deploy a CMP Application to OC4J.....	11-2
11.6	Deploying BMP Applications.....	11-2
11.7	Performing Hot Deployment of EJB.....	11-3
11.7.1	How to Perform Hot Deployment in a CMP Application.....	11-3
11.7.2	How to Perform Hot Deployment in a POJO Application.....	11-3

## Part IV Optimization and Customization of a TopLink Application

### 12 Optimizing the TopLink Application

12.1	Introduction to Optimization.....	12-1
12.2	Identifying Sources of Application Performance Problems.....	12-2
12.3	Measuring TopLink Performance with the TopLink Profiler.....	12-2
12.3.1	How to Configure the TopLink Performance Profiler.....	12-3
12.3.2	How to Access the TopLink Profiler Results.....	12-4
12.4	Measuring TopLink Performance with the Oracle Dynamic Monitoring System (DMS).....	12-4
12.4.1	How to Configure the Oracle DMS Profiler.....	12-8
12.4.1.1	Configuring the Oracle DMS Profiler in a TopLink CMP Application on OC4J.....	12-9
12.4.1.2	Configuring the Oracle DMS Profiler in a EclipseLink JPA Application on OC4J.....	12-9
12.4.2	How to Access Oracle DMS Profiler Data Using JMX.....	12-9
12.4.3	How to Access Oracle DMS Profiler Data Using the DMS Spy Servlet.....	12-10
12.5	Identifying General Performance Optimization.....	12-11
12.6	Optimizing for a Production Environment.....	12-11
12.7	Optimizing Schema.....	12-11
12.7.1	Schema Case 1: Aggregation of Two Tables Into One.....	12-12

12.7.2	Schema Case 2: Splitting One Table Into Many .....	12-12
12.7.3	Schema Case 3: Collapsed Hierarchy .....	12-14
12.7.4	Schema Case 4: Choosing One Out of Many .....	12-15
12.8	Optimizing Mappings and Descriptors.....	12-16
12.9	Optimizing Sessions .....	12-16
12.10	Optimizing Cache .....	12-16
12.11	Optimizing Data Access.....	12-17
12.11.1	How to Optimize JDBC Driver Properties.....	12-17
12.11.2	How to Optimize Data Format .....	12-18
12.11.3	How to Use Batch Writing for Optimization.....	12-18
12.11.4	How to Use Outer-Join Reading with Inherited Subclasses.....	12-19
12.11.5	How to Use Parameterized SQL (Parameter Binding) and Prepared Statement Caching for Optimization 12-19	
12.12	Optimizing Queries .....	12-21
12.12.1	How to Use Parameterized SQL and Prepared Statement Caching for Optimization.....	12-21
12.12.2	How to Use Named Queries for Optimization .....	12-21
12.12.3	How to Use Batch and Join Reading for Optimization.....	12-21
12.12.4	How to Use Partial Object Queries and Fetch Groups for Optimization .....	12-22
12.12.5	How to Use Read-Only Queries for Optimization .....	12-22
12.12.6	How to Use JDBC Fetch Size for Optimization.....	12-22
12.12.7	How to Use Cursored Streams and Scrollable Cursors for Optimization.....	12-23
12.12.8	How to Use Result Set Pagination for Optimization.....	12-24
12.12.9	Read Optimization Examples .....	12-24
12.12.9.1	Reading Case 1: Displaying Names in a List.....	12-26
12.12.9.1.1	Partial Object Reading .....	12-26
12.12.9.1.2	Report Query.....	12-27
12.12.9.1.3	Fetch Groups .....	12-28
12.12.9.2	Reading Case 2: Batch Reading Objects .....	12-28
12.12.9.3	Reading Case 3: Using Complex Custom SQL Queries.....	12-30
12.12.9.4	Reading Case 4: Using View Objects .....	12-30
12.12.9.5	Reading Case 5: Inheritance Subclass Outer-Joining .....	12-31
12.12.10	Write Optimization Examples .....	12-32
12.12.10.1	Writing Case: Batch Writes .....	12-33
12.12.10.1.1	Cursors .....	12-34
12.12.10.1.2	Batch Writing and Parameterized SQL.....	12-34
12.12.10.1.3	Sequence Number Preallocation .....	12-34
12.12.10.1.4	Multiprocessing .....	12-35
12.13	Optimizing the Unit of Work.....	12-35
12.14	Optimizing Using Weaving.....	12-36
12.15	Optimizing the Application Server and Database Optimization .....	12-36
12.16	Optimizing Storage and Retrieval of Binary Data in XML.....	12-36
12.16.1	How to Use an Attachment Marshaller and Unmarshaller.....	12-39

## 13 Customizing the TopLink Application

13.1	Introduction to Customization.....	13-1
13.2	Creating Custom Data Types .....	13-1

13.3	Using Public Source .....	13-2
13.4	Using the Session Customizer Class .....	13-2
13.5	Using the Descriptor Customizer Class.....	13-2
13.6	Using the Descriptor Amendment Methods.....	13-3
13.7	Using EclipseLink JPA Extensions .....	13-3

## Part V Mapping and Configuration Overview

### 14 Introduction to TopLink Mapping and Configuration

14.1	Mapping and Configuration Concepts .....	14-1
14.1.1	Projects .....	14-1
14.1.2	Descriptors.....	14-2
14.1.3	Mappings .....	14-2

## Part VI Projects

### 15 Introduction to Projects

15.1	TopLink Project Types .....	15-1
15.2	Project Concepts.....	15-2
15.2.1	Project Architecture .....	15-2
15.2.2	Relational and Nonrelational Projects .....	15-2
15.2.3	Persistent and Nonpersistent Projects .....	15-2
15.2.4	Projects and Login .....	15-3
15.2.4.1	POJO Session Role .....	15-3
15.2.4.2	CMP Deployment Role .....	15-3
15.2.4.3	Development Role .....	15-4
15.2.5	Projects and Platforms .....	15-4
15.2.6	Projects and Sequencing .....	15-4
15.2.6.1	Configuring How to Obtain Sequence Values .....	15-5
15.2.6.2	Configuring Where to Write Sequence Values .....	15-5
15.2.7	XML Namespaces .....	15-5
15.3	Project API .....	15-6
15.3.1	Project Inheritance Hierarchy .....	15-6
15.4	XML Namespaces Overview.....	15-6
15.4.1	TopLink Workbench Namespace Resolution.....	15-6
15.4.2	Element and Attribute Form Options.....	15-6
15.4.2.1	Element Form Default Qualified and Attribute Form Default Unqualified .....	15-6
15.4.2.2	Element and Attribute Form Default Unqualified .....	15-7
15.4.2.3	Element and Attribute Form Default Qualified .....	15-9
15.4.3	TopLink Runtime Namespace Resolution .....	15-10

## Part VII Descriptors

### 16 Introduction to Descriptors

16.1	Descriptor Types .....	16-1
16.2	Descriptor Concepts .....	16-2



16.2.1	Descriptor Architecture .....	16-2
16.2.2	Descriptors and Inheritance .....	16-3
16.2.3	Descriptors and CMP and BMP .....	16-3
16.2.3.1	Nondeferred Changes .....	16-3
16.2.3.2	Creating a New Entity Bean and ejbCreate / ejbPostCreate Methods .....	16-4
16.2.3.3	Inheritance .....	16-5
16.2.4	Fetch Groups .....	16-5
16.2.5	Descriptors and Aggregation .....	16-5
16.2.6	Descriptor Customization .....	16-5
16.2.7	Amendment and After-Load Methods .....	16-6
16.2.8	Descriptor Event Manager .....	16-6
16.2.9	Descriptor Query Manager .....	16-6
16.2.10	Descriptors and Sequencing .....	16-7
16.2.11	Descriptors and Locking .....	16-7
16.2.12	Default Root Element .....	16-7
16.3	Descriptors and Inheritance .....	16-9
16.3.1	How to Specify a Class Indicator .....	16-10
16.3.1.1	Using Class Indicator Fields .....	16-10
16.3.1.2	Using Class Extraction Methods .....	16-11
16.3.1.2.1	Specifying Expressions for Only-Instances and With-All-Subclasses .....	16-12
16.3.2	Inheritance and Primary Keys .....	16-12
16.3.3	Single and Multi-Table Inheritance .....	16-12
16.3.4	Aggregate and Composite Descriptors and Inheritance .....	16-12
16.3.5	Inheritance and CMP and BMP .....	16-13
16.4	Descriptors and Locking .....	16-13
16.4.1	Optimistic Version Locking Policies .....	16-13
16.4.2	Optimistic Version Locking Policies and Cascading .....	16-14
16.4.3	Optimistic Locking and Rollbacks .....	16-16
16.4.4	Optimistic Field Locking Policies .....	16-16
16.4.5	Pessimistic Locking Policy .....	16-17
16.4.6	Locking in a Three-Tier Application .....	16-18
16.4.6.1	Optimistic Locking in a Three-Tier Application .....	16-18
16.4.6.2	Pessimistic Locking in a Three-Tier Application .....	16-18
16.5	Descriptor API .....	16-18
16.5.1	Descriptor Inheritance Hierarchy .....	16-19

## Part VIII Mappings

### 17 Introduction to Mappings

17.1	Mapping Types .....	17-1
17.2	Mapping Concepts .....	17-2
17.2.1	Mapping Architecture .....	17-3
17.2.2	Example Mapping .....	17-4
17.2.3	Automatic Mappings .....	17-4
17.2.3.1	JPA Automapping .....	17-5
17.2.3.2	Automapping with Oracle JDeveloper at Development Time .....	17-5

17.2.3.3	Automapping with TopLink Workbench at Development Time.....	17-5
17.2.3.4	Default Mapping in EJB 2. <i>n</i> CMP Projects Using OC4J at Run Time .....	17-5
17.2.3.5	JAXB Project Generation at Development Time .....	17-6
17.2.4	Indirection (Lazy Loading).....	17-6
17.2.4.1	Value Holder Indirection.....	17-8
17.2.4.2	Transparent Indirect Container Indirection .....	17-9
17.2.4.3	Proxy Indirection .....	17-10
17.2.4.3.1	Proxy Indirection Restrictions .....	17-10
17.2.4.4	Weaved Indirection .....	17-11
17.2.4.5	Indirection and JPA .....	17-11
17.2.4.6	Indirection and EJB 2. <i>n</i> CMP .....	17-11
17.2.4.7	Indirection, Serialization, and Detachment .....	17-11
17.2.5	Method Accessors and Attribute Accessors .....	17-12
17.2.6	Mapping Converters and Transformers.....	17-12
17.2.6.1	Serialized Object Converter .....	17-12
17.2.6.2	Type Conversion Converter.....	17-13
17.2.6.3	Object Type Converter .....	17-14
17.2.6.4	Simple Type Translator.....	17-15
17.2.6.4.1	Default Read Conversions .....	17-16
17.2.6.4.2	Default Write Conversions.....	17-16
17.2.6.5	Transformation Mappings .....	17-17
17.2.7	Mappings and XPath.....	17-17
17.2.7.1	XPath by Position .....	17-18
17.2.7.2	XPath by Path and Name .....	17-18
17.2.7.3	XPath by Name .....	17-19
17.2.7.4	Self XPath.....	17-19
17.2.8	Mappings and xsd:list and xsd:union Types.....	17-20
17.2.8.1	Mapping an xsd:union Type.....	17-20
17.2.8.2	Mapping an xsd:list Type.....	17-21
17.2.8.3	Mapping a List of Unions.....	17-21
17.2.8.4	Mapping a Union of Lists.....	17-22
17.2.8.5	Mapping a Union of Unions .....	17-22
17.2.9	Mappings and the jaxb:class Customization .....	17-23
17.2.9.1	all, choice, or sequence Structure .....	17-23
17.2.9.2	group Structure .....	17-24
17.2.9.3	sequence or choice Structure Containing a group.....	17-24
17.2.9.4	group Structure Containing a sequence or choice.....	17-25
17.2.9.5	group Structure Containing a group .....	17-26
17.2.9.6	Limitations of jaxb:class Customization Support .....	17-26
17.2.10	Mappings and JAXB Typesafe Enumerations .....	17-27
17.3	Mapping API .....	17-28
17.4	Relational Mappings.....	17-28
17.5	Object-Relational Data Type Mappings.....	17-28
17.6	XML Mappings.....	17-29
17.7	EIS Mappings.....	17-29

## Part IX Relational Projects

## 18 Introduction to Relational Projects

18.1	Building Relational Projects .....	18-1
18.1.1	How to Build Relational Projects for a Relational Database .....	18-1
18.1.2	How to Build Relational Projects for an Object-Relational Data Type Database ....	18-2
18.2	Sequencing in Relational Projects.....	18-3
18.2.1	Sequencing Configuration Options.....	18-4
18.2.2	Sequencing Types .....	18-4
18.2.2.1	Table Sequencing.....	18-5
18.2.2.1.1	Default Versus Custom Sequence Table .....	18-5
18.2.2.2	Unary Table Sequencing.....	18-6
18.2.2.3	Query Sequencing .....	18-6
18.2.2.4	Default Sequencing .....	18-7
18.2.2.5	Native Sequencing with an Oracle Database Platform.....	18-7
18.2.2.5.1	Understanding the Oracle SEQUENCE Object.....	18-7
18.2.2.5.2	Using SEQUENCE Objects .....	18-8
18.2.2.6	Native Sequencing with a Non-Oracle Database Platform .....	18-8
18.2.3	Sequencing and Preallocation Size.....	18-9
18.2.4	Sequencing with EJB 2.n Entity Beans with Container-Managed Persistence.....	18-10

## 19 Creating a Relational Project

19.1	Introduction to the Relational Project Creation.....	19-1
19.2	Creating a Project from an Existing Object and Data Model.....	19-2
19.2.1	How to Create a Project from an Existing Object and Data Model Using TopLink Workbench	19-2
19.3	Creating a Project from an Existing Object Model.....	19-2
19.3.1	How to Create a Project from an Existing Object Model Using TopLink Workbench.....	19-2
19.4	Creating a Project from an Existing Data Model.....	19-3
19.4.1	How to Create a Project from an Existing Data Model Using TopLink Workbench.....	19-3
19.5	Creating a Project from an OC4J EJB CMP EAR at Deployment Time.....	19-3
19.6	Exporting Project Information .....	19-3
19.6.1	How to Export Project Java Source Using TopLink Workbench.....	19-3
19.6.2	How to Export Table Creator Files Using TopLink Workbench .....	19-4
19.7	Working with the ejb-xml.File .....	19-4
19.7.1	How to Write to the ejb-jar.xml File Using TopLink Workbench.....	19-5
19.7.2	How to Read from the ejb-jar.xml File Using TopLink Workbench .....	19-6

## 20 Configuring a Relational Project

20.1	Introduction to Relational Project Configuration.....	20-1
20.2	Configuring Relational Database Platform at the Project Level .....	20-3
20.2.1	How to Configure Relational Database Platform at the Project Level Using TopLink Workbench	20-3
20.3	Configuring Sequencing at the Project Level.....	20-3
20.3.1	How to Configure Sequencing at the Project Level Using TopLink Workbench....	20-4
20.3.2	How to Configure Sequencing at the Project Level Using Java.....	20-5

20.4	Configuring Login Information at the Project Level .....	20-5
20.4.1	How to Configure Login Information at the Project Level Using TopLink Workbench ...	20-5
20.5	Configuring Development and Deployment Logins .....	20-6
20.5.1	How to Configure Development and Deployment Logins Using TopLink Workbench..	20-7
20.6	Logging In to the Database .....	20-8
20.7	Configuring Named Query Parameterized SQL and Statement Caching at the Project Level	20-8
20.7.1	How to Configure Named Query Parameterized SQL and Statement Caching at the Project Level Using TopLink Workbench	20-9
20.8	Configuring Table Generation Options .....	20-10
20.8.1	How to Configure Table Generation Options Using TopLink Workbench .....	20-10
20.9	Configuring Table Creator Java Source Options .....	20-11
20.9.1	How to Configure Table Creator Java Source Options Using TopLink Workbench .....	20-11
20.10	Configuring Project Java Source Code Options .....	20-12
20.10.1	How to Configure Project Java Source Code Options Using TopLink Workbench .....	20-12
20.11	Configuring Deprecated Direct Mappings .....	20-13
20.11.1	How to Configure Deprecated Direct Mappings Using TopLink Workbench .....	20-14

## Part X Relational Descriptors

### 21 Introduction to Relational Descriptors

21.1	Relational Descriptors .....	21-1
21.2	Aggregate and Composite Descriptors in Relational Projects .....	21-1
21.2.1	Relational Aggregates and Nesting .....	21-2
21.2.2	Relational Aggregates and Inheritance .....	21-3
21.2.3	Relational Aggregates and EJB 2.n Entity Beans .....	21-4
21.3	Descriptors and Inheritance in Relational Projects .....	21-4
21.3.1	Inheritance and Primary Keys in Relational Projects .....	21-4
21.3.2	Single- and Multi-Table Inheritance in Relational Projects .....	21-4
21.3.2.1	Single-Table Inheritance .....	21-4
21.3.2.2	Multi-Table Inheritance .....	21-5
21.3.2.2.1	Inheritance Outer-Joins .....	21-6

### 22 Creating a Relational Descriptor

22.1	Introduction to Relational Descriptor Creation .....	22-1
22.2	Creating a Relational Descriptor .....	22-1
22.2.1	How to Create a Relational Descriptor Using TopLink Workbench .....	22-1
22.2.1.1	Creating Relational Class Descriptors .....	22-2
22.2.1.2	Creating Relational Aggregate Descriptors .....	22-2
22.2.1.3	Creating Relational Interface Descriptors .....	22-2
22.2.2	How to Create a Relational Descriptor Using Java .....	22-2

## 23 Configuring a Relational Descriptor

23.1	Introduction to Relational Descriptor Configuration .....	23-1
23.2	Configuring Associated Tables .....	23-3
23.2.1	How to Configure Associated Tables Using TopLink Workbench .....	23-3
23.2.2	How to Configure Associated Tables Using Java .....	23-4
23.3	Configuring Sequencing at the Descriptor Level .....	23-4
23.3.1	How to Configure Sequencing at the Descriptor Level Using TopLink Workbench .....	23-4
23.3.2	How to Configure Sequencing at the Descriptor Level Using Java .....	23-6
23.3.2.1	Configuring a Sequence by Name .....	23-6
23.3.2.2	Configuring the Same Sequence for Multiple Descriptors .....	23-6
23.3.2.3	Configuring the Platform Default Sequence .....	23-7
23.4	Configuring Custom SQL Queries for Basic Persistence Operations .....	23-7
23.4.1	How to Configure Custom SQL Queries for Basic Persistence Operations Using TopLink Workbench .....	23-8
23.4.2	How to Configure Custom SQL Queries for Basic Persistence Operations Using Java .....	23-9
23.5	Configuring Interface Alias .....	23-11
23.5.1	How to Configure Interface Alias Using TopLink Workbench .....	23-12
23.5.2	How to Configure Interface Alias Using Java .....	23-12
23.6	Configuring a Relational Descriptor as a Class or Aggregate Type .....	23-12
23.6.1	How to Configure a Relational Descriptor as a Class or Aggregate Type Using TopLink Workbench .....	23-13
23.6.2	How to Configure a Relational Descriptor as a Class or Aggregate Type Using Java .....	23-13
23.7	Configuring Multitable Information .....	23-14
23.7.1	How to Configure Multitable Information Using TopLink Workbench .....	23-14
23.7.2	How to Configure Multitable Information Using Java .....	23-15

## Part XI Object-Relational Data Type Descriptors

### 24 Introduction to Object-Relational Data Type Descriptors

24.1	Object-Relational Data Type Descriptors .....	24-1
------	---	------

### 25 Creating an Object-Relational Data Type Descriptor

25.1	Introduction to Object-Relational Data Type Descriptor Creation .....	25-1
25.2	Creating an Object-Relational Data Type Descriptor .....	25-1
25.2.1	How to Create an Object-Relational Data Type Descriptor Using Java .....	25-1

### 26 Configuring an Object-Relational Data Type Descriptor

26.1	Introduction to Object-Relational Data Type Descriptor Configuration .....	26-1
26.2	Configuring Field Ordering .....	26-2
26.2.1	How to Configure Field Ordering Using Java .....	26-2

## Part XII Relational Mappings

## 27 Introduction to Relational Mappings

27.1	Relational Mapping Types.....	27-2
27.2	Relational Mapping Concepts.....	27-2
27.2.1	Directionality .....	27-2
27.2.2	Converters and Transformers .....	27-3
27.2.2.1	Using a Direct Mapping .....	27-3
27.2.2.2	Using a Converter Mapping .....	27-3
27.2.2.3	Using a Transformation Mapping.....	27-4
27.2.3	Relational Mappings and EJB 2. <i>n</i> CMP .....	27-4
27.3	Direct-to-Field Mapping .....	27-4
27.4	Direct-to-XMLType Mapping .....	27-5
27.5	One-to-One Mapping .....	27-5
27.5.1	One-to-One Mappings and EJB 2. <i>n</i> CMP .....	27-6
27.6	Variable One-to-One Mapping .....	27-7
27.7	One-to-Many Mapping .....	27-8
27.7.1	One-to-Many Mappings and EJB 2. <i>n</i> CMP .....	27-9
27.8	Many-to-Many Mapping .....	27-9
27.8.1	Many-to-Many Mappings and EJB 2. <i>n</i> CMP .....	27-10
27.9	Aggregate Collection Mapping.....	27-10
27.9.1	Aggregate Collection Mappings and Inheritance .....	27-11
27.9.2	Aggregate Collection Mappings and EJB.....	27-11
27.9.3	How to Implement Aggregate Collection Mappings .....	27-11
27.10	Direct Collection Mapping .....	27-12
27.11	Direct Map Mapping .....	27-13
27.12	Aggregate Object Mapping .....	27-13
27.12.1	Aggregate Object Mappings with a Single Source Object .....	27-14
27.12.2	Aggregate Object Mappings with Multiple Source Objects .....	27-15
27.12.3	How to Implement an Aggregate Object Relationship Mapping.....	27-15
27.13	Transformation Mapping.....	27-16

## 28 Configuring a Relational Mapping

28.1	Introduction to Relational Mapping Configuration .....	28-1
28.2	Configuring Common Relational Mapping Options.....	28-2
28.3	Configuring a Database Field.....	28-3
28.3.1	How to Configure a Database Field Using TopLink Workbench.....	28-4
28.4	Configuring Reference Descriptor.....	28-5
28.4.1	How to Configure a Reference Descriptor Using TopLink Workbench.....	28-6
28.5	Configuring Batch Reading .....	28-7
28.5.1	How to Configure Batch Reading Using TopLink Workbench .....	28-8
28.5.2	How to Configure Batch Reading Using Java .....	28-8
28.6	Configuring Query Key Order.....	28-8
28.6.1	How to Configure Query Key Order Using TopLink Workbench.....	28-9
28.6.2	How to Configure Query Key Order Using Java.....	28-9
28.7	Configuring Table and Field References (Foreign and Target Foreign Keys) .....	28-10
28.7.1	How to Configure Table and Field References (Foreign and Target Foreign Keys) Using TopLink Workbench	28-11

28.7.2	How to Configure Table and Field References (Foreign and Target Foreign Keys) Using Java	28-12
28.8	Configuring Joining at the Mapping Level .....	28-13
28.8.1	How to Configure Joining at the Mapping Level Using TopLink Workbench .....	28-13
28.8.2	How to Configure Joining at the Mapping Level Using Java .....	28-14
<b>29</b>	<b>Configuring a Relational Direct-to-Field Mapping</b>	
29.1	Introduction to Relational Direct-to-Field Mapping Configuration .....	29-1
<b>30</b>	<b>Configuring a Relational Direct-to-XMLType Mapping</b>	
30.1	Introduction to Relational Direct-to-XMLType Mapping.....	30-1
30.2	Configuring Read Whole Document.....	30-2
30.2.1	How to Configure Read Whole Document Using TopLink Workbench.....	30-2
30.2.2	How to Configure Read Whole Document Using Java.....	30-2
<b>31</b>	<b>Configuring a Relational One-to-One Mapping</b>	
31.1	Introduction to Relational One-to-One Mapping Configuration .....	31-1
<b>32</b>	<b>Configuring a Relational Variable One-to-One Mapping</b>	
32.1	Introduction to Relational Variable One-to-One Mapping Configuration.....	32-1
32.2	Configuring Class Indicator .....	32-2
32.2.1	How to Configure a Class Indicator Using TopLink Workbench .....	32-3
32.2.2	How to Configure a Class Indicator Using Java .....	32-3
32.3	Configuring Unique Primary Key .....	32-4
32.3.1	How to Configure a Unique Primary Key Using TopLink Workbench.....	32-4
32.3.2	How to Configure a Unique Primary Key Using Java .....	32-5
32.3.3	What You May Need to Know About Unique Primary Keys.....	32-5
32.4	Configuring Query Key Association.....	32-6
32.4.1	How to Configure a Query Key Association Using TopLink Workbench.....	32-6
32.4.2	How to Configure a Query Key Association Using Java.....	32-6
<b>33</b>	<b>Configuring a Relational One-to-Many Mapping</b>	
33.1	Introduction to Relational One-to-Many Mapping Configuration.....	33-1
<b>34</b>	<b>Configuring a Relational Many-to-Many Mapping</b>	
34.1	Introduction to Relational Many-to-Many Mapping Configuration.....	34-1
34.2	Configuring a Relation Table .....	34-2
34.2.1	How to Configure a Relation Table Using TopLink Workbench .....	34-2
34.2.2	How to Configure a Relation Table Using Java .....	34-3
<b>35</b>	<b>Configuring a Relational Aggregate Collection Mapping</b>	
35.1	Introduction to Relational Aggregate Collection Mapping Configuration.....	35-1

<b>36</b>	<b>Configuring a Relational Direct Collection Mapping</b>	
36.1	Introduction to Relational Direct Collection Mapping Configuration.....	36-1
36.2	Configuring Target Table .....	36-2
36.2.1	How to Configure a Target Table Using TopLink Workbench .....	36-2
36.2.2	How to Configure a Target Table Using Java.....	36-3
36.3	Configuring Direct Value Field.....	36-4
36.3.1	How to Configure a Direct Value Field Using TopLink Workbench.....	36-4
36.3.2	How to Configure Direct Value Field Using Java.....	36-5
<b>37</b>	<b>Configuring a Relational Aggregate Object Mapping</b>	
37.1	Introduction to Relational Aggregate Object Mapping Configuration.....	37-1
37.2	Configuring Aggregate Fields.....	37-2
37.2.1	How to Configure Aggregate Fields Using TopLink Workbench.....	37-2
37.2.2	How to Configure Aggregate Fields Using Java.....	37-3
37.3	Configuring Allowing Null Values.....	37-3
37.3.1	How to Configure Allowing Null Values Using TopLink Workbench.....	37-3
37.3.2	How to Configure Allowing Null Values Using Java .....	37-4
<b>38</b>	<b>Configuring a Relational Direct Map Mapping</b>	
38.1	Introduction to Relational Direct Map Mapping Configuration .....	38-1
38.2	Configuring Direct Value Field.....	38-2
38.2.1	How to Configure Direct Value Fields Using TopLink Workbench.....	38-2
38.2.2	How to Configure Direct Value Fields Using Java .....	38-3
38.3	Configuring Direct Key Field.....	38-3
38.3.1	How to Configure Direct Key Field Using TopLink Workbench.....	38-3
38.3.2	How to Configure Direct Key Field Using Java .....	38-4
38.4	Configuring Key Converters .....	38-4
38.4.1	How to Configure Key Converters Using TopLink Workbench .....	38-4
38.4.2	How to Configure Key Converters Using Java .....	38-5
38.5	Configuring Value Converters.....	38-5
38.5.1	How to Configure Value Converters Using TopLink Workbench.....	38-5
<b>39</b>	<b>Configuring a Relational Transformation Mapping</b>	
39.1	Introduction to Relational Transformation Mapping Configuration.....	39-1

## **Part XIII Object-Relational Data Type Mappings**

<b>40</b>	<b>Introduction to Object-Relational Data Type Mappings</b>	
40.1	Object-Relational Data Type Mapping Types.....	40-1
40.1.1	Object-Relational Data Type Structure Mapping.....	40-2
40.1.2	Object-Relational Data Type Reference Mapping.....	40-2
40.1.3	Object-Relational Data Type Array Mapping.....	40-3
40.1.4	Object-Relational Data Type Object Array Mapping .....	40-3
40.1.5	Object-Relational Data Type Nested Table Mapping.....	40-3



<b>41</b>	<b>Configuring an Object-Relational Data Type Mapping</b>	
41.1	Introduction to Object-Relational Data Type Mapping Configuration .....	41-1
41.2	Configuring Common Object-Relational Data Type Mapping Options.....	41-2
41.3	Configuring Reference Class .....	41-2
41.3.1	How to Configure Reference Class Using Java .....	41-3
41.4	Configuring Attribute Name.....	41-3
41.4.1	How to Configure Attribute Name Using Java.....	41-4
41.5	Configuring Field Name .....	41-4
41.5.1	How to Configure Field Name Using Java .....	41-5
41.6	Configuring Structure Name.....	41-5
41.6.1	How to Configure Structure Name Using Java.....	41-6
<b>42</b>	<b>Configuring an Object-Relational Data Type Structure Mapping</b>	
42.1	Introduction to Object-Relational Data Type Structure Mapping Configuration .....	42-1
<b>43</b>	<b>Configuring an Object-Relational Data Type Reference Mapping</b>	
43.1	Introduction to Object-Relational Data Type Reference Mapping Configuration .....	43-1
<b>44</b>	<b>Configuring an Object-Relational Data Type Array Mapping</b>	
44.1	Introduction to Object-Relational Data Type Array Mapping Configuration .....	44-1
<b>45</b>	<b>Configuring an Object-Relational Data Type Object Array Mapping</b>	
45.1	Introduction to Object-Relational Data Type Object Array Mapping Configuration....	45-1
<b>46</b>	<b>Configuring an Object-Relational Data Type Nested Table Mapping</b>	
46.1	Introduction to Object-Relational Data Type Nested Table Mapping Configuration ...	46-1
<b>Part XIV</b>	<b>XML Projects</b>	
<b>47</b>	<b>Introduction to XML Projects</b>	
47.1	XML Project Concepts .....	47-1
47.1.1	TopLink Support for Java Architecture for XML Binding (JAXB).....	47-2
47.1.1.1	Generating TopLink Project and XML Schema Using JAXB Annotations.....	47-2
47.1.1.2	Working with JAXB-Specific Generated Files .....	47-4
47.1.1.2.1	Implementation Classes .....	47-4
47.1.1.3	Using TopLink JAXB Compiler-Generated Files at Run Time .....	47-5
47.1.1.3.1	How to Use TopLink XMLContext.....	47-5
47.1.1.3.2	How to Use Marshal and Unmarshal Events .....	47-5
47.1.1.3.3	How to Use TopLink XMLBinder.....	47-7
47.1.1.3.4	How to Use JAXBContext .....	47-8
47.1.1.3.5	How to Use JAXBElement.....	47-8
47.1.2	JAXB Validation .....	47-9

<b>48</b>	<b>Creating an XML Project</b>	
48.1	Introduction to XML Project Creation .....	48-1
48.2	Creating an XML Project from an XML Schema .....	48-1
48.2.1	How to Create an XML Project from an XML Schema Using TopLink Workbench .....	48-2
48.2.2	How to Create an XML Project from an XML Schema Using the Command Line. ....	48-3

## 49 Configuring an XML Project

49.1	Introduction to XML Project Configuration.....	49-1
------	--	------

## Part XV XML Descriptors

### 50 Introduction to XML Descriptors

50.1	XML Descriptor Concepts .....	50-1
50.1.1	XML Descriptors and Aggregation.....	50-1
50.1.1.1	Composite Descriptors in XML Projects .....	50-1

### 51 Creating an XML Descriptor

51.1	Introduction to XML Descriptor Creation.....	51-1
51.2	Creating an XML Descriptor .....	51-1
51.2.1	How to Create an XML Descriptor Using TopLink Workbench .....	51-1
51.2.2	How to Create an XML Descriptor Using Java .....	51-1

### 52 Configuring an XML Descriptor

52.1	Introduction to XML Descriptor Configuration.....	52-1
52.2	Configuring Schema Context for an XML Descriptor .....	52-2
52.2.1	How to Configure Schema Context for an XML Descriptor Using TopLink Workbench .....	52-2
52.2.1.1	Choosing a Schema Context.....	52-3
52.2.2	How to Configure Schema Context for an XML Descriptor Using Java .....	52-3
52.3	Configuring for Complex Type of anyType .....	52-3
52.3.1	How to Configure Complex Type of anyType Using TopLink Workbench.....	52-5
52.4	Configuring Default Root Element.....	52-5
52.4.1	How to Configure Default Root Element Using TopLink Workbench.....	52-5
52.4.1.1	Choosing a Root Element .....	52-6
52.5	Configuring Document Preservation.....	52-6
52.5.1	How to Configure Document Preservation Using TopLink Workbench.....	52-6
52.5.2	How to Configure Document Preservation Using Java .....	52-7

## Part XVI XML Mappings

### 53 Introduction to XML Mappings

53.1	XML Mapping Types.....	53-1
53.2	XML Mapping Concepts.....	53-3
53.2.1	Mapping to Simple and Complex Types.....	53-4

53.2.2	Mapping Order .....	53-5
53.2.3	XPath Support .....	53-5
53.2.4	xsd:list and xsd:union Support .....	53-5
53.2.5	xs:any and xs:anyType Support .....	53-5
53.2.6	jaxb:class Support .....	53-6
53.2.7	Typesafe Enumeration Support.....	53-6
53.2.8	Mapping Extensions .....	53-6
53.2.9	Key On Source-Based Mapping Support .....	53-6
53.2.10	Substitution Groups .....	53-7
53.2.11	Mixed Content Mapping .....	53-7
53.2.12	XML Adapter.....	53-7
53.3	XML Direct Mapping .....	53-7
53.3.1	Mapping to a Text Node .....	53-8
53.3.1.1	Mapping to a Simple Text Node .....	53-8
53.3.1.2	Mapping to a Text Node in a Simple Sequence .....	53-8
53.3.1.3	Mapping to a Text Node in a Subelement .....	53-9
53.3.1.4	Mapping to a Text Node by Position.....	53-10
53.3.2	Mapping to an Attribute.....	53-11
53.3.3	Mapping to a Specified Schema Type.....	53-12
53.3.4	Mapping to a List Field with an XML Direct Mapping .....	53-13
53.3.5	Mapping to a Union Field with an XML Direct Mapping .....	53-13
53.3.6	Mapping to a Union of Lists with an XML Direct Mapping .....	53-15
53.3.7	Mapping to a Union of Unions with an XML Direct Mapping.....	53-16
53.3.8	Mapping with a Simple Type Translator .....	53-17
53.4	XML Composite Direct Collection Mapping .....	53-18
53.4.1	Mapping to Multiple Text Nodes.....	53-18
53.4.1.1	Mapping to a Simple Sequence .....	53-18
53.4.1.2	Mapping to a Sequence in a Subelement .....	53-19
53.4.2	Mapping to Multiple Attributes .....	53-20
53.4.3	Mapping to a Single Text Node with an XML Composite Direct Collection Mapping ....	53-20
53.4.4	Mapping to a Single Attribute with an XML Composite Direct Collection Mapping.....	53-21
53.4.5	Mapping to a List of Unions with an XML Composite Direct Collection Mapping.....	53-22
53.4.6	Mapping to a Union of Lists with an XML Composite Direct Collection Mapping.....	53-23
53.4.7	Specifying the Content Type of a Collection with an XML Composite Direct Collection Mapping	53-24
53.5	XML Composite Object Mapping.....	53-24
53.5.1	Mapping into the Parent Record .....	53-25
53.5.2	Mapping to an Element.....	53-26
53.5.3	Mapping to Different Elements by Element Name .....	53-26
53.5.4	Mapping to Different Elements by Element Position.....	53-28
53.6	XML Composite Collection Mapping .....	53-29
53.7	XML Any Object Mapping .....	53-31
53.8	XML Any Collection Mapping.....	53-33
53.9	XML Transformation Mapping.....	53-35

53.10	XML Object Reference Mapping.....	53-36
53.10.1	Mapping Using a Single Key .....	53-37
53.10.2	Mapping Using a Composite Key .....	53-38
53.10.3	Mapping Using JAXB.....	53-38
53.11	XML Collection Reference Mapping.....	53-40
53.12	XML Binary Data Mapping .....	53-42
53.13	XML Binary Data Collection Mapping .....	53-43
53.14	XML Fragment Mapping .....	53-43
53.15	XML Fragment Collection Mapping .....	53-43
53.16	XML Choice Object Mapping.....	53-44
53.17	XML Choice Collection Mapping .....	53-44
53.18	XML Any Attribute Mapping .....	53-44

## 54 Configuring an XML Mapping

54.1	Introduction to XML Mapping Configuration .....	54-1
54.2	Configuring Common XML Mapping Options.....	54-2
54.3	Configuring Reference Descriptor.....	54-3
54.3.1	How to Configure a Reference Descriptor Using TopLink Workbench.....	54-4
54.4	Configuring Maps to Wildcard.....	54-5
54.4.1	How to Configure Maps to Wildcard Using TopLink Workbench.....	54-6
54.5	Configuring Source to Target Key Field Association .....	54-7
54.5.1	How to Configure Source to Target Key Field Association Using Java .....	54-8
54.6	Configuring Reference Class .....	54-8
54.6.1	How to Configure Reference Class Using Java .....	54-10
54.7	Configuring the Use of Inline Binary Data.....	54-10
54.7.1	How to Configure the Use of Inline Binary Data Using Java.....	54-11
54.8	Configuring the Use of SwaRef Type.....	54-11
54.8.1	How to Configure the Use of SwaRef Type Using Java.....	54-13
54.9	Configuring the Choice Element .....	54-13
54.9.1	How to Configure the Choice Element Using Java .....	54-14

## 55 Configuring an XML Direct Mapping

55.1	Introduction to XML Direct Mapping Configuration.....	55-1
------	---	------

## 56 Configuring an XML Composite Direct Collection Mapping

56.1	Introduction to XML Composite Direct Collection Mapping Configuration .....	56-1
------	---	------

## 57 Configuring an XML Composite Object Mapping

57.1	Introduction to XML Composite Object Mapping Configuration.....	57-1
------	---	------

## 58 Configuring an XML Composite Collection Mapping

58.1	Introduction to XML Composite Collection Mapping Configuration .....	58-1
------	--	------

## 59 Configuring an XML Any Object Mapping

59.1	Introduction to XML Any Object Mapping Configuration .....	59-1
------	--	------

<b>60</b>	<b>Configuring an XML Any Collection Mapping</b>	
60.1	Introduction to XML Any Collection Mapping Configuration.....	60-1
<b>61</b>	<b>Configuring an XML Transformation Mapping</b>	
61.1	Introduction to XML Transformation Mapping Configuration.....	61-1
<b>62</b>	<b>Configuring an XML Object Reference Mapping</b>	
62.1	Introduction to XML Object Reference Mapping.....	62-1
<b>63</b>	<b>Configuring an XML Collection Reference Mapping</b>	
63.1	Introduction to XML Collection Reference Mapping .....	63-1
<b>64</b>	<b>Configuring an XML Binary Data Mapping</b>	
64.1	Introduction to XML Binary Data Mapping .....	64-1
<b>65</b>	<b>Configuring an XML Binary Data Collection Mapping</b>	
65.1	Introduction to XML Binary Data Collection Mapping .....	65-1
<b>66</b>	<b>Configuring an XML Fragment Mapping</b>	
66.1	Introduction to XML Fragment Mapping .....	66-1
<b>67</b>	<b>Configuring an XML Fragment Collection Mapping</b>	
67.1	Introduction to XML Fragment Collection Mapping .....	67-1
<b>68</b>	<b>Configuring an XML Choice Object Mapping</b>	
68.1	Introduction to XML Choice Object Mapping.....	68-1
<b>69</b>	<b>Configuring an XML Choice Collection Mapping</b>	
69.1	Introduction to XML Choice Collection Mapping Configuration .....	69-1
<b>70</b>	<b>Configuring an XML Any Attribute Mapping</b>	
70.1	Introduction to XML Any Attribute Mapping Configuration .....	70-1
<b>Part XVII EIS Projects</b>		
<b>71</b>	<b>Introduction to EIS Projects</b>	
71.1	EIS Project Concepts .....	71-1
<b>72</b>	<b>Creating an EIS Project</b>	
72.1	Introduction to EIS Project Creation .....	72-1
72.2	Creating an EIS Project with XML Records .....	72-1

72.2.1	How to Create an EIS Project with XML Records Using Oracle JDeveloper .....	72-2
72.2.2	How to Create an EIS Project with XML Records Using TopLink Workbench .....	72-2
72.3	Creating an EIS Project with Indexed or Mapped Records .....	72-2
72.3.1	How to Create an EIS Project with Indexed or Mapped Records Using Java .....	72-2

## 73 Configuring an EIS Project

73.1	Introduction to EIS Project Configuration.....	73-1
73.2	Configuring EIS Data Source Platform at the Project Level .....	73-2
73.2.1	How to Configure EIS Data Source Platform at the Project Level Using TopLink Workbench	73-2
73.3	Configuring EIS Connection Specification Options at the Project Level.....	73-3
73.3.1	How to Configure EIS Connection Specification Options at the Project Level Using TopLink Workbench	73-3

## Part XVIII EIS Descriptors

### 74 Introduction to EIS Descriptors

74.1	EIS Descriptor Concepts .....	74-1
74.2	EIS Descriptors and Aggregation .....	74-1
74.2.1	Root and Composite Descriptors in EIS Projects .....	74-2
74.3	EIS Descriptors and Inheritance .....	74-2
74.3.1	Inheritance and Primary Keys in EIS Projects .....	74-2

### 75 Creating an EIS Descriptor

75.1	Introduction to EIS Descriptor Creation.....	75-1
75.2	Creating an EIS Descriptor .....	75-1
75.2.1	How to Create an EIS Descriptor Using TopLink Workbench .....	75-1
75.2.1.1	EIS Root Descriptors .....	75-1
75.2.1.2	EIS Composite Descriptors.....	75-2
75.2.2	How to Create an EIS Descriptor Using Java .....	75-2

### 76 Configuring an EIS Descriptor

76.1	Introduction to EIS Descriptor Configuration.....	76-1
76.2	Configuring Schema Context for an EIS Descriptor .....	76-3
76.2.1	How to Configure Schema Context for an EIS Descriptor Using TopLink Workbench ...	76-3
76.2.1.1	Choosing a Schema Context.....	76-3
76.2.2	How to Configure Schema Context for an EIS Descriptor Using Java .....	76-4
76.3	Configuring Default Root Element.....	76-4
76.3.1	How to Configure Default Root Element Using TopLink Workbench.....	76-4
76.3.1.1	Choosing a Root Element .....	76-5
76.3.2	How to Configure Default Root Element Using Java.....	76-5
76.4	Configuring Record Format .....	76-6
76.4.1	How to Configure Record Format Using Java .....	76-6
76.5	Configuring Custom EIS Interactions for Basic Persistence Operations .....	76-6

76.5.1	How to Configure Custom EIS Interactions for Basic Persistence Operations Using TopLink Workbench	76-7
76.5.2	How to Configure Custom EIS Interactions for Basic Persistence Operations Using Java	76-9
76.6	Configuring an EIS Descriptor as a Root or Composite Type.....	76-10
76.6.1	How to Configure an EIS Descriptor as a Root or Composite Type Using TopLink Workbench	76-10
76.6.2	How to Configure an EIS Descriptor as a Root or Composite Type Using Java...	76-10

## Part XIX EIS Mappings

### 77 Introduction to EIS Mappings

77.1	EIS Mapping Types.....	77-1
77.2	EIS Mapping Concepts.....	77-2
77.2.1	EIS Record Type.....	77-2
77.2.1.1	Indexed Records .....	77-3
77.2.1.2	Mapped Records.....	77-3
77.2.1.3	XML Records.....	77-3
77.2.2	XPath Support .....	77-3
77.2.3	xsd:list and xsd:union Support .....	77-3
77.2.4	jaxb:class Support .....	77-3
77.2.5	Typesafe Enumeration Support.....	77-4
77.2.6	Composite and Reference EIS Mappings .....	77-4
77.2.6.1	Composite EIS Mappings.....	77-4
77.2.6.2	Reference EIS Mappings.....	77-4
77.2.7	EIS Mapping Architecture.....	77-5
77.3	EIS Direct Mapping .....	77-6
77.4	EIS Composite Direct Collection Mapping .....	77-7
77.5	EIS Composite Object Mapping.....	77-7
77.6	EIS Composite Collection Mapping .....	77-8
77.7	EIS One-to-One Mapping .....	77-9
77.7.1	EIS One-to-One Mappings with Key on Source.....	77-10
77.7.2	EIS One-to-One Mappings with Key on Target .....	77-11
77.8	EIS One-to-Many Mapping .....	77-13
77.8.1	EIS One-to-Many Mappings with Key on Source.....	77-14
77.8.2	EIS One-to-Many Mappings with Key on Target .....	77-16
77.9	EIS Transformation Mapping.....	77-18

### 78 Configuring an EIS Mapping

78.1	Introduction to EIS Mapping Configuration .....	78-1
78.2	Configuring Common EIS Mapping Options.....	78-2
78.3	Configuring Reference Descriptors .....	78-2
78.3.1	How to Configure Reference Descriptors Using TopLink Workbench.....	78-3
78.4	Configuring Selection Interaction.....	78-4
78.4.1	How to Configure Selection Interaction Using TopLink Workbench.....	78-5

<b>79</b>	<b>Configuring an EIS Direct Mapping</b>	
79.1	Introduction to EIS Direct Mapping Configuration.....	79-1
<b>80</b>	<b>Configuring an EIS Composite Direct Collection Mapping</b>	
80.1	Introduction to EIS Composite Direct Collection Mapping Configuration .....	80-1
<b>81</b>	<b>Configuring an EIS Composite Object Mapping</b>	
81.1	Introduction to EIS Composite Object Mapping Configuration.....	81-1
<b>82</b>	<b>Configuring an EIS Composite Collection Mapping</b>	
82.1	Introduction to EIS Composite Collection Mapping Configuration .....	82-1
<b>83</b>	<b>Configuring an EIS One-to-One Mapping</b>	
83.1	Introduction to EIS One-to-One Mapping Configuration .....	83-1
83.2	Configuring Foreign Key Pairs .....	83-2
83.2.1	How to Configure Foreign Key Pairs Using TopLink Workbench .....	83-2
<b>84</b>	<b>Configuring an EIS One-to-Many Mapping</b>	
84.1	Introduction to EIS One-to-Many Mapping Configuration.....	84-1
84.2	Configuring Foreign Key Pairs .....	84-2
84.2.1	How to Configure Foreign Key Pairs Using TopLink Workbench .....	84-2
84.3	Configuring Delete All Interactions .....	84-4
84.3.1	How to Configure Delete All Interactions Using TopLink Workbench.....	84-4
<b>85</b>	<b>Configuring an EIS Transformation Mapping</b>	
85.1	Introduction EIS Transformation Mapping Configuration .....	85-1
<b>Part XX Using TopLink</b>		
<b>86</b>	<b>Introduction to Persistence Layer</b>	
86.1	Persistence Layer Concepts .....	86-1
86.1.1	Sessions .....	86-1
86.1.2	Data Access.....	86-2
86.1.3	Cache .....	86-2
86.1.4	Queries and Expressions .....	86-2
86.1.5	Transactions.....	86-3
<b>Part XXI TopLink Sessions</b>		
<b>87</b>	<b>Introduction to TopLink Sessions</b>	
87.1	Session Types.....	87-1
87.2	Session Concepts.....	87-2
87.2.1	Session Architecture .....	87-3



87.2.1.1	Object Cache .....	87-3
87.2.1.2	Connection Pools .....	87-4
87.2.1.3	Query Mechanism .....	87-4
87.2.1.4	Java Object Builder .....	87-4
87.2.2	Session Configuration and the sessions.xml File .....	87-5
87.2.3	Session Customization .....	87-5
87.2.4	Acquiring a Session at Run Time with the Session Manager .....	87-5
87.2.5	Managing Session Events with the Session Event Manager .....	87-6
87.2.5.1	Session Event Manager Events .....	87-6
87.2.5.2	Session Event Listeners .....	87-8
87.2.6	Logging .....	87-8
87.2.6.1	Log Types .....	87-8
87.2.6.1.1	TopLink Native Logging .....	87-8
87.2.6.1.2	java.util Logging .....	87-9
87.2.6.1.3	Server Logging .....	87-10
87.2.6.2	Log Output .....	87-10
87.2.6.3	Log Level .....	87-10
87.2.6.4	Logging SQL .....	87-10
87.2.6.5	Logging Chained Exceptions .....	87-11
87.2.6.6	Logging Inside a Java EE Container .....	87-11
87.2.6.7	Logging Outside of a Java EE Container .....	87-11
87.2.7	Profiler .....	87-11
87.2.7.1	TopLink Profiler .....	87-12
87.2.7.2	Oracle Dynamic Monitoring System (DMS) .....	87-12
87.2.8	Integrity Checker .....	87-12
87.2.9	Exception Handlers .....	87-12
87.2.10	Registering Descriptors .....	87-13
87.2.11	Sessions and CMP .....	87-13
87.2.12	Sessions and Sequencing .....	87-13
87.3	Server and Client Sessions .....	87-14
87.3.1	Three-Tier Architecture Overview .....	87-15
87.3.2	Advantages of the TopLink Three-Tier Architecture .....	87-15
87.3.2.1	Shared Resources .....	87-16
87.3.2.2	Providing Read Access .....	87-16
87.3.2.3	Providing Write Access .....	87-17
87.3.2.4	Security and User Privileges .....	87-18
87.3.2.5	Concurrency .....	87-19
87.3.2.6	Connection Allocation .....	87-19
87.4	Unit of Work Sessions .....	87-20
87.5	Isolated Client Sessions .....	87-20
87.5.1	Isolated Client Sessions and Oracle Virtual Private Database (VPD) .....	87-22
87.5.1.1	VPD with Oracle Database Proxy Authentication .....	87-22
87.5.1.2	VPD Without Oracle Database Proxy Authentication .....	87-22
87.5.1.3	Isolated Client Session Life Cycle .....	87-23
87.5.2	Isolated Client Session Limitations .....	87-24
87.6	Historical Sessions .....	87-25
87.6.1	Historical Session Limitations .....	87-26

87.7	Session Broker and Client Sessions .....	87-26
87.7.1	Session Broker Architecture .....	87-27
87.7.2	Committing a Transaction with a Session Broker.....	87-28
87.7.2.1	Committing a Session with a JTA Driver: Two-Phase Commits.....	87-28
87.7.2.2	Committing a Session Without a JTA Driver: Two-Stage Commits .....	87-28
87.7.3	Session Broker Session Limitations.....	87-28
87.7.3.1	Many-to-Many Join Tables and Direct Collection Tables.....	87-28
87.7.4	Session Broker Alternatives.....	87-29
87.7.4.1	Database Linking .....	87-29
87.7.4.2	Multiple Sessions .....	87-29
87.8	Database Sessions .....	87-29
87.9	Remote Sessions .....	87-30
87.9.1	Architectural Overview .....	87-31
87.9.1.1	Application Layer.....	87-32
87.9.1.2	Transport Layer .....	87-32
87.9.1.3	Server Layer.....	87-32
87.9.2	Remote Session Concepts .....	87-32
87.9.2.1	Securing Remote Session Access.....	87-33
87.9.2.2	Queries .....	87-33
87.9.2.3	Refreshing.....	87-33
87.9.2.4	Indirection.....	87-33
87.9.2.5	Cursored Streams .....	87-34
87.9.2.6	Unit of Work.....	87-34
87.10	Sessions and the Cache.....	87-34
87.10.1	Server and Database Session Cache .....	87-34
87.10.2	Isolated Session Cache .....	87-34
87.10.3	Historical Session Cache .....	87-34
87.11	Session API.....	87-34

## 88 Creating a Session

88.1	Introduction to the Session Creation.....	88-1
88.2	Creating a Sessions Configuration .....	88-1
88.2.1	How to Create a Sessions Configuration Using TopLink Workbench .....	88-2
88.3	Configuring a Sessions Configuration.....	88-3
88.3.1	How to Configure a Sessions Configuration Using TopLink Workbench.....	88-3
88.4	Creating a Server Session.....	88-4
88.4.1	How to Create a Server Session Using TopLink Workbench.....	88-4
88.4.2	How to Create a Server Session Using Java.....	88-6
88.5	Creating Session Broker and Client Sessions.....	88-6
88.5.1	How to Create a Session Broker and Client Sessions Using TopLink Workbench.	88-7
88.5.2	How to Create a Session Broker and Client Sessions Using Java .....	88-8
88.6	Creating Database Sessions .....	88-8
88.6.1	How to Create Database Sessions Using TopLink Workbench .....	88-9
88.6.2	How to Create Database Sessions Using Java .....	88-10
88.7	Creating Remote Sessions.....	88-11
88.7.1	How to Create Remote Sessions Using Java .....	88-11
88.7.1.1	Server.....	88-11

88.7.1.2	Client .....	88-11
----------	--------------	-------

## 89 Configuring a Session

89.1	Configuring Common Session Options.....	89-2
89.2	Configuring a Primary Mapping Project.....	89-3
89.2.1	How to Configure a Primary Mapping Project Using TopLink Workbench.....	89-3
89.2.2	How to Configure a Primary Mapping Project Using Java .....	89-4
89.3	Configuring a Session Login .....	89-5
89.4	Configuring Logging.....	89-5
89.4.1	How to Configure Logging Using TopLink Workbench.....	89-6
89.4.2	How to Configure Logging Using Session API in Java.....	89-8
89.4.3	How to Configure Logging Using Oracle Enterprise Manager .....	89-9
89.4.4	How to Configure Logging in a Java EE Container.....	89-9
89.4.5	How to Configure a Session to use the java.util.logging Package .....	89-9
89.4.5.1	logging.properties .....	89-9
89.4.5.2	Formatters.....	89-10
89.4.5.3	Namespace .....	89-10
89.5	Configuring Multiple Mapping Projects .....	89-11
89.5.1	How to Configure Multiple Mapping Projects Using TopLink Workbench .....	89-11
89.5.2	How to Configure Multiple Mapping Projects Using Java .....	89-12
89.6	Configuring a Performance Profiler.....	89-12
89.6.1	How to Configure a Performance Profiler Using TopLink Workbench.....	89-13
89.6.2	How to Configure a Performance Profiler Using Java .....	89-14
89.7	Configuring an Exception Handler .....	89-14
89.7.1	How to Configure an Exception Handler Using TopLink Workbench.....	89-14
89.7.2	How to Configure an Exception Handler Using Java .....	89-15
89.8	Configuring a Session Customizer Class.....	89-15
89.8.1	How to Configure Customizer Class Using TopLink Workbench.....	89-16
89.8.2	How to Configure Customizer Class Using Java .....	89-16
89.9	Configuring the Server Platform .....	89-17
89.9.1	How to Configure the Server Platform Using TopLink Workbench .....	89-18
89.9.2	How to Configure the Server Platform Using Java .....	89-19
89.10	Configuring Session Event Listeners .....	89-19
89.10.1	How to Configure Session Event Listeners Using TopLink Workbench .....	89-20
89.10.2	How to Configure Session Event Listeners Using Java .....	89-20
89.11	Configuring the Integrity Checker .....	89-21
89.11.1	How to Configure the Integrity Checker Using Java .....	89-22
89.12	Configuring Connection Policy .....	89-22
89.12.1	How to Configure Connection Policy Using TopLink Workbench .....	89-23
89.12.2	How to Configure Connection Policy Using Java.....	89-24
89.13	Configuring Named Queries at the Session Level .....	89-24
89.13.1	How to Configure Named Queries at the Session Level Using Java .....	89-25

## 90 Acquiring and Using Sessions at Run Time

90.1	Introduction to Session Acquisition.....	90-1
90.1.1	Session Manager .....	90-2

90.1.2	Multiple Sessions .....	90-2
90.2	Acquiring the Session Manager .....	90-2
90.3	Acquiring a Session from the Session Manager .....	90-2
90.3.1	How to Load a Session from sessions.xml Using Defaults.....	90-3
90.3.2	How to Load a Session from sessions.xml with an Alternative Class Loader.....	90-4
90.3.3	How to Load a Session from an Alternative Session Configuration File .....	90-4
90.3.4	How to Load a Session Without Logging In.....	90-5
90.3.5	How to Reload and Refresh Session Configuration .....	90-5
90.3.6	How to Refresh a Session when the Class Loader Changes.....	90-5
90.4	Acquiring a Client Session.....	90-6
90.4.1	How to Acquire an Isolated Client Session.....	90-6
90.4.2	How to Acquire a Client Session that Uses Exclusive Connections.....	90-7
90.4.3	How to Acquire a Client Session that Uses Connection Properties .....	90-8
90.4.4	How to Acquire a Client Session that Uses a Named Connection Pool .....	90-8
90.4.5	How to Acquire a Client Session that Does Not Use Lazy Connection Allocation	90-8
90.5	Acquiring a Historical Session .....	90-9
90.6	Logging In to a Session .....	90-9
90.7	Using Session API.....	90-9
90.8	Logging Out of a Session .....	90-10
90.9	Storing Sessions in the Session Manager Instance .....	90-10
90.10	Destroying Sessions in the Session Manager Instance .....	90-10

## 91 Configuring Server Sessions

91.1	Introduction to Server Session Configuration .....	91-1
91.2	Configuring Internal Connection Pools.....	91-2
91.3	Configuring External Connection Pools.....	91-2

## 92 Configuring Exclusive Isolated Client Sessions for Virtual Private Database

92.1	Introduction to Exclusive Isolated Client Session Configuration.....	92-1
92.2	Using PostAcquireExclusiveConnection Event Handler .....	92-2
92.2.1	How to Use Java.....	92-2
92.3	Using PreReleaseExclusiveConnection Event Handler.....	92-2
92.3.1	How to Use Java.....	92-3
92.4	Using NoRowsModifiedSessionEvent Event Handler .....	92-3
92.4.1	How to Use Java.....	92-4
92.5	Accessing Indirection .....	92-4

## 93 Configuring Historical Sessions

93.1	Introduction to Historical Session Configuration .....	93-1
93.1.1	How to Configure Historical Sessions Using an Oracle Platform.....	93-1
93.1.2	How to Configure Historical Sessions Using a TopLink HistoryPolicy.....	93-2

## 94 Configuring Session Broker and Client Sessions

94.1	Introduction to Session Broker and Client Session Configuration .....	94-1
94.2	Removing, Renaming, or Adding Sessions .....	94-2
94.2.1	How to Use TopLink Workbench to Remove, Rename, or Add Sessions.....	94-2

## 95 Configuring Database Sessions

95.1	Introduction to Database Session Configuration .....	95-1
95.2	Configuring External Connection Pools .....	95-2

## Part XXII Data Access

### 96 Introduction to Data Access

96.1	Data Access Concepts.....	96-1
96.1.1	Externally Managed Transactional Data Sources .....	96-1
96.1.2	Data Source Login Types.....	96-2
96.1.2.1	DatabaseLogin .....	96-2
96.1.2.2	EISLogin.....	96-3
96.1.3	Data Source Platform Types.....	96-3
96.1.3.1	Database Platforms.....	96-3
96.1.3.2	EIS Platforms .....	96-4
96.1.4	Authentication.....	96-5
96.1.4.1	Simple JDBC Authentication .....	96-5
96.1.4.2	Oracle Database Proxy Authentication .....	96-5
96.1.4.3	Auditing.....	96-6
96.1.5	Connections .....	96-7
96.1.6	Connection Pools .....	96-7
96.1.6.1	Internal Connection Pools .....	96-7
96.1.6.2	External Connection Pools .....	96-8
96.1.6.3	Default (Write) and Read Connection Pools .....	96-8
96.1.6.4	Sequence Connection Pools .....	96-8
96.1.6.5	Application-Specific Connection Pools .....	96-9
96.2	Data Access API .....	96-10
96.2.1	Login Inheritance Hierarchy .....	96-10
96.2.2	Platform Inheritance Hierarchy.....	96-10

### 97 Configuring a Data Source Login

97.1	Configuring Common Data Source Login Options .....	97-1
97.2	Configuring User Name and Password .....	97-2
97.2.1	How to Configure User Name and Password Using TopLink Workbench .....	97-2
97.3	Configuring Password Encryption.....	97-3
97.3.1	How to Configure Password Encryption Using Java.....	97-3
97.4	Configuring External Connection Pooling.....	97-4
97.4.1	How to Configure External Connection Pooling Using TopLink Workbench.....	97-5
97.4.2	How to Configure External Connection Pooling Using Java .....	97-6
97.5	Configuring Properties.....	97-6
97.5.1	How to Configure Properties Using TopLink Workbench.....	97-7
97.5.2	How to Configure Properties Using Java.....	97-8
97.6	Configuring a Default Null Value at the Login Level .....	97-8
97.6.1	How to Configure a Default Null Value at the Login Level Using Java .....	97-8

## 98 Configuring a Database Login

98.1	Introduction to Database Login Configuration.....	98-1
98.2	Configuring a Relational Database Platform at the Session Level .....	98-2
98.2.1	How to Configure a Relational Database Platform at the Session Level Using TopLink Workbench	98-2
98.3	Configuring Database Login Connection Options.....	98-3
98.3.1	How to Configure Database Login Connection Options Using TopLink Workbench.....	98-3
98.4	Configuring Sequencing at the Session Level.....	98-5
98.4.1	How to Configure Sequencing at the Session Level Using TopLink Workbench...	98-5
98.4.2	How to Configure Sequencing at the Session Level Using Java.....	98-6
98.4.2.1	Using the Platform Default Sequence.....	98-6
98.4.2.2	Configuring Multiple Sequences.....	98-7
98.4.2.3	Configuring Query Sequencing.....	98-8
98.5	Configuring a Table Qualifier .....	98-9
98.5.1	How to Configure a Table Qualifier Using TopLink Workbench .....	98-9
98.5.2	How to Configure a Table Qualifier Using Java .....	98-9
98.6	Configuring JDBC Options.....	98-10
98.6.1	How to Configure JDBC Options Using TopLink Workbench.....	98-10
98.6.2	How to Configure JDBC Options Using Java .....	98-11
98.7	Configuring Advanced Options .....	98-12
98.7.1	How to Configure Advanced Options Using TopLink Workbench .....	98-12
98.7.2	How to Configure Advanced Options Using Java .....	98-13
98.8	Configuring Oracle Database Proxy Authentication.....	98-13
98.8.1	How to Configure Oracle Database Proxy Authentication Using Java.....	98-14

## 99 Configuring an EIS Login

99.1	Introduction to EIS Login Configuration .....	99-1
99.2	Configuring an EIS Data Source Platform at the Session Level .....	99-1
99.2.1	How to Configure an EIS Data Source Platform at the Session Level Using TopLink Workbench	99-2
99.3	Configuring EIS Connection Specification Options at the Session Level.....	99-2
99.3.1	How to Configure EIS Connection Specification Options at the Session Level Using TopLink Workbench	99-2

## 100 Creating an Internal Connection Pool

100.1	Introduction to the Internal Connection Pool Creation.....	100-1
100.2	Creating an Internal Connection Pool.....	100-1
100.2.1	How to Create an Internal Connection Pool Using TopLink Workbench.....	100-1
100.2.2	How to Create an Internal Connection Pool Using Java.....	100-2

## 101 Configuring an Internal Connection Pool

101.1	Introduction to the Internal Connection Pool Configuration.....	101-1
101.2	Configuring Connection Pool Sizes.....	101-2
101.2.1	How to Configure Connection Pool Size Using TopLink Workbench .....	101-2
101.2.2	How to Configure Connection Pool Size Using Java .....	101-3

101.3	Configuring Properties.....	101-3
101.3.1	How to Configure Properties Using TopLink Workbench.....	101-3
101.3.2	How to Configure Properties Using Java.....	101-4
101.4	Configuring a Nontransactional Read Login.....	101-4
101.4.1	How to Configure Nontransactional Read Login Using TopLink Workbench.....	101-4
101.4.2	How to Configure Nontransactional Read Login Using Java.....	101-5
101.5	Configuring Connection Pool Connection Options.....	101-5
101.5.1	How to Configure Connection Pool Connection Options Using TopLink Workbench.... 101-6	
101.6	Configuring Exclusive Read Connections.....	101-7
101.6.1	How to Configure Exclusive Read Connections Using TopLink Workbench.....	101-8

## Part XXIII Cache

### 102 Introduction to Cache

102.1	Cache Architecture .....	102-1
102.1.1	Session Cache .....	102-2
102.1.2	Unit of Work Cache.....	102-2
102.2	Cache Concepts .....	102-2
102.2.1	Cache Type and Object Identity .....	102-3
102.2.1.1	Full Identity Map .....	102-3
102.2.1.2	Weak Identity Map.....	102-4
102.2.1.3	Soft Identity Map.....	102-4
102.2.1.4	Soft Cache Weak Identity Map and Hard Cache Weak Identity Map.....	102-4
102.2.1.5	No Identity Map .....	102-4
102.2.1.6	Guidelines for Configuring the Cache and Identity Maps.....	102-4
102.2.1.7	What You May Need to Know About the Internals of Weak, Soft, and Hard Identity Maps 102-5	
102.2.2	Querying and the Cache .....	102-6
102.2.3	Handling Stale Data .....	102-6
102.2.3.1	Configuring a Locking Policy .....	102-7
102.2.3.2	Configuring the Cache on a Per-Class Basis.....	102-7
102.2.3.3	Forcing a Cache Refresh when Required on a Per-Query Basis.....	102-7
102.2.3.4	Configuring Cache Invalidation.....	102-7
102.2.3.5	Configuring Cache Coordination.....	102-7
102.2.4	Explicit Query Refreshes .....	102-7
102.2.4.1	Refresh Policy.....	102-8
102.2.4.2	EJB 2.n CMP Finders and Refresh Policy .....	102-8
102.2.5	Cache Invalidation.....	102-8
102.2.6	Cache Coordination.....	102-9
102.2.7	Cache Isolation .....	102-9
102.2.8	Cache Locking and Transaction Isolation .....	102-10
102.2.9	Cache Optimization.....	102-10
102.3	Cache Coordination.....	102-10
102.3.1	When to Use Cache Coordination.....	102-11
102.3.2	Coordinated Cache Architecture.....	102-11
102.3.2.1	Session.....	102-12

102.3.2.2	Descriptor .....	102-12
102.3.2.3	Unit of Work.....	102-12
102.3.3	Coordinated Cache Types .....	102-12
102.3.3.1	JMS Coordinated Cache.....	102-12
102.3.3.2	RMI Coordinated Cache .....	102-13
102.3.3.3	CORBA Coordinated Cache.....	102-13
102.3.4	Custom Coordinated Cache .....	102-13
102.4	Cache API.....	102-14
102.4.1	Object Identity API .....	102-14
102.4.2	Cache Refresh API.....	102-14
102.4.3	Cache Invalidation API.....	102-14
102.4.4	Cache Coordination API.....	102-15

## 103 Configuring a Coordinated Cache

103.1	Configuring Common Coordinated Cache Options.....	103-1
103.2	Configuring the Synchronous Change Propagation Mode .....	103-2
103.2.1	How to Configure the Synchronous Change Propagation Mode Using TopLink Workbench 103-3	
103.2.2	How to Configure the Synchronous Change Propagation Mode Using Java .....	103-3
103.3	Configuring a Service Channel .....	103-4
103.3.1	How to Configure a Service Channel Using TopLink Workbench .....	103-4
103.3.2	How to Configure a Service Channel Using Java .....	103-5
103.4	Configuring a Multicast Group Address.....	103-5
103.4.1	How to Configure a Multicast Group Address Using TopLink Workbench.....	103-6
103.4.2	How to Configure a Multicast Group Address Using Java.....	103-7
103.5	Configuring a Multicast Port.....	103-7
103.5.1	How to Configure a Multicast Port Using TopLink Workbench.....	103-8
103.5.2	How to Configure a Multicast Port Using Java.....	103-9
103.6	Configuring a Naming Service Type .....	103-9
103.7	Configuring JNDI Naming Service Information .....	103-9
103.7.1	How to Configure JNDI Naming Service Information Using TopLink Workbench .....	103-10
103.7.2	How to Configure JNDI Naming Service Information Using Java .....	103-12
103.8	Configuring RMI Registry Naming Service Information.....	103-13
103.8.1	How to Configure RMI Registry Naming Service Information Using TopLink Workbench 103-14	
103.8.2	How to Configure RMI Registry Naming Service Information Using Java.....	103-15
103.9	Configuring an Announcement Delay.....	103-15
103.9.1	How to Configure an Announcement Delay Using TopLink Workbench .....	103-15
103.9.2	How to Configure an Announcement Delay Using Java.....	103-16
103.10	Configuring Connection Handling .....	103-16
103.10.1	How to Configure Connection Handling Using TopLink Workbench .....	103-17
103.10.2	How to Configure Connection Handling Using Java.....	103-18
103.11	Configuring Context Properties.....	103-18
103.11.1	How to Configure Context Properties Using TopLink Workbench.....	103-18
103.11.2	How to Configure Context Properties Using Java.....	103-19
103.12	Configuring a Packet Time-to-Live .....	103-19



103.12.1	How to Configure a Packet Time-to-Live Using TopLink Workbench.....	103-20
103.12.2	How to Configure a Packet Time-to-Live Using Java .....	103-21

## 104 Configuring a JMS Coordinated Cache

104.1	Introduction to JMS Coordinated Cache Configuration .....	104-1
104.2	Configuring a Topic Name .....	104-2
104.2.1	How to Configure a Topic Name Using TopLink Workbench.....	104-2
104.2.2	How to Configure a Topic Name Java.....	104-2
104.3	Configuring a Topic Connection Factory Name .....	104-3
104.3.1	How to Configure a Topic Connection Factory Name Using TopLink Workbench .....	104-3
104.3.2	How to Configure a Topic Connection Factory Name Using Java .....	104-4
104.4	Configuring a Topic Host URL .....	104-4
104.4.1	How to Configure a Topic Host URL Using TopLink Workbench.....	104-4
104.4.2	How to Configure a Topic Host URL Using Java .....	104-4
104.5	Configuring Connection Handling .....	104-5
104.5.1	How to Configure Connection Handling Using TopLink Workbench .....	104-5
104.5.2	How to Configure Connection Handling Using Java.....	104-6

## 105 Configuring an RMI Coordinated Cache

105.1	Introduction to RMI Coordinated Cache Configuration.....	105-1
-------	--	-------

## 106 Configuring a CORBA Coordinated Cache

106.1	Introduction to CORBA Coordinated Cache Configuration .....	106-1
-------	---	-------

## 107 Configuring a Custom Coordinated Cache

107.1	Introduction to Custom Coordinated Cache Configuration .....	107-1
107.2	Configuring Transport Class.....	107-1
107.2.1	How to Configure Transport Class Using TopLink Workbench.....	107-1
107.2.2	How to Configure Transport Class Using Java.....	107-2

## Part XXIV Queries

### 108 Introduction to TopLink Queries

108.1	Query Types .....	108-1
108.2	Query Concepts.....	108-2
108.2.1	Call .....	108-3
108.2.2	DatabaseQuery .....	108-3
108.2.3	Data-Level and Object-Level Queries .....	108-3
108.2.4	Summary Queries .....	108-3
108.2.5	Descriptor Query Manager .....	108-3
108.2.6	TopLink Expressions.....	108-4
108.2.7	Query Keys .....	108-4
108.2.8	Query Languages.....	108-4
108.2.8.1	SQL Queries.....	108-5

108.2.8.2	EJB QL Queries .....	108-5
108.2.8.3	JP QL Queries .....	108-6
108.2.8.4	XML Queries .....	108-6
108.2.8.5	EIS Interactions .....	108-6
108.2.8.6	Query-by-Example .....	108-6
108.3	Building Queries .....	108-6
108.4	Executing Queries .....	108-7
108.5	Handling Query Results .....	108-8
108.5.1	Collection Query Results .....	108-8
108.5.2	Report Query Results .....	108-8
108.5.3	Stream and Cursor Query Results.....	108-9
108.6	Session Queries.....	108-9
108.6.1	Read-Object Session Queries.....	108-10
108.6.2	Create, Update, and Delete Object Session Queries .....	108-10
108.7	Database Queries .....	108-10
108.7.1	Object-Level Read Query.....	108-11
108.7.1.1	ReadObjectQuery .....	108-11
108.7.1.2	ReadAllQuery .....	108-11
108.7.1.3	Partial Object Queries .....	108-11
108.7.1.4	Read-Only Query.....	108-12
108.7.1.5	Join Reading and Object-Level Read Queries .....	108-12
108.7.1.5.1	Avoiding Join-Reading Duplicate Data .....	108-13
108.7.1.6	Fetch Groups and Object-Level Read Queries .....	108-14
108.7.2	Data-Level Read Query.....	108-14
108.7.2.1	DataReadQuery .....	108-14
108.7.2.2	DirectReadQuery .....	108-14
108.7.2.3	ValueReadQuery .....	108-14
108.7.3	Object-Level Modify Query.....	108-14
108.7.3.1	WriteObjectQuery.....	108-15
108.7.3.2	UpdateObjectQuery .....	108-15
108.7.3.3	InsertObjectQuery .....	108-15
108.7.3.4	DeleteObjectQuery .....	108-15
108.7.3.5	UpdateAllQuery .....	108-15
108.7.3.6	DeleteAllQuery .....	108-15
108.7.3.7	Object-Level Modify Queries and Privately Owned Parts.....	108-15
108.7.4	Data-Level Modify Query .....	108-16
108.7.5	Report Query .....	108-16
108.8	Named Queries .....	108-17
108.9	Call Queries .....	108-17
108.9.1	SQL Calls.....	108-18
108.9.1.1	SQLCall .....	108-18
108.9.1.2	StoredProcedureCall .....	108-18
108.9.1.3	StoredFunctionCall.....	108-18
108.9.2	EJB QL Calls.....	108-19
108.9.3	Enterprise Information System (EIS) Interactions .....	108-19
108.9.3.1	IndexedInteraction .....	108-19
108.9.3.2	MappedInteraction.....	108-19

108.9.3.3	XMLInteraction.....	108-19
108.9.3.4	XQueryInteraction.....	108-20
108.9.3.5	QueryStringInteraction.....	108-20
108.10	Redirect Queries.....	108-20
108.11	Historical Queries.....	108-21
108.11.1	Using an ObjectLevelReadQuery with an AsOfClause.....	108-21
108.11.2	Using an ObjectLevelReadQuery with Expression Operator asOf.....	108-22
108.11.3	Using an ObjectLevelReadQuery in a Historical Session.....	108-22
108.12	Interface and Inheritance Queries.....	108-22
108.13	Descriptor Query Manager Queries.....	108-22
108.13.1	How to Configure Named Queries.....	108-23
108.13.2	How to Configure Default Query Implementations.....	108-23
108.13.3	How to Configure Additional Join Expressions.....	108-23
108.14	Oracle Extensions.....	108-24
108.14.1	Hints.....	108-24
108.14.2	Hierarchical Queries.....	108-24
108.14.3	Flashback Queries.....	108-24
108.14.4	Stored Functions.....	108-24
108.15	EJB 2.n CMP Finders.....	108-25
108.15.1	Predefined Finders.....	108-25
108.15.1.1	Predefined CMP Finders.....	108-26
108.15.1.2	Predefined BMP Finders.....	108-26
108.15.2	Default Finders.....	108-27
108.15.3	Call Finders.....	108-27
108.15.4	DatabaseQuery Finders.....	108-27
108.15.5	Named Query Finders.....	108-27
108.15.6	Primary Key Finders.....	108-28
108.15.7	Expression Finders.....	108-28
108.15.8	EJB QL Finders.....	108-28
108.15.9	SQL Finders.....	108-28
108.15.10	Redirect Finders.....	108-29
108.15.11	The ejbSelect Method.....	108-29
108.16	Queries and the Cache.....	108-30
108.16.1	How to Configure the Cache.....	108-30
108.16.2	How to Use In-Memory Queries.....	108-31
108.16.2.1	Configuring Cache Usage for In-Memory Queries.....	108-31
108.16.2.2	Expression Options for In-Memory Queries.....	108-32
108.16.2.3	Handling Exceptions Resulting from In-Memory Queries.....	108-34
108.16.3	Primary Key Queries and the Cache.....	108-35
108.16.4	How to Disable the Identity Map Cache Update During a Read Query.....	108-35
108.16.5	How to Refresh the Cache.....	108-36
108.16.5.1	Object Refresh.....	108-36
108.16.5.2	Cascading Object Refresh.....	108-36
108.16.5.3	Refreshing the Identity Map Cache During a Read Query.....	108-36
108.16.6	How to Cache Query Results in the Session Cache.....	108-37
108.16.7	How to Cache Query Results in the Query Cache.....	108-37
108.16.7.1	Internal Query Cache Restrictions.....	108-38

108.16.8	How to Use Caching and EJB 2.n CMP Finders .....	108-38
108.16.8.1	Caching Options .....	108-38
108.16.8.2	Disabling Cache for Returned Finder Results .....	108-39
108.16.8.3	Refreshing Finder Results .....	108-39
108.17	Query API .....	108-39

## 109 Using Basic Query API

109.1	Using Session Queries .....	109-1
109.1.1	How to Read Objects with a Session Query .....	109-2
109.1.1.1	Reading an Object with a Session Query .....	109-2
109.1.1.2	Reading All Objects with a Session Query .....	109-2
109.1.1.3	Refreshing an Object with a Session Query .....	109-2
109.1.2	How to Create, Update, and Delete Objects with a Session Query .....	109-3
109.1.2.1	Writing a Single Object to the Database with a Session Query .....	109-3
109.1.2.2	Writing All Objects to the Database with a Session Query .....	109-3
109.1.2.3	Adding New Objects to the Database with a Session Query .....	109-4
109.1.2.4	Modifying Existing Objects in the Database with a Session Query .....	109-4
109.1.2.5	Deleting Objects in the Database with a Session Query .....	109-4
109.2	Using DatabaseQuery Queries.....	109-4
109.2.1	How to Read Objects Using a DatabaseQuery .....	109-4
109.2.1.1	Performing Basic DatabaseQuery Read Operations .....	109-5
109.2.1.2	Reading Objects Using Partial Object Queries .....	109-6
109.2.1.3	Reading Objects Using Report Queries.....	109-6
109.2.1.4	Reading Objects Using Query-By-Example.....	109-6
109.2.1.5	Specifying Read Ordering .....	109-8
109.2.1.6	Specifying a Collection Class .....	109-9
109.2.1.7	Specifying the Maximum Rows Returned.....	109-9
109.2.1.8	Configuring Query Timeout at the Query Level .....	109-10
109.2.1.9	Using Batch Reading.....	109-10
109.2.1.10	Using Join Reading with ObjectLevelReadQuery .....	109-11
109.2.1.10.1	Using Java.....	109-11
109.2.2	How to Create, Update, and Delete Objects with a DatabaseQuery.....	109-12
109.2.2.1	Using Write Query .....	109-12
109.2.2.2	Performing Noncascading Write Queries.....	109-13
109.2.2.3	Disabling the Identity Map Cache During a Write Query .....	109-13
109.2.3	How to Update and Delete Multiple Objects with a DatabaseQuery.....	109-14
109.2.3.1	Using UpdateAll Queries .....	109-14
109.2.3.2	Using DeleteAll Queries .....	109-14
109.2.4	How to Read Data with a DatabaseQuery.....	109-15
109.2.4.1	Using a DataReadQuery .....	109-15
109.2.4.2	Using a DirectReadQuery .....	109-15
109.2.4.3	Using a ValueReadQuery .....	109-15
109.2.5	How to Update Data with a DatabaseQuery.....	109-16
109.2.6	How to Specify a Custom SQL String in a DatabaseQuery .....	109-16
109.2.7	How to Specify a Custom JPQL String in a DatabaseQuery .....	109-16
109.2.8	How to Specify a Custom EJB QL String in a DatabaseQuery.....	109-16
109.2.9	How to Use Parameterized SQL and Statement Caching in a DatabaseQuery...	109-17

109.3	Using Named Queries.....	109-18
109.4	Using a SQLCall .....	109-18
109.4.1	How to Configure a SQLCall Without Arguments .....	109-19
109.4.2	How to Configure a SQLCall with Arguments Using JDBC Data Types.....	109-19
109.4.3	What You May Need to Know About Using a SQLCall.....	109-21
109.5	Using a StoredProcedureCall.....	109-21
109.5.1	How to Configure a StoredProcedureCall Without Arguments .....	109-22
109.5.2	How to Configure a StoredProcedureCall with Arguments Using JDBC Data Types .....	109-22
109.5.3	How to Configure a PLSQLStoredProcedureCall with PL/SQL Data Type Arguments.	109-24
109.5.4	How to Specify a Simple Optimistic Version Locking Value with a StoredProcedureCall Using JDBC Data Types	109-27
109.5.5	How to Configure a StoredProcedureCall Output Parameter Event Using JDBC or PL/SQL Data Types	109-29
109.5.6	What You May Need to Know About Using a StoredProcedureCall.....	109-30
109.6	Using a StoredFunctionCall.....	109-31
109.6.1	What You May Need to Know About Using a StoredFunctionCall .....	109-32
109.7	Using Java Persistence Query Language (JPQL) Calls .....	109-32
109.8	Using EIS Interactions .....	109-32
109.9	Handling Exceptions .....	109-33
109.10	Handling Collection Query Results .....	109-33
109.11	Handling Report Query Results .....	109-33

## 110 Introduction to TopLink Expressions

110.1	Expression Framework .....	110-1
110.1.1	Expressions Compared to SQL .....	110-1
110.2	Expression Components .....	110-2
110.2.1	Boolean Logic .....	110-3
110.2.2	Database Functions and Operators.....	110-3
110.2.3	Mathematical Functions.....	110-4
110.2.4	XMLType Functions .....	110-5
110.2.5	Platform and User-Defined Functions.....	110-5
110.2.6	Expressions for One-to-One and Aggregate Object Relationships .....	110-5
110.2.7	Expressions for Joining and Complex Relationships .....	110-6
110.2.7.1	What You May Need to Know About Joins .....	110-6
110.2.7.2	Using TopLink Expression API for Joins .....	110-7
110.3	Parameterized Expressions.....	110-8
110.3.1	Expression Method getParameter .....	110-8
110.3.2	Expression Method getField .....	110-9
110.4	Query Keys and Expressions.....	110-9
110.5	Multiple Expressions .....	110-10
110.5.1	How to Use Subselects and Subqueries.....	110-10
110.5.2	How to Use Parallel Expressions.....	110-11
110.6	Data Queries and Expressions .....	110-11
110.6.1	How to Use the getField Method .....	110-12
110.6.2	How to Use the getTable Method.....	110-12

110.7	Creating an Expression .....	110-12
110.7.1	How to Create an Expression Using TopLink Workbench .....	110-13
110.7.1.1	Adding Arguments .....	110-14
110.7.2	How to Create an Expression Using Java .....	110-15
110.8	Creating and Using a User-Defined Function .....	110-16
110.8.1	How to Make a User-Defined Function Available to a Specific Platform .....	110-16
110.8.2	How to Make a User-Defined Function Available to All Platforms .....	110-17
110.8.2.1	Using a User-Defined Function .....	110-17

## 111 Using Advanced Query API

111.1	Using Redirect Queries .....	111-1
111.1.1	How to Create a Redirect Query .....	111-1
111.2	Using Historical Queries .....	111-2
111.3	Using Queries with Fetch Groups .....	111-3
111.3.1	How to Configure Default Fetch Group Behavior .....	111-3
111.3.2	How to Query with a Static Fetch Group .....	111-3
111.3.3	How to Query with a Dynamic Fetch Group .....	111-4
111.4	Using Read-Only Queries .....	111-4
111.5	Querying on Interfaces .....	111-4
111.6	Querying on an Inheritance Hierarchy .....	111-5
111.7	Appending Additional Join Expressions .....	111-5
111.7.1	How to Append Additional Join Expressions Using Java .....	111-5
111.8	Using Queries on Variable One-to-One Mappings .....	111-5
111.9	Using Oracle Database Features .....	111-6
111.9.1	How to Use Oracle Hints .....	111-6
111.9.2	How to Use Hierarchical Queries .....	111-7
111.9.2.1	Using startWith Parameter .....	111-7
111.9.2.2	Using connectBy Parameter .....	111-7
111.9.2.3	Using orderSibling Parameter .....	111-7
111.10	Using EJB 2.n CMP Finders .....	111-8
111.10.1	How to Create a Finder .....	111-8
111.10.1.1	ejb-jar.xml Finder Options .....	111-9
111.10.2	How to Use DatabaseQuery Finders .....	111-10
111.10.3	How to Use Named Query Finders .....	111-11
111.10.4	How to Use Primary Key Finders .....	111-11
111.10.5	How to Use EJB QL Finders .....	111-11
111.10.6	How to Use SQL Finders .....	111-12
111.10.7	How to Use Redirect Finders .....	111-12
111.10.8	How to Use the ejbSelect Method .....	111-15
111.11	Handling Cursor and Stream Query Results .....	111-16
111.11.1	How to Handle Cursors and Java Iterators .....	111-16
111.11.1.1	Traversing Data with Scrollable Cursors .....	111-16
111.11.2	How to Handle Java Streams .....	111-17
111.11.2.1	Using Cursored Stream Support .....	111-17
111.11.3	How to Optimize Streams .....	111-18
111.11.4	How to Use Cursors and Streams with EJB 2.n CMP Finders .....	111-18
111.11.4.1	Building the Query .....	111-19

111.11.4.2	Executing the Finder from the Client .....	111-19
111.12	Handling Query Results Using Pagination.....	111-20
111.13	Using Queries and the Cache.....	111-20
111.13.1	How to Cache Results in a ReadQuery .....	111-20
111.13.2	How to Configure Cache Expiration at the Query Level.....	111-21

## 112 Introduction to TopLink Support for Oracle Spatial

112.1	TopLink Support for Oracle Spatial .....	112-1
112.2	Using Structure Converters .....	112-1
112.2.1	How to Configure the Database Platform to Use Structure Converters .....	112-2
112.2.2	How to Set Up Mappings Using Structure Converters.....	112-2
112.3	Using JGeometry .....	112-2
112.3.1	How to Configure the Database Platform to Use JGeometry .....	112-2
112.3.2	How to Map JGeometry Attributes.....	112-2
112.3.3	How to Perform Queries Using Spatial Operator Expressions.....	112-3

## Part XXV Transactions

### 113 Introduction to TopLink Transactions

113.1	Unit of Work Architecture.....	113-1
113.1.1	Unit of Work Transaction Context .....	113-2
113.1.2	Unit of Work Transaction Demarcation .....	113-2
113.1.2.1	JTA Controlled Transactions.....	113-3
113.1.2.2	OTS Controlled Transactions.....	113-3
113.1.2.3	CMP-Controlled Transactions .....	113-3
113.1.3	Unit of Work Transaction Isolation.....	113-4
113.2	Unit of Work Concepts.....	113-4
113.2.1	Unit of Work Benefits.....	113-4
113.2.2	Unit of Work Life Cycle .....	113-5
113.2.3	Unit of Work and Change Policy.....	113-7
113.2.3.1	Deferred Change Detection Policy.....	113-7
113.2.3.2	Object-Level Change Tracking Policy .....	113-7
113.2.3.2.1	EJB CMP and JPA.....	113-8
113.2.3.3	Attribute Change Tracking Policy .....	113-8
113.2.3.3.1	JPA Entities.....	113-8
113.2.3.3.2	Plain Old Java Object (POJO) Classes .....	113-8
113.2.3.3.3	EJB CMP on OC4J.....	113-9
113.2.3.4	Change Policy Mapping Support.....	113-9
113.2.4	Clones and the Unit of Work .....	113-9
113.2.5	Nested and Parallel Units of Work .....	113-10
113.2.5.1	Nested Unit of Work .....	113-10
113.2.5.2	Parallel Unit of Work .....	113-10
113.2.6	Commit and Rollback Transactions .....	113-10
113.2.6.1	Commit Transactions .....	113-10
113.2.6.1.1	Commit and JTA.....	113-11
113.2.6.2	Rollback Transactions .....	113-11

113.2.6.2.1	Rollback and JTA.....	113-11
113.2.7	Primary Keys.....	113-11
113.2.8	Unit of Work Optimization.....	113-11
113.3	Unit of Work API.....	113-12
113.3.1	Unit of Work as Session.....	113-12
113.3.1.1	Reading and Querying Objects with the Unit of Work.....	113-12
113.3.1.1.1	Reading Objects with the Unit of Work.....	113-12
113.3.1.1.2	Querying Objects with the Unit of Work.....	113-12
113.3.1.2	Locking and the Unit of Work.....	113-12
113.4	Example Model Object and Schema.....	113-13

## 114 Using Basic Unit of Work API

114.1	Acquiring a Unit of Work.....	114-1
114.2	Creating an Object.....	114-2
114.3	Modifying an Object.....	114-2
114.4	Associating a New Target to an Existing Source Object.....	114-3
114.4.1	How to Associate a New Target to an Existing Source Object in a Unidirectional Relationship: Reference to the New Cache Object After Commit not Required	114-3
114.4.2	How to Associate a New Target to an Existing Source Object in a Unidirectional Relationship: Reference to the New Cache Object After Commit Required	114-4
114.4.3	How to Associate a New Target to an Existing Source Object in a Bidirectional Relationship: Query for Target Before Commit not Required	114-5
114.4.4	How to Associate a New Target to an Existing Source Object in a Bidirectional Relationship: Query for Target Object Before Commit Required	114-7
114.5	Associating a New Source to an Existing Target Object.....	114-8
114.6	Associating an Existing Source to an Existing Target Object.....	114-9
114.7	Deleting Objects.....	114-10
114.7.1	How to Use the privateOwnedRelationship Attribute.....	114-11
114.7.2	How to Explicitly Delete from the Database.....	114-12
114.7.3	What You May Need to Know About the Order in which Objects Are Deleted	114-12

## 115 Using Advanced Unit of Work API

115.1	Registering and Unregistering Objects.....	115-1
115.1.1	How to Create and Register a New Object in One Step Using UnitOfWork Method newInstance	115-2
115.1.2	How to Use the registerAllObjects Method.....	115-2
115.1.3	How to Use Registration and Existence Checking.....	115-3
115.1.3.1	Using Check Database.....	115-3
115.1.3.2	Using Assume Existence.....	115-3
115.1.3.3	Using Assume Nonexistence.....	115-3
115.1.4	How to Work with Aggregates.....	115-3
115.1.5	How to Unregister Working Clones.....	115-4
115.1.6	What You May Need to Know About Object Registration.....	115-4
115.2	Declaring Read-Only Classes.....	115-5
115.2.1	How to Configure Read-Only Classes for a Single Unit of Work.....	115-6
115.2.2	How to Configure Default Read-Only Classes.....	115-6
115.2.3	How to Declare Read-Only Descriptors.....	115-6



115.3	Writing Changes Before Commit Time .....	115-6
115.4	Using Conforming Queries and Descriptors .....	115-7
115.4.1	How to Use Conforming.....	115-7
115.4.1.1	Ensuring that the Query Supports Conforming .....	115-8
115.4.1.2	Considering how Conforming Affects Database Results .....	115-8
115.4.1.3	Registering New Objects and Instantiate Relationships.....	115-9
115.4.2	How to Use Conforming Queries.....	115-10
115.4.3	How to Use Conforming Descriptors .....	115-11
115.4.4	What You May Need to Know About Conforming Query Alternatives.....	115-11
115.4.4.1	Using Unit of Work Method writeChanges Instead of Conforming .....	115-11
115.4.4.2	Using Unit of Work Properties Instead of Conforming.....	115-12
115.5	Merging Changes in Working Copy Clones .....	115-12
115.6	Resuming a Unit of Work After Commit .....	115-13
115.7	Reverting a Unit of Work.....	115-14
115.8	Using a Nested or Parallel Unit of Work.....	115-14
115.8.1	How to Use Parallel Unit of Work .....	115-14
115.8.2	How to Use Nested Unit of Work .....	115-15
115.9	Using a Unit of Work with Custom SQL.....	115-15
115.10	Controlling the Order of Delete Operations .....	115-16
115.10.1	How to Use the setShouldPerformDeletesFirst Method of the Unit of Work .....	115-16
115.10.2	How to Use the addConstraintDependencies Method of the Descriptor .....	115-16
115.10.3	How to Use the deleteAllObjects Method Without the addConstraintDependencies Method	115-16
115.10.4	How to Use the deleteAllObjects Method with the addConstraintDependencies Method	115-17
115.11	Using Optimistic Read Locking with the forceUpdateToVersionField Method .....	115-17
115.11.1	How to Force a Check of the Optimistic Read Lock.....	115-18
115.11.2	How to Force a Version Field Update .....	115-18
115.11.3	How to Disable the forceUpdateToVersionField Configuration.....	115-20
115.12	Implementing User and Date Auditing with the Unit of Work.....	115-20
115.13	Integrating the Unit of Work with an External Transaction Service .....	115-20
115.13.1	How to Acquire a Unit of Work with an External Transaction Service.....	115-21
115.13.2	How to Use a Unit of Work when an External Transaction Exists.....	115-21
115.13.3	How to Use a Unit of Work when No External Transaction Exists .....	115-22
115.13.4	How to Use the Unit of Work to Handle External Transaction Timeouts and Exceptions	115-23
115.13.4.1	Handling External Transaction Commit Timeouts .....	115-23
115.13.4.2	Handling External Transaction Commit Exceptions.....	115-23
115.14	Integrating the Unit of Work with CMP.....	115-24
115.14.1	How to Use CMP Transaction Attribute .....	115-24
115.14.2	How to Use Local Transactions .....	115-25
115.14.3	How to Use Nondeferred Changes.....	115-25
115.15	Database Transaction Isolation Levels.....	115-26
115.15.1	What You May Need to Know About General Factors Affecting Transaction Isolation Level	115-26
115.15.1.1	External Applications.....	115-26
115.15.1.2	TopLink Coordinated Cache.....	115-27
115.15.1.3	DatabaseLogin Method setTransactionIsolation .....	115-27

115.15.1.4	Reading Through the Write Connection .....	115-27
115.15.1.4.1	Pessimistic Locking Query .....	115-28
115.15.1.4.2	Unit of Work Method beginTransactionEarly .....	115-28
115.15.1.4.3	ConnectionPolicy Method setShouldUseExclusiveConnection .....	115-28
115.15.1.5	Managing Cache Access .....	115-29
115.15.1.5.1	Isolated Client Session Cache .....	115-29
115.15.1.5.2	ReadObjectQuery .....	115-29
115.15.1.5.3	ReadAllQuery .....	115-29
115.15.1.5.4	Descriptor Method disableCacheHits .....	115-29
115.15.1.5.5	DatabaseQuery Method dontMaintainCache .....	115-29
115.15.1.6	CMP and External Transactions .....	115-29
115.15.2	What You May Need to Know About Read Uncommitted Level .....	115-30
115.15.3	What You May Need to Know About Read Committed Level .....	115-30
115.15.4	What You May Need to Know About Repeatable Read Levels .....	115-30
115.15.5	What You May Need to Know About Serializable Read Levels .....	115-30
115.16	Troubleshooting a Unit of Work .....	115-31
115.16.1	How to Avoid the Use of Post-Commit Clones .....	115-31
115.16.2	How to Determine Whether or Not an Object Is the Cache Object .....	115-31
115.16.3	How to Dump the Contents of a Unit of Work .....	115-32
115.16.4	How to Handle Exceptions .....	115-33
115.16.4.1	Handling Exceptions at Commit Time .....	115-33
115.16.4.2	Handling Exceptions During Conforming .....	115-34
115.16.5	How to Validate a Unit of Work .....	115-34
115.16.5.1	Validating the Unit of Work Before Commit Time .....	115-34

## Part XXVI Creation and Configuration of Projects

### 116 Creating a Project

116.1	Introduction to the Project Creation .....	116-1
116.1.1	How to Create a Project Using Oracle JDeveloper .....	116-1
116.1.2	How to Create a Project Using TopLink Workbench .....	116-2
116.1.2.1	Creating New TopLink Workbench Projects .....	116-2
116.1.3	How to Create a Project Using Java .....	116-3
116.2	Working with Projects .....	116-5
116.2.1	How to Open Existing Projects .....	116-5
116.2.2	How to Save Projects .....	116-6
116.2.2.1	Saving Projects with a New Name or Location .....	116-7
116.2.3	How to Generate the Project Status Report .....	116-8
116.3	Exporting Project Information .....	116-9
116.3.1	How to Export Deployment XML Information Using TopLink Workbench .....	116-9
116.3.2	How to Export Model Java Source Using TopLink Workbench .....	116-10

### 117 Configuring a Project

117.1	Configuring Common Project Options .....	117-2
117.2	Configuring Project Save Location .....	117-2
117.2.1	How to Configure Project Save Location Using TopLink Workbench .....	117-3

117.3	Configuring Project Classpath .....	117-3
117.3.1	How to Configure Project Classpath Using TopLink Workbench .....	117-4
117.4	Configuring Method or Direct Field Access at the Project Level.....	117-5
117.4.1	How to Configure Method or Direct Field Access at the Project Level Using TopLink Workbench	117-5
117.5	Configuring Persistence Type .....	117-6
117.5.1	How to Configure Persistence Type Using TopLink Workbench .....	117-7
117.6	Configuring Default Descriptor Advanced Properties .....	117-8
117.6.1	How to Configure Default Descriptor Advanced Properties Using TopLink Workbench	117-9
117.7	Configuring Existence Checking at the Project Level.....	117-10
117.7.1	How to Configure Existence Checking at the Project Level Using TopLink Workbench.	117-10
117.8	Configuring Project Deployment XML Options .....	117-11
117.8.1	How to Configure Project Deployment XML Options Using TopLink Workbench .....	117-12
117.9	Configuring Model Java Source Code Options .....	117-13
117.9.1	How to Configure Model Java Source Code Options Using TopLink Workbench .....	117-13
117.10	Configuring Cache Type and Size at the Project Level .....	117-14
117.10.1	How to Configure Cache Type and Size at the Project Level Using TopLink Workbench	117-15
117.10.2	How to Configure Cache Type and Size at the Project Level Using Java .....	117-17
117.11	Configuring Cache Isolation at the Project Level.....	117-17
117.11.1	How to Configure Cache Isolation at the Project Level Using TopLink Workbench.....	117-18
117.12	Configuring Cache Coordination Change Propagation at the Project Level .....	117-19
117.12.1	How to Configure Cache Coordination Change Propagation at the Project Level Using TopLink Workbench	117-20
117.13	Configuring Cache Expiration at the Project Level.....	117-22
117.13.1	How to Configure Cache Expiration at the Project Level Using TopLink Workbench.....	117-22
117.14	Configuring Project Comments .....	117-23
117.14.1	How to Configure Project Comments Using TopLink Workbench .....	117-24

## **Part XXVII Creation and Configuration of Descriptors**

### **118 Creating a Descriptor**

118.1	Introduction to Descriptor Creation.....	118-1
118.2	Validating Descriptors .....	118-1
118.3	Generating Java Code for Descriptors .....	118-1

### **119 Configuring a Descriptor**

119.1	Configuring Common Descriptor Options .....	119-2
119.2	Configuring Primary Keys .....	119-4
119.2.1	How to Configure Primary Keys Using TopLink Workbench .....	119-5
119.2.2	How to Configure Primary Keys Using Java.....	119-5

119.2.2.1	Relational Projects .....	119-6
119.2.2.2	EIS Projects .....	119-6
119.3	Configuring Read-Only Descriptors .....	119-6
119.3.1	How to Use Read-Only EJB CMP Entity Beans.....	119-7
119.3.2	How to Configure Read-Only Descriptors Using TopLink Workbench .....	119-7
119.3.3	How to Configure Read-Only Descriptors Using Java .....	119-7
119.4	Configuring Unit of Work Conforming at the Descriptor Level .....	119-8
119.4.1	How to Configure Unit of Work Conforming at the Descriptor Level Using TopLink Workbench	119-8
119.4.2	How to Configure Unit of Work Conforming at the Descriptor Level Using Java .....	119-9
119.5	Configuring Descriptor Alias.....	119-9
119.5.1	How to Configure Descriptor Alias Using TopLink Workbench.....	119-10
119.5.2	How to Configure Descriptor Alias Using Java .....	119-10
119.6	Configuring Descriptor Comments.....	119-11
119.6.1	How to Configure Descriptor Comments Using TopLink Workbench.....	119-11
119.7	Configuring Named Queries at the Descriptor Level .....	119-12
119.7.1	How to Configure Named Queries at the Descriptor Level Using TopLink Workbench	119-13
119.7.1.1	Adding Named Queries .....	119-14
119.7.1.2	Configuring Named Query Type and Parameters .....	119-15
119.7.1.3	Configuring Named Query Selection Criteria .....	119-17
119.7.1.4	Configuring Read All Query Order .....	119-18
119.7.1.5	Configuring Named Query Optimization .....	119-18
119.7.1.6	Configuring Named Query Attributes.....	119-20
119.7.1.6.1	Adding Report Query Attributes.....	119-21
119.7.1.7	Configuring Named Query Group/Order Options .....	119-22
119.7.1.7.1	Adding Ordering Attributes.....	119-22
119.7.1.8	Creating an EIS Interaction for a Named Query .....	119-23
119.7.1.9	Configuring Named Query Options.....	119-25
119.7.1.10	Configuring Named Query Advanced Options .....	119-27
119.7.2	How to Configure Named Queries at the Descriptor Level Using Java.....	119-28
119.8	Configuring Query Timeout at the Descriptor Level .....	119-29
119.8.1	How to Configure Query Timeout at the Descriptor Level TopLink Workbench.....	119-29
119.8.2	How to Configure Query Timeout at the Descriptor Level Java.....	119-30
119.9	Configuring Cache Refreshing.....	119-30
119.9.1	How to Configure Cache Refreshing Using TopLink Workbench.....	119-31
119.9.2	How to Configure Cache Refreshing Using Java.....	119-33
119.10	Configuring Query Keys.....	119-33
119.10.1	How to Configure Query Keys Using TopLink Workbench.....	119-35
119.10.2	How to Configure Query Keys Using Java.....	119-35
119.11	Configuring Interface Query Keys .....	119-37
119.11.1	How to Configure Interface Query Keys Using TopLink Workbench .....	119-38
119.11.2	How to Configure Interface Query Keys Using Java .....	119-38
119.12	Configuring Cache Type and Size at the Descriptor Level.....	119-39
119.12.1	How to Configure Cache Type and Size at the Descriptor Level Using TopLink Workbench	119-40

119.12.2	How to Configure Cache Type and Size at the Descriptor Level Using Java .....	119-41
119.13	Configuring Cache Isolation at the Descriptor Level .....	119-42
119.13.1	How to Configure Cache Isolation at the Descriptor Level Using TopLink Workbench . 119-42	
119.13.2	How to Configure Cache Isolation at the Descriptor Level Using Java .....	119-43
119.14	Configuring Unit of Work Cache Isolation at the Descriptor Level.....	119-43
119.14.1	How to Configure Unit of Work Cache Isolation at the Descriptor Level Using Java..... 119-44	
119.15	Configuring Cache Coordination Change Propagation at the Descriptor Level .....	119-44
119.15.1	How to Configure Cache Coordination Change Propagation at the Descriptor Level Using TopLink Workbench	119-45
119.15.2	How to Configure Cache Coordination Change Propagation at the Descriptor Level Using Java	119-46
119.16	Configuring Cache Expiration at the Descriptor Level.....	119-47
119.16.1	How to Configure Cache Expiration at the Descriptor Level Using TopLink Workbench	119-47
119.16.2	How to Configure Cache Expiration at the Descriptor Level Using Java .....	119-48
119.17	Configuring Cache Existence Checking at the Descriptor Level .....	119-49
119.17.1	How to Configure Cache Existence Checking at the Descriptor Level Using TopLink Workbench	119-49
119.17.2	How to Configure Cache Existence Checking at the Descriptor Level Using Java .....	119-50
119.18	Configuring a Descriptor with EJB CMP and BMP Information.....	119-51
119.18.1	How to Configure a Descriptor with EJB CMP and BMP Information Using TopLink Workbench	119-52
119.18.2	How to Configure a Descriptor with EJB CMP and BMP Information Using Java.....	119-54
119.18.2.1	Configuring CMP Information.....	119-54
119.18.2.2	Configuring BMP Information .....	119-54
119.19	Configuring Reading Subclasses on Queries.....	119-54
119.19.1	How to Configure Reading Subclasses on Queries Using TopLink Workbench	119-55
119.19.2	How to Configure Reading Subclasses on Queries Using Java .....	119-56
119.20	Configuring Inheritance for a Child (Branch or Leaf) Class Descriptor .....	119-57
119.20.1	How to Configure Inheritance for a Child (Branch or Leaf) Class Descriptor Using TopLink Workbench	119-58
119.20.2	How to Configure Inheritance for a Child (Branch or Leaf) Class Descriptor Using Java 119-59	
119.21	Configuring Inheritance for a Parent (Root) Descriptor .....	119-59
119.21.1	How to Configure Inheritance for a Parent (Root) Descriptor Using TopLink Workbench	119-59
119.21.2	How to Configure Inheritance for a Parent (Root) Descriptor Using Java .....	119-61
119.22	Configuring Inheritance Expressions for a Parent (Root) Class Descriptor.....	119-62
119.22.1	How to Configure Inheritance Expressions for a Parent (Root) Class Descriptor Using Java	119-64
119.23	Configuring Inherited Attribute Mapping in a Subclass .....	119-65
119.23.1	How to Configure Inherited Attribute Mapping in a Subclass Using TopLink Workbench	119-65
119.23.2	How to Configure Inherited Attribute Mapping in a Subclass Using Java .....	119-66
119.24	Configuring a Domain Object Method as an Event Handler .....	119-66

119.24.1	How to Configure a Domain Object Method as an Event Handler Using TopLink Workbench	119-67
119.24.2	How to Configure a Domain Object Method as an Event Handler Using Java ..	119-69
119.25	Configuring a Descriptor Event Listener as an Event Handler .....	119-69
119.25.1	How to Configure a Descriptor Event Listener as an Event Handler Using TopLink Workbench	119-71
119.25.2	How to Configure a Descriptor Event Listener as an Event Handler Using Java.....	119-71
119.26	Configuring Locking Policy .....	119-71
119.26.1	How to Configure Locking Policy Using TopLink Workbench .....	119-72
119.26.2	How to Configure Locking Policy Using Java.....	119-74
119.26.2.1	Configuring an Optimistic Locking Policy .....	119-75
119.26.2.2	Configuring Optimistic Locking Policy Cascading .....	119-75
119.26.2.3	Configuring a Pessimistic Locking Policy .....	119-75
119.27	Configuring Returning Policy .....	119-75
119.27.1	How to Configure Returning Policy Using TopLink Workbench.....	119-76
119.27.2	How to Configure Returning Policy Using Java .....	119-77
119.28	Configuring Instantiation Policy .....	119-78
119.28.1	How to Configure Instantiation Policy Using TopLink Workbench .....	119-78
119.28.2	How to Configure Instantiation Policy Using Java .....	119-79
119.29	Configuring Copy Policy .....	119-79
119.29.1	How to Configure Copy Policy Using TopLink Workbench .....	119-80
119.29.2	How to Configure Copy Policy Using Java .....	119-81
119.30	Configuring Change Policy .....	119-81
119.30.1	How to Configure Change Policy Using Java .....	119-82
119.30.1.1	Configuring Deferred Change Detection Policy .....	119-82
119.30.1.2	Configuring Object Change Tracking Policy .....	119-82
119.30.1.3	Configuring Attribute Change Tracking Policy .....	119-83
119.31	Configuring a History Policy .....	119-84
119.31.1	How to Configure a History Policy Using Java .....	119-85
119.31.1.1	Configuring Write Responsibility .....	119-86
119.32	Configuring Wrapper Policy .....	119-86
119.32.1	How to Configure Wrapper Policy Using Java .....	119-87
119.33	Configuring Fetch Groups .....	119-88
119.33.1	How to Configure Fetch Groups Using Java .....	119-89
119.34	Configuring a Descriptor Customizer Class .....	119-89
119.34.1	How to Configure Customizer Class Using Java .....	119-90
119.35	Configuring Amendment Methods.....	119-90
119.35.1	How to Configure Amendment Methods Using TopLink Workbench.....	119-91

## Part XXVIII Creation and Configuration of Mappings

### 120 Creating a Mapping

120.1	Introduction to Mapping Creation .....	120-1
120.2	Creating Mappings Manually During Development .....	120-2
120.2.1	How to Create Mappings Manually During Development Using TopLink Workbench .	120-2

120.2.2	How to Create Mappings During Development Using Java .....	120-2
120.3	Creating Mappings Automatically During Development.....	120-3
120.3.1	How to Create Mappings Automatically During Development Using TopLink Workbench	120-3
120.4	Creating Mappings Automatically During Deployment.....	120-3
120.5	Creating Mappings to Oracle LOB Database Objects.....	120-3
120.5.1	How to Create Mappings to Oracle LOB Database Objects Using the Oracle JDBC Thin Driver	120-4
120.6	Removing Mappings .....	120-5
120.6.1	How to Remove Mappings Using TopLink Workbench .....	120-5
120.6.2	How to Remove Mappings Using Java .....	120-6

## 121 Configuring a Mapping

121.1	Configuring Common Mapping Options.....	121-2
121.2	Configuring Read-Only Mappings .....	121-3
121.2.1	How to Configure Read-Only Mappings Using TopLink Workbench .....	121-4
121.2.2	How to Configure Read-Only Mappings Using Java.....	121-4
121.3	Configuring Indirection (Lazy Loading) .....	121-4
121.3.1	How to Configure Indirection Using TopLink Workbench .....	121-6
121.3.2	How to Configure Indirection Using Java .....	121-7
121.3.2.1	Configuring Value Holder Indirection .....	121-7
121.3.2.2	Configuring Value Holder Indirection with Method Accessing.....	121-8
121.3.2.3	Configuring Value Holder Indirection with JPA.....	121-9
121.3.2.4	Configuring IndirectContainer Indirection .....	121-10
121.3.2.5	Configuring Proxy Indirection .....	121-10
121.4	Configuring XPath.....	121-12
121.4.1	How to Configure XPath Using TopLink Workbench.....	121-13
121.4.1.1	Choosing the XPath.....	121-14
121.5	Configuring a Default Null Value at the Mapping Level .....	121-14
121.5.1	How to Configure a Default Null Value at the Mapping Level Using TopLink Workbench	121-15
121.5.2	How to Configure a Default Null Value at the Mapping Level Using Java .....	121-16
121.6	Configuring Method or Direct Field Accessing at the Mapping Level.....	121-16
121.6.1	How to Configure Method or Direct Field Accessing Using TopLink Workbench.....	121-17
121.6.2	How to Configure Method or Direct Field Accessing Using Java.....	121-18
121.7	Configuring Private or Independent Relationships.....	121-18
121.7.1	How to Configure Private or Independent Relationships Using TopLink Workbench....	121-20
121.7.2	How to Configure Private or Independent Relationships Using Java.....	121-20
121.8	Configuring Mapping Comments .....	121-21
121.8.1	How to Configure Mapping Comments Using TopLink Workbench.....	121-21
121.9	Configuring a Serialized Object Converter .....	121-21
121.9.1	How to Configure a Serialized Object Converter Using TopLink Workbench...	121-22
121.9.2	How to Configure a Serialized Object Converter Using Java .....	121-23
121.10	Configuring a Type Conversion Converter .....	121-23
121.10.1	How to Configure a Type Conversion Converter Using TopLink Workbench ..	121-24

121.10.2	How to Configure a Type Conversion Converter Using Java .....	121-25
121.11	Configuring an Object Type Converter .....	121-25
121.11.1	How to Configure an Object Type Converter Using TopLink Workbench .....	121-26
121.11.2	How to Configure an Object Type Converter Using Java .....	121-27
121.12	Configuring a Simple Type Translator .....	121-28
121.12.1	How to Configure a Simple Type Translator Using TopLink Workbench .....	121-28
121.12.2	How to Configure a Simple Type Translator Using Java .....	121-29
121.13	Configuring a JAXB Typesafe Enumeration Converter .....	121-29
121.13.1	How to Configure a JAXB Typesafe Enumeration Converter Using Java .....	121-30
121.14	Configuring Container Policy .....	121-30
121.14.1	How to Configure Container Policy Using TopLink Workbench .....	121-32
121.14.2	How to Configure Container Policy Using Java .....	121-34
121.15	Configuring Attribute Transformer .....	121-34
121.15.1	How to Configure Attribute Transformer Using TopLink Workbench .....	121-35
121.15.2	How to Configure Attribute Transformer Using Java .....	121-36
121.16	Configuring Field Transformer Associations .....	121-36
121.16.1	How to Configure Field Transformer Associations Using TopLink Workbench .....	121-37
121.16.1.1	Specifying Field-to-Transformer Associations.....	121-38
121.16.2	How to Configure Field Transformer Associations Using Java .....	121-38
121.17	Configuring Mutable Mappings.....	121-39
121.17.1	How to Configure Mutable Mappings Using TopLink Workbench.....	121-39
121.17.2	How to Configure Mutable Mappings Using Java .....	121-40
121.18	Configuring Bidirectional Relationship.....	121-40
121.18.1	How to Configure Bidirectional Relationship Using TopLink Workbench.....	121-41
121.18.2	How to Configure Bidirectional Relationship Using Java .....	121-42
121.19	Configuring the Use of a Single Node .....	121-43
121.19.1	How to Configure the Use of a Single Node Using TopLink Workbench .....	121-43
121.19.2	How to Configure the Use of a Single Node Using Java .....	121-44
121.20	Configuring the Use of CDATA .....	121-44
121.20.1	How to Configure the Use of CDATA Using Java .....	121-45

## **A Troubleshooting a TopLink Application**

A.1	TopLink Support for Oracle Application Server Manageability and Diagnosability.....	A-1
A.1.1	Oracle Application Server Manageability and Diagnosability Logging Enhancements .. A-1	
A.1.2	Oracle Dynamic Monitoring System (DMS) Sensor Enhancements .....	A-2
A.1.3	Manageability and Diagnosability JMX Enhancements .....	A-2
A.2	TopLink Exception Error Reference .....	A-2
A.2.1	Descriptor Exceptions .....	A-3
A.2.2	Concurrency Exceptions .....	A-4
A.2.3	Conversion Exceptions.....	A-4
A.2.4	Database Exceptions.....	A-4
A.2.5	Optimistic Lock Exceptions.....	A-5
A.2.6	Query Exceptions.....	A-5
A.2.7	Validation Exceptions .....	A-5
A.2.8	EJB QL Exceptions .....	A-6
A.2.9	Session Loader Exceptions .....	A-6



A.2.10	EJB Exception Factory Exceptions.....	A-6
A.2.11	Communication Exceptions.....	A-7
A.2.12	EIS Exceptions.....	A-7
A.2.13	JMS Processing Exceptions.....	A-7
A.2.14	Default Mapping Exceptions.....	A-7
A.2.15	Discovery Exceptions.....	A-8
A.2.16	Remote Command Manager Exceptions.....	A-8
A.2.17	Transaction Exceptions.....	A-8
A.2.18	XML Conversion Exceptions.....	A-8
A.2.19	XML Marshal Exceptions.....	A-9
A.2.20	Migration Utility Exceptions.....	A-9
A.2.21	XML Platform Exceptions.....	A-9
A.2.22	Entity Manager Setup Exceptions.....	A-9
A.2.23	EJB JAR XML Exceptions.....	A-10
A.3	TopLink Workbench Error Reference.....	A-10
A.3.1	Miscellaneous Errors (1 – 89, 106 – 133).....	A-10
A.3.2	Project Errors (100 – 102).....	A-12
A.3.3	Descriptor Errors (200 – 399).....	A-12
A.3.4	Mapping Errors (400 – 483).....	A-23
A.3.5	Table Errors (500 – 610).....	A-27
A.3.6	XML Schema Errors (700 – 706).....	A-32
A.3.7	Session Errors (800 – 812).....	A-33
A.3.8	Common Classpath Problems.....	A-34
A.3.9	Database Connection Problems.....	A-35

## Glossary

## Index



---

---

# Preface

*Oracle Fusion Middleware Developer's Guide for Oracle TopLink* explains how to use Oracle TopLink to design, implement, deploy, and optimize an advanced persistence or object-to-XML transformation layer for a wide range of Java 2 Enterprise Edition (Java EE) and Java applications. It describes TopLink mapping metadata, sessions, data access, queries, transactions (both with and without an external transaction controller), and cache.

It describes how to use TopLink application development tools and how to integrate TopLink with a variety of Java EE containers. It also introduces the concepts with which you should be familiar to get the most out of TopLink.

## Audience

*Oracle Fusion Middleware Developer's Guide for Oracle TopLink* is intended for application developers creating applications that use TopLink to manage persistence.

This document assumes that you are familiar with the concepts of object-oriented programming, the Enterprise JavaBeans (EJB) specification, and your own particular Java development environment.

The document also assumes that you are familiar with your particular operating system (such as Windows, UNIX, or other). The general operation of any operating system is described in the user documentation for that system, and is not repeated in this manual.

## Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at <http://www.oracle.com/accessibility/>.

### Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

### Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

### Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at

<http://www.fcc.gov/cgb/consumerfacts/trs.html>, and a list of phone numbers is available at <http://www.fcc.gov/cgb/dro/trsphonebk.html>.

## Related Documentation

For more information, refer to the documentation section of

<http://www.oracle.com/technology/products/ias/toplink/index.html>

## Conventions

The following text conventions are used in this document:

Convention	Meaning
<b>boldface</b>	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

# Part I

---

## TopLink Application Development Overview

This part describes the architecture of TopLink and how TopLink fits into your development process. It contains the following chapters:

- [Chapter 1, "Introduction to TopLink"](#)

This chapter contains general information on TopLink. It discusses the TopLink application space, development process, components, and the TopLink metamodel.

- [Chapter 2, "Introduction to TopLink Application Development"](#)

This chapter contains an overview of the key stages in the TopLink development process including choosing an application architecture and platform, object and relational mapping, building the persistence layer, and deploying and maintaining a TopLink application.



# Introduction to TopLink

Oracle TopLink is an advanced, object-persistence and object-transformation framework that provides development tools and run-time capabilities that reduce development and maintenance efforts, and increase enterprise application functionality.

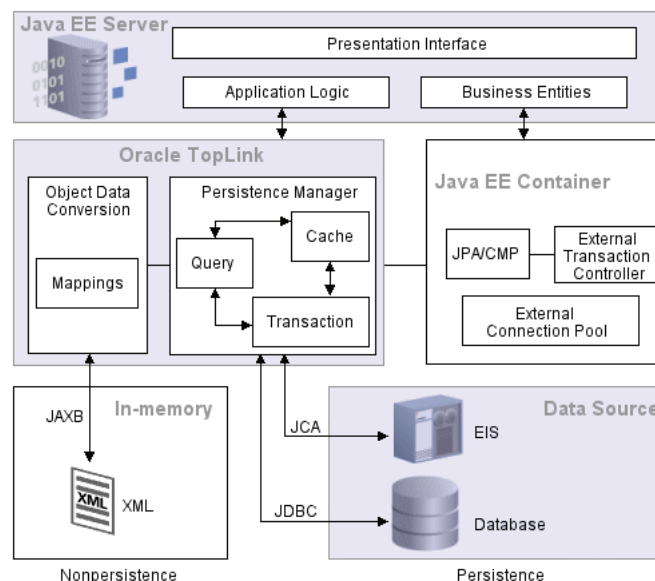
This chapter includes the following sections:

- [What Is TopLink?](#)
- [What Is the Object-Persistence Impedance Mismatch](#)
- [TopLink Key Features](#)
- [TopLink Application Architectures](#)

## 1.1 What Is TopLink?

Oracle TopLink builds high-performance applications that store persistent object-oriented data in a relational database. It successfully transforms object-oriented data into either relational data or Extensible Markup Language (XML) elements.

**Figure 1-1 TopLink Runtime Architecture**



Using TopLink, you can integrate persistence and object-transformation into your application, while staying focused on your primary domain problem by taking advantage of an efficient, flexible, and field-proven solution (see [Section 1.2, "What Is the Object-Persistence Impedance Mismatch"](#)).

TopLink is suitable for use with a wide range of Java 2 Enterprise Edition (Java EE) and Java application architectures (see [Section 1.4, "TopLink Application Architectures"](#)). Use TopLink to design, implement, deploy, and optimize an advanced, object-persistence and object-transformation layer that supports a variety of data sources and formats, including the following:

- Relational—for transactional persistence of Java objects to a relational database accessed using Java Database Connectivity (JDBC) drivers.
- Object-Relational Data Type—for transactional persistence of Java objects to special purpose structured data source representations optimized for storage in object-relational data type databases such as Oracle Database.
- Enterprise information system (EIS)—for transactional persistence of Java objects to a nonrelational data source accessed using a Java EE Connector architecture (JCA) adapter, and any supported EIS record type, including indexed, mapped, or XML.
- XML—for nontransactional, nonpersistent (in-memory) conversion between Java objects and XML Schema Document (XSD)-based XML documents using Java Architecture for XML Binding (JAXB).

TopLink supports EJB 3.0 in Java EE and Java SE environments including integration with a variety of application servers, such as Oracle WebLogic Server, OC4J, SunAS, JBoss, and IBM WebSphere application server. TopLink also includes support for EJB 2.*n* container-managed persistence (CMP) in OC4J.

The extensive suite of development tools that TopLink provides, including Oracle JDeveloper TopLink Editor, Eclipse Dali, and Oracle TopLink Workbench, lets you quickly capture and define object-to-data source and object-to-data representation mappings in a flexible, efficient metadata format (see [Section 2.9, "Working with TopLink Metadata"](#)).

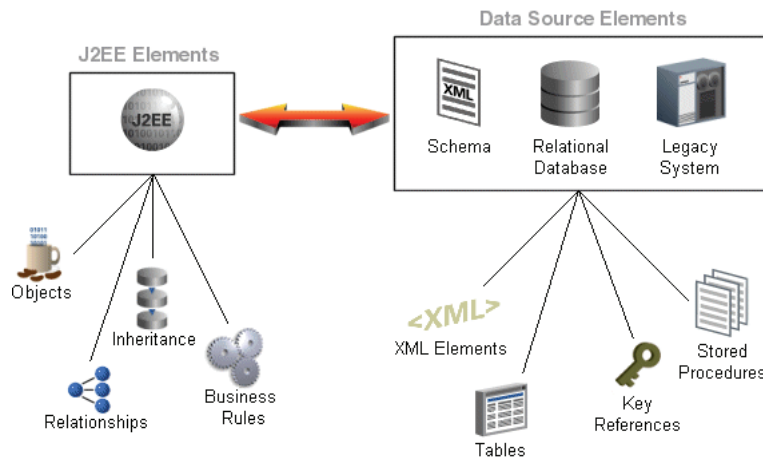
The TopLink runtime lets your application exploit this mapping metadata with a simple session facade that provides in-depth support for data access, queries, transactions (both with and without an external transaction controller), and caching.

For more information about TopLink, see [Section 1.3, "TopLink Key Features"](#).

## 1.2 What Is the Object-Persistence Impedance Mismatch

Java-to-data source integration is a widely underestimated problem when creating enterprise Java applications. This complex problem involves more than simply reading from and writing to a data source. The data source elements include tables, rows, columns, and primary and foreign keys. The Java and Java EE include entity classes (regular Java classes or Enterprise JavaBeans (EJB) entity beans), business rules, complex relationships, and inheritance. In a nonrelational data source, you must match your Java entities with EIS records or XML elements and schemas. These differences (as shown in [Figure 1-2](#)) are known as the object-persistence impedance mismatch.



**Figure 1–2 Solving Object-Persistence Impedance Mismatch**

Successful solution requires bridging these different technologies, and solving the object-persistence impedance mismatch—a challenging and resource-intensive problem. To solve this problem, you must resolve the following issues between Java EE and data source elements:

- Fundamentally different technologies.
- Different skill sets.
- Different staff and ownership for each of the technologies.
- Different modeling and design principles.

As an application developer, you need a product that lets integrate Java applications with any data source, without compromising ideal application design or data integrity. In addition, as a Java developer, you need the ability to store (that is, **persist**) and retrieve business domain objects using a relational database or a nonrelational data source as a repository.

### TopLink Solution

TopLink addresses the disparity between Java objects and data sources. TopLink is a persistence framework that manages relational, object-relational data type, EIS, and XML mappings in a seamless manner. This lets you rapidly build applications that combine the best aspects of object technology and the specific data source.

TopLink lets you do the following:

- Persist Java objects to virtually *any* relational database supported by a JDBC-compliant driver.
- Persist Java objects to virtually *any* nonrelational data source supported by a Java EE Connector architecture (JCA) adapter using indexed, mapped, or XML enterprise information system (EIS) records.
- Perform in-memory conversions between Java objects and XML Schema (XSD) based XML documents using JAXB.
- Map *any* object model to *any* relational or nonrelational schema, using the Oracle TopLink Workbench graphical mapping tool or Oracle JDeveloper TopLink editor.
- Use TopLink successfully, even if you are unfamiliar with SQL or JDBC, because TopLink offers a clear, object-oriented view of data sources.

## 1.3 TopLink Key Features

TopLink provides an extensive and thorough set of features. You can use these features to rapidly build high-performance enterprise applications that are both scalable and maintainable.

Some of the primary features of TopLink are the following:

- Nonintrusive, flexible, metadata-based architecture (see [Section 2.9, "Working with TopLink Metadata"](#))
- Architectural flexibility: Plain Old Java Objects (POJO), Container-Managed Persistence (CMP), as well as Java Persistence API (JPA), Java API for XML Binding (JAXB), Service Data Objects (SDO), and Web services provided by EclipseLink.
- Advanced mapping support and flexibility: relational, object-relational data type, Enterprise Information Systems (EIS), and XML.
- Optimized for highly scalable performance and concurrency with extensive performance tuning options.
- Comprehensive object caching support including cluster integration for some application servers (such as, for example, Oracle Application Server).
- Extensive query capability including: TopLink Expressions framework, Java Persistence Query Language (JP QL), Enterprise JavaBeans Query Language (EJB QL), and native SQL.
- Just-in-time reading
- Object-level transaction support and integration with popular application servers and databases.
- Optimistic and pessimistic locking options and locking policies.
- Comprehensive visual design tools: Oracle JDeveloper TopLink Editor, Eclipse Dali, and Oracle TopLink Workbench.

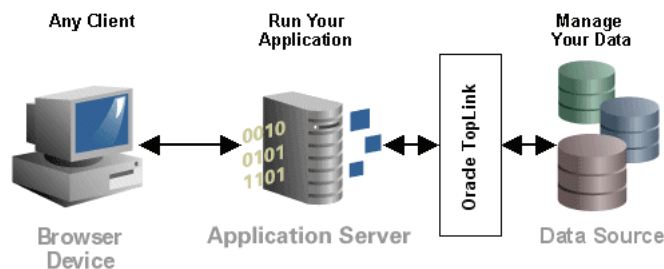
For additional information, see the TopLink page on OTN:

<http://www.oracle.com/technology/products/ias/toplink/index.html>

## 1.4 TopLink Application Architectures

You can use TopLink in a variety of application architectures, including three- and two-tier architectures, with or without Java EE, to access a variety of data types on both relational and nonrelational data sources.

**Figure 1–3 TopLink and Your Application Architecture**



For more information on strategies for incorporating TopLink into your application architecture, see [Section 2.2, "Designing Your Application with TopLink"](#).

This section introduces some of the following common enterprise architectures used by TopLink applications:

- **Three-Tier**

The three-tier (or Java EE Web) application is one of the most common TopLink architectures. This architecture is characterized by a server-hosted environment in which the business logic, persistent entities, and the Oracle TopLink Foundation Library all exist in a single Java Virtual Machine (JVM). See [Section 2.11, "Considering Three-Tier Architecture"](#) for more information.

The most common example of this architecture is a simple three-tier application in which the client browser accesses the application through servlets, JavaServer Pages (JSP) and HTML. The presentation layer communicates with TopLink through other Java classes in the same JVM, to provide the necessary persistence logic. This architecture supports multiple servers in a clustered environment, but there is no separation across JVMs from the presentation layer and the code that invokes the persistence logic against the persistent entities using TopLink.

- **EJB Session Bean Facade**

A popular variation on the three-tier application involves wrapping the business logic, including the TopLink access, in EJB session beans. This architecture provides a scalable deployment and includes integration with transaction services from the host application server. See [Section 2.13, "Considering EJB Session Bean Facade Architecture"](#) for more information.

Communication from the presentation layer occurs through calls to the EJB session beans. This architecture separates the application into different tiers for the deployment. The session bean architecture can persist either Java objects or EJB entity beans.

- **EJB 3.0 Entities with JPA**

The EJB 3.0 specification includes an additional persistence specification called the Java Persistence API (JPA). You can use this API for creating, reading, updating, and deleting plain old Java objects (POJO) within both a compliant EJB 3.0 Java EE container and a standard Java SE 5 (or later) environment.

EclipseLink JPA is a standards-compliant JPA persistence provider built on the EclipseLink foundation library. EclipseLink JPA offers a variety of vendor extensions (annotations and persistence unit properties) that give you full access to the underlying EclipseLink API.

For more information, see the following:

- "Java Persistence API Overview" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Introduction\\_to\\_Java\\_Persistence\\_API\\_%28ELUG%29](http://wiki.eclipse.org/Introduction_to_Java_Persistence_API_%28ELUG%29)
- "Introduction to EclipseLink JPA" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Introduction\\_to\\_EclipseLink\\_JPA\\_%28ELUG%29](http://wiki.eclipse.org/Introduction_to_EclipseLink_JPA_%28ELUG%29)
- [http://wiki.eclipse.org/EclipseLink/UserGuide/Developing\\_JPA\\_Projects\\_%28ELUG%29](http://wiki.eclipse.org/EclipseLink/UserGuide/Developing_JPA_Projects_%28ELUG%29)
- [Section 2.16, "Considering JPA Entity Architecture"](#)

- **EJB Entity Beans with CMP**

TopLink provides CMP support for applications that require the use of EJB entity beans. This support is available on the leading application servers. TopLink CMP support provides you with an EJB 1.0, 1.1, 2.0 and 2.1 CMP solution transparent to the application code, but still offers all the TopLink run-time benefits. See [Section 2.14, "Considering EJB Entity Beans with CMP Architecture"](#) for more information.

Applications can access TopLink-enabled entity beans with container-managed persistence directly from the client, or from within a session bean layer. TopLink also offers the ability to use regular Java objects in relationships with enterprise entity beans.

---

---

**Note:** Even though TopLink fully supports EJB 1.0, 1.1, 2.0, 2.1 and JPA, this edition of *Oracle TopLink Developer's Guide* only focuses on EJB 2.0, 2.1, and JPA. For detailed information on TopLink support for EJB 1.0 and 1.1, refer to the earlier editions of *Oracle TopLink Developer's Guide*.

---

---

- **EJB Entity Beans with BMP**

Another option for using EJB entity beans is to use TopLink BMP in the application. This architecture enables you to access the persistent data through the EJB application programming interface (API), but is platform independent. See [Section 2.15, "Considering EJB Entity Beans with BMP Architecture"](#) for complete information.

The BMP approach is portable—that is, after you create an application, you can move it from one application server platform to another.

- **Web Services**

A Web services architecture is similar to the three-tier or session-bean architecture. However, in a Web services architecture you encapsulate business logic (the service) in a Web service instead of (or in addition to) using session beans. In a Web services architecture, clients communicate with your application using XML.

As in any architecture, you can use TopLink to persist objects to relational or EIS data sources. However, in a Web services architecture you can also use TopLink to map your object model to an XML schema for use with the Web service or as the Web service XML serializer.

See [Section 2.17, "Considering Web Services Architecture"](#) for more information

- **Two-Tier**

A two-tier (or client/server) application is one in which the TopLink application accesses the database directly. Although less common than the other architectures discussed here, TopLink supports this architecture for smaller or embedded data processing applications. See [Section 2.12, "Considering Two-Tier Architecture"](#) for more information.

---

---

# Introduction to TopLink Application Development

This chapter describes how to build a TopLink application, including suggested development processes, architectures, and technologies.

This chapter includes the following sections:

- Introduction to TopLink Application Development
- Designing Your Application with TopLink
- Selecting an Architecture with TopLink
- Building and Using the Persistence Layer
- Deploying the Application
- Optimizing and Customizing the Application
- Persisting Objects
- Working with TopLink Metadata
- Using Weaving
- Considering Three-Tier Architecture
- Considering Two-Tier Architecture
- Considering EJB Session Bean Facade Architecture
- Considering EJB Entity Beans with CMP Architecture
- Considering EJB Entity Beans with BMP Architecture
- Considering JPA Entity Architecture
- Considering Web Services Architecture
- Considering EclipseLink Service Data Objects (SDO) Architecture

## 2.1 Introduction to TopLink Application Development

To ensure the best design for your TopLink application, Oracle recommends that you follow an iterative step-by-step development process. The flexibility of TopLink lets you use *any* development tool.

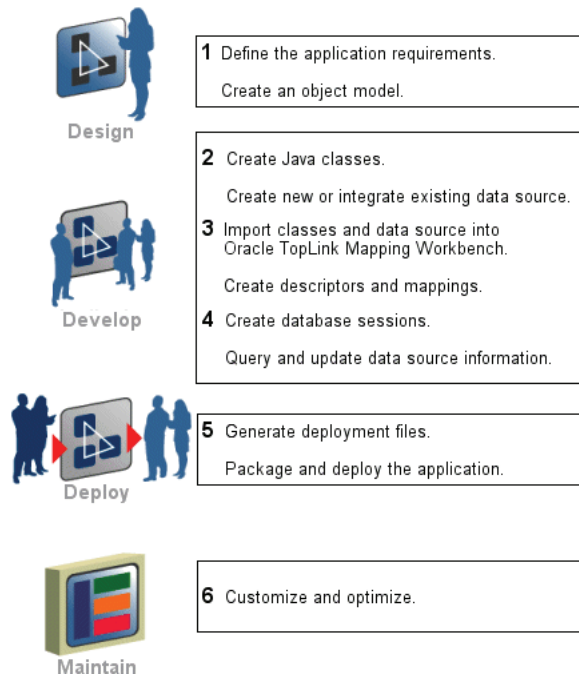
This section describes following recommended development process:

- Typical Development Stages
- Oracle Development Support

## 2.1.1 Typical Development Stages

This section describes the general development stages of a TopLink application. [Figure 2–1](#) illustrates the TopLink development process.

**Figure 2–1 TopLink Development Process**



### Design the Application (1)

Define your application requirements, select an architecture, and determine the target platform. See [Section 2.2, "Designing Your Application with TopLink"](#) for more information. Remember, TopLink works with *any* architecture and *any* platform.

When designing the application, you should also create an object model for the application. See [Section 2.8, "Persisting Objects"](#) for more information. It is important to create the object model *before* using TopLink to map objects, because defining persistent mappings for an incorrect or rapidly changing model can be very difficult.

### Develop the Application (2, 3, 4)

Create the Java classes and decide how the classes should be implemented by the data source. When working with a legacy system, decide how the classes relate to the existing data. If there is no legacy data source to integrate, decide how to store each class in the data source and create the required schema. Alternatively, you may use TopLink to create your initial tables.

Using Oracle JDeveloper TopLink Editor or TopLink Workbench, create descriptors and mappings for the persistent classes. Use TopLink sessions to manipulate the persistent classes, including querying and changing data. See [Part II, "TopLink Development Tools Overview"](#) for more information.

Avoid building all your model's descriptors in a single iteration. Start with a small subset of your classes. Build and test their descriptors, then gradually add new descriptors and relationships. This lets you catch common problems before they proliferate through your entire design.

Write Java code to use database sessions. Sessions are used to query for database objects and write objects to the database. See [Chapter 87, "Introduction to TopLink Sessions"](#) for more information.

### **Deploy the Application (5)**

Generate, package, then deploy the necessary files to your application server. The required information will vary, depending on your environment and architecture. See [Part III, "TopLink Application Deployment"](#) for more information.

### **Maintain the Application (6)**

TopLink includes many options that can enhance application performance. You can customize most aspects of TopLink to suit your requirements. Use advanced TopLink features or write custom querying routines to access the database in specific ways, and to optimize performance. See [Part IV, "Optimization and Customization of a TopLink Application"](#) for more information.

## **2.1.2 Oracle Development Support**

Oracle provides additional support for you as a TopLink developer on the Oracle Technology Network (OTN), including the following:

- Metalink support
- Discussion forums
- How-to documents
- Examples

You must register online before using OTN; registration is free of charge and can be done at

<http://www.oracle.com/technology/membership>

If you already have a user name and password for OTN, then you can go directly to the documentation section of the OTN Web site at

<http://www.oracle.com/technology/docs>

Using your OTN user name and password, you can also access the TopLink developer's forum to post questions and get answers about using TopLink at

<http://forums.oracle.com/forums/forum.jsp?forum=48>

## **2.2 Designing Your Application with TopLink**

When you design your application, you must choose how and where to use TopLink. You can use TopLink to perform a variety of persistence and data transformation functions (see [Section 2.2.1, "How to Use TopLink in Your Application Design"](#)) on a variety of Java-supporting platforms (see [Section 2.2.2, "Target Platforms"](#)). When you design your application architecture, keep these capabilities in mind (see [Section 2.3, "Selecting an Architecture with TopLink"](#)).

### **2.2.1 How to Use TopLink in Your Application Design**

This section describes the basic ways in which you can use TopLink, including the following usage types:

- [Relational Database Usage](#)

- [Object-Relational Data Type Database Usage](#)
- [Oracle XML Database \(XDB\) Usage](#)
- [Enterprise Information System \(EIS\) Usage](#)
- [XML Usage](#)

### **2.2.1.1 Relational Database Usage**

You can use TopLink to persist Java objects to relational databases that support SQL data types accessed using JDBC.

For more information, see [Section 18.1.1, "How to Build Relational Projects for a Relational Database"](#).

### **2.2.1.2 Object-Relational Data Type Database Usage**

You can use TopLink to persist Java objects to object-relational data type databases that support data types specialized for object storage (such as Oracle Database) accessed using JDBC.

For more information, see [Section 18.1.2, "How to Build Relational Projects for an Object-Relational Data Type Database"](#).

### **2.2.1.3 Oracle XML Database (XDB) Usage**

You can use TopLink to persist XML documents to an Oracle XML database using TopLink direct-to-XMLType mappings.

For more information, see [Chapter 18, "Introduction to Relational Projects"](#) and [Section 27.4, "Direct-to-XMLType Mapping"](#).

### **2.2.1.4 Enterprise Information System (EIS) Usage**

You can use TopLink to persist Java objects to an EIS data source using a JCA adapter.

In this scenario, the application invokes EIS data source-defined operations by sending EIS interactions to the JCA adapter. Operations can take (and return) EIS records. Using TopLink EIS descriptors and mappings, you can easily map Java objects to the EIS record types supported by your JCA adapter and EIS data source.

This usage is common in applications that connect to legacy data sources and is also applicable to Web services.

For more information, see [Chapter 71, "Introduction to EIS Projects"](#).

### **2.2.1.5 XML Usage**

You can use TopLink for in-memory, nonpersistent Java object-to-XML transformation with XML Schema (XSD) based XML documents and JAXB.

You can use the TopLink JAXB compiler with your XSD to generate both JAXB-specific artifacts (such as content and element interfaces, implementation classes, and object factory class) and TopLink-specific artifacts (such as sessions and project XML files and TopLink Workbench project). For more information, see [Section 47.1.1, "TopLink Support for Java Architecture for XML Binding \(JAXB\)"](#).

This usage has many applications, including messaging and Web services.

For more information, see [Chapter 47, "Introduction to XML Projects"](#).



## 2.2.2 Target Platforms

TopLink supports any enterprise architecture that uses Java, including the following:

- Java EE
- Spring
- Java Web servers such as Tomcat
- Java clients such as Java SE and Web browsers
- Server Java platforms

Application packaging requirements of the specific target platform (for deployment in the host Java or Java EE environment) influences how you use and configure TopLink. For example, you package a Java EE application in an Enterprise Archive (EAR) file. Within the EAR file, there are several ways to package persistent entities within Web Archive (WAR) and Java Archive (JAR). How you configure TopLink depends, in part, on how you package the application and how you use the host application server class loader.

In addition, TopLink provides custom CMP integration for a variety of application servers.

For detailed information about supported application server versions, custom integration, and configuration requirements, see [Chapter 8, "Integrating TopLink with an Application Server"](#).

## 2.3 Selecting an Architecture with TopLink

This section describes some of the key aspects of application architecture that apply to TopLink and discusses the various options available for each, including the following:

- [Tiers](#)
- [Service Layer](#)
- [Data Access](#)
- [Caching](#)
- [Locking](#)

### 2.3.1 Tiers

This section describes choices you need to make when deciding on how to separate client and server functionality in your application architecture.

These choices can be summarized as follows:

- [Three Tier](#)
  - [Java EE or Non-Java EE](#)
  - [Client](#)
    - \* [Web client](#)
    - \* [XML/Web service client](#)
    - \* [Java \(fat\) client](#)
- [Two Tier](#)

### 2.3.1.1 Three Tier

Oracle recommends a three-tier application architecture. With a three-tier architecture, Oracle recommends using EclipseLink JPA or server sessions and client sessions (see [Section 87.3, "Server and Client Sessions"](#)) and the TopLink unit of work (see [Chapter 113, "Introduction to TopLink Transactions"](#)).

For more information, see [Section 2.11, "Considering Three-Tier Architecture"](#).

**2.3.1.1.1 Java EE or Non-Java EE** You can use TopLink in a Java EE or non-Java EE application architecture. Oracle recommends that you use a Java EE application architecture.

With a Java EE application, you should use external connection pools (see [Section 96.1.6.2, "External Connection Pools"](#)). You may consider using JPA or CMP, EJB session beans, and Java Transaction API (JTA) integration (see [Section 113.1.2.1, "JTA Controlled Transactions"](#)).

With a non-Java EE application, you should use internal connection pools (see [Section 96.1.6.1, "Internal Connection Pools"](#)). You may still consider using JPA.

**2.3.1.1.2 Client** In a three-tier application architecture, you can implement any of the following types of client:

- Web client—Oracle recommends that you implement a Web client.
- XML/Web service client—With this client type, you can use TopLink XML (see [Section 2.2.1.5, "XML Usage"](#)).
- Java (fat) client—With this client type, you can choose the means of communicating with the server:
  - EJB session beans—Oracle recommends this approach. You may consider using the `UnitOfWork` method `mergeClone` to handle merging deserialized objects (see [Section 115.5, "Merging Changes in Working Copy Clones"](#)). The disadvantage of this approach is that your application must handle serialization. Avoid serializing deep object graphs. You should use indirection, also known in JPA as lazy loading (see [Section 121.3, "Configuring Indirection \(Lazy Loading\)"](#)). Consider using the data-transfer-object pattern.
  - XML/Web service—Use TopLink XML (see [Section 2.2.1.5, "XML Usage"](#)).
  - EJB entity bean—Use TopLink CMP integration or BMP support. The disadvantage of this approach is that remote entity beans may not perform or scale well.
  - RMI—You may consider using a TopLink remote session (see [Section 87.9, "Remote Sessions"](#)). The disadvantage of this approach is that a remote session is stateful and may not scale well.

See also [Section 2.3.2, "Service Layer"](#).

### 2.3.1.2 Two Tier

With a two-tier application architecture, Oracle recommends using TopLink database sessions (see [Section 87.8, "Database Sessions"](#)) and the TopLink unit of work (see [Chapter 113, "Introduction to TopLink Transactions"](#)). The disadvantages of this architecture are that it is not Web-enabled and does not scale well to large deployments.

For more information, see [Section 2.12, "Considering Two-Tier Architecture"](#).

## 2.3.2 Service Layer

This section describes choices you need to make when deciding on how to encapsulate your application's business logic (or service).

These choices can be summarized as follows:

- [EJB Session Beans](#)
  - [Stateful](#)
  - [Stateless](#)
- [EJB Entity Beans](#)
  - [Container-Managed Persistence \(CMP\)](#)
  - [Bean-Managed Persistence \(BMP\)](#)
- [JPA Entities](#)
- [Plain Old Java Objects \(POJO\)](#)

See also:

- [Section 2.3.3, "Data Access"](#)
- [Section 2.3.4, "Caching"](#)

### 2.3.2.1 EJB Session Beans

Oracle recommends using EJB session beans.

With EJB session beans, you should use JTA integration (see [Section 113.1.2.1, "JTA Controlled Transactions"](#)) and external connection pools (see [Section 96.1.6.2, "External Connection Pools"](#)). You should acquire a unit of work (see [Section 115.13.1, "How to Acquire a Unit of Work with an External Transaction Service"](#)).

For more information, see [Section 2.13, "Considering EJB Session Bean Facade Architecture"](#).

**2.3.2.1.1 Stateful** If you are using stateful session beans, then note that a reference to a client session cannot be passivated. In this case, you must reacquire a client session (see [Section 87.2.4, "Acquiring a Session at Run Time with the Session Manager"](#)) on activate or per request.

**2.3.2.1.2 Stateless** If you are using stateless session beans, you must acquire new client session (see [Section 87.2.4, "Acquiring a Session at Run Time with the Session Manager"](#)) for each request.

### 2.3.2.2 EJB Entity Beans

EJB entity bean architectures are slightly different from other TopLink architectures, because the EJB entity bean interfaces hide TopLink functionality completely from the client application developer.

You can use entity beans in almost any Java EE application. For TopLink, *how the application uses the entity beans is not important; how entity beans are mapped and implemented is important to TopLink.*

**2.3.2.2.1 Container-Managed Persistence (CMP)** Oracle recommends using entity beans with container-managed persistence. In this case, you should use the TopLink CMP integration for your application server. You must ensure that you are using a Java EE

server that TopLink supports (see [Chapter 8, "Integrating TopLink with an Application Server"](#)).

For more information, see [Section 2.14, "Considering EJB Entity Beans with CMP Architecture"](#).

**2.3.2.2 Bean-Managed Persistence (BMP)** If you are using entity beans with bean-managed persistence, you should use the TopLink BMP integration. The disadvantages of this architecture are that the BMP architecture is restrictive and may not provide good performance.

For more information, see [Section 2.15, "Considering EJB Entity Beans with BMP Architecture"](#).

### 2.3.2.3 JPA Entities

Java Persistence API (JPA) is a specification for persistence in Java EE and Java SE applications. In JPA, a persistent class is referred to as an entity. An entity is a plain old Java object (POJO) class (see [Section 2.3.2.4, "Plain Old Java Objects \(POJO\)"](#)) that is mapped to the database and configured for usage through JPA using annotations, persistence XML, or both.

With JPA, when your application is running inside a container, all of the benefits of the container support and ease of use apply. Note that you can configure the same application to run outside the container.

You can use session beans (see [Section 2.3.2.1, "EJB Session Beans"](#)) as the means for your application to interact with JPA.

EclipseLink JPA is a standards-compliant JPA persistence provider built on the EclipseLink foundation library. EclipseLink JPA offers a variety of vendor extensions (annotations and persistence unit properties) that give you full access to the underlying EclipseLink API to take advantage of additional functionality and performance benefits.

For more information, see the following:

- [Section 2.16, "Considering JPA Entity Architecture"](#)
- [Section 1.4, "TopLink Application Architectures"](#)

### 2.3.2.4 Plain Old Java Objects (POJO)

If you choose to build your service layer with non-EJB Java objects with a Java EE application server, you should use external connection pools (see [Section 96.1.6.2, "External Connection Pools"](#)). If you use a non-Java EE Web server, you should use internal connection pools (see [Section 96.1.6.1, "Internal Connection Pools"](#)). In either case, you may consider using JTA integration (see [Section 113.1.2.1, "JTA Controlled Transactions"](#)).

## 2.3.3 Data Access

This section describes choices you need to make when deciding on what type of data your application architecture must support.

These choices can be summarized as follows:

- [Data Type](#)
- [Multiple Data Sources](#)
- [Isolating Data Access](#)

- [Historical Data Access](#)

See also [Section 2.3.5, "Locking"](#).

### 2.3.3.1 Data Type

You can use TopLink to manage any of the following types of data:

- relational (see [Section 2.2.1.1, "Relational Database Usage"](#));
- object-relational data type (see [Section 2.2.1.2, "Object-Relational Data Type Database Usage"](#));
- Oracle XDB (see [Section 2.2.1.3, "Oracle XML Database \(XDB\) Usage"](#));
- EIS, nonrelational, legacy data (see [Section 2.2.1.4, "Enterprise Information System \(EIS\) Usage"](#));
- XML and Web service data (see [Section 2.2.1.5, "XML Usage"](#)).

### 2.3.3.2 Multiple Data Sources

If your application architecture must access more than one data source, Oracle recommends that you use a session broker (see [Section 87.7, "Session Broker and Client Sessions"](#)) and JTA integration (see [Section 113.1.2.1, "JTA Controlled Transactions"](#)) for two-phase commit.

Alternatively, you may use multiple sessions.

### 2.3.3.3 Isolating Data Access

If your application architecture requires that some data be restricted to a private cache and isolated from the TopLink shared session cache, Oracle recommends that you use an isolated session (see [Section 87.5, "Isolated Client Sessions"](#)). You can also use an isolated session with the Oracle Virtual Private Database (VPD) feature (see [Section 87.5.1, "Isolated Client Sessions and Oracle Virtual Private Database \(VPD\)"](#)).

### 2.3.3.4 Historical Data Access

If your data source maintains past or historical versions of objects, Oracle recommends that you use a TopLink historical session (see [Section 87.6, "Historical Sessions"](#)) to access this historical data so that you can express read queries conditional on how your objects are changing over time.

## 2.3.4 Caching

This section describes choices you need to make when deciding on how to use the TopLink cache (see [Chapter 102, "Introduction to Cache"](#)) in your application architecture.

These choices can be summarized as follows:

- [Cache Type](#)
- [Refreshing](#)
- [Cache Coordination](#)
  - [Protocol](#)
  - [Synchronization](#)

See also [Section 2.3.5, "Locking"](#).

### 2.3.4.1 Cache Type

Choose a cache type (see [Section 102.2.1, "Cache Type and Object Identity"](#)) appropriate for the type of data your application processes. For example, consider a weak identity map for volatile data (see [Section 102.2.1.6, "Guidelines for Configuring the Cache and Identity Maps"](#)).

### 2.3.4.2 Refreshing

Consider how your application architecture may be affected by stale data (see [Section 102.2.3, "Handling Stale Data"](#)): for example, consider using query or descriptor refresh options (see [Section 2.3.4.2, "Refreshing"](#)) or cache invalidation (see [Section 102.2.5, "Cache Invalidation"](#)). Consider using an isolated session's cache (see [Section 87.5, "Isolated Client Sessions"](#)) for volatile data.

Avoid using no identity map (see [Section 102.2.1.5, "No Identity Map"](#)) for objects that are involved in relationships or that require object identity.

### 2.3.4.3 Cache Coordination

TopLink provides a distributed cache coordination feature that allows multiple, possibly distributed, instances of a session to broadcast object changes among each other so that each session's cache is kept up to date (see [Section 102.3, "Cache Coordination"](#)). Before using cache coordination, ensure that it is appropriate for your application (see [Section 102.3.1, "When to Use Cache Coordination"](#)).

**2.3.4.3.1 Protocol** You can configure a coordinated cache to broadcast changes using any of the following communication protocols:

- Java Message Service (JMS)—Oracle recommends using a JMS coordinated cache (see [Section 102.3.3.1, "JMS Coordinated Cache"](#)).
- Remote Method Invocation (RMI)—Oracle recommends that you use RMI cache coordination only if you require synchronous change propagation (see [Section 103.2, "Configuring the Synchronous Change Propagation Mode"](#)). For more information, see [Section 102.3.3.2, "RMI Coordinated Cache"](#).
- Common Object Request Broker Architecture (CORBA)—Currently, TopLink provides support for the Sun ORB (see [Section 102.3.3.3, "CORBA Coordinated Cache"](#)).

**2.3.4.3.2 Synchronization** You can configure synchronization strategy that a coordinated cache uses to determine what it broadcasts when an object changes. You can configure this at the project (see [Section 117.12, "Configuring Cache Coordination Change Propagation at the Project Level"](#)) or descriptor ([Section 119.15, "Configuring Cache Coordination Change Propagation at the Descriptor Level"](#)) level as follows:

- Invalidate changed objects—Propagate an object invalidation that marks the object as invalid in all other sessions. This tells other sessions that they must update their cache from the data source the next time this object is read. Oracle recommends using this synchronization strategy.
- Synchronize changes—Propagate a change notification that contains each changed attribute.
- Synchronize changes and new objects—Propagate a change notification that contains each changed attribute. For new objects, propagate an object creation (along with all the new instance's attributes).

## 2.3.5 Locking

This section describes choices you need to make when deciding on how to use TopLink locking options in your application architecture. Oracle strongly recommends always using a locking policy in a concurrent system (see [Section 119.26, "Configuring Locking Policy"](#)).

These choices can be summarized as follows:

- [Optimistic Locking](#)
- [Pessimistic Locking](#)

If you are building a three-tier application, be aware of how that architecture affects the way you use locking (see [Section 16.4.6, "Locking in a Three-Tier Application"](#)).

For more information, see [Section 16.4, "Descriptors and Locking"](#).

### 2.3.5.1 Optimistic Locking

Oracle recommends using TopLink optimistic locking. With optimistic locking, all users have read access to the data. When a user attempts to write a change, the application checks to ensure the data has not changed since the user read the data.

You can use version (see [Section 16.4.1, "Optimistic Version Locking Policies"](#)) or field (see [Section 16.4.4, "Optimistic Field Locking Policies"](#)) locking policies. Oracle recommends using version locking policies.

### 2.3.5.2 Pessimistic Locking

With pessimistic locking, the first user who accesses the data with the purpose of updating it locks the data until completing the update. The disadvantage of this approach is that it may lead to reduced concurrency and deadlocks.

Consider using pessimistic locking support at the query level (see [Section 119.7.1.9, "Configuring Named Query Options"](#)).

If are using CMP, you may consider using bean-level pessimistic locking support (see [Section 119.7.1.9, "Configuring Named Query Options"](#)).

## 2.4 Building and Using the Persistence Layer

Oracle TopLink requires that classes must meet certain minimum requirements before they can become persistent. TopLink also provides alternatives to most requirements. TopLink uses a nonintrusive approach by employing a metadata architecture that allows for minimal object model intrusions.

This section includes the following information:

- [Implementation Options](#)
- [Persistent Class Requirements](#)
- [Persistence Layer Components](#)
- [How to Use the Persistence Layer](#)

### 2.4.1 Implementation Options

When implementing your persistence layer using TopLink, consider the following options:

- [Using EclipseLink JPA Metatdata, Annotations, and XML](#)

- [Using TopLink Metadata XML](#)
- [Using TopLink Metadata Java API](#)
- [Using Method and Direct Field Access](#)
- [Using Weaving](#)

#### **2.4.1.1 Using EclipseLink JPA Metadata, Annotations, and XML**

When using JPA, you can specify persistence layer components using any combination of standard JPA annotations and `persistence.xml`, EclipseLink JPA annotation extensions, and EclipseLink JPA `persistence.xml` extensions.

For more information, see [Section 2.16, "Considering JPA Entity Architecture"](#).

#### **2.4.1.2 Using TopLink Metadata XML**

Persistence layer components may be generated as metadata from Oracle JDeveloper or TopLink Workbench.

Oracle recommends using Oracle JDeveloper or TopLink Workbench to create the necessary metadata (stored as XML). You can easily export and update the `project.xml` and `sessions.xml` files. This reduces development effort by eliminating the need to regenerate and recompile Java code each time you change the project. With Oracle JDeveloper or TopLink Workbench, you write Java code only for your own application classes and any necessary amendment methods. For information about the XML structure of the `project.xml` and `sessions.xml` files, refer to the appropriate XML schemas (XSD) in the `TOPLINK_HOME/xsds` directory.

TopLink Workbench provides Ant tasks that you can use to integrate TopLink Workbench with your automated builds.

For more information, see the following:

- [Section 2.9, "Working with TopLink Metadata"](#)
- [Section 5.8, "Integrating TopLink Workbench with Apache Ant"](#)

#### **2.4.1.3 Using TopLink Metadata Java API**

Persistence layer components may be coded or generated as Java from Oracle JDeveloper or TopLink Workbench.

To use Java code, you must manually write code for each element of the TopLink project including: project, login, platform, descriptors, and mappings. This may be more efficient if your application is model-based and relies heavily on code generation. Depending on the type of project you are creating, Oracle JDeveloper and TopLink Workbench can export Java code for projects, tables, and your model source.

TopLink Workbench provides Ant tasks that you can use to integrate TopLink Workbench with your automated builds.

For more information, see the following:

- [Section 116.3, "Exporting Project Information"](#)
- [Section 5.8, "Integrating TopLink Workbench with Apache Ant"](#)

#### **2.4.1.4 Using Method and Direct Field Access**

You can access the fields (data members) of a class by using a getter/setter method (also known as property access) or by accessing the field itself directly.



When to use method or direct field access depends on your application design. Consider the following guidelines:

- Use method access outside of a class.
 

This is the natural public API of the class. The getter/setter methods handle any necessary side-effects and the client need not know anything about those details.
- Use direct field access within a class to improve performance.
 

In this case, you are responsible for taking into consideration any side-effects not invoked by bypassing the getter/setter methods.

When considering using method or direct field access, consider the following limitations.

If you enable change tracking on a getter/setter method (for example, you decorate method `setPhone` with `@ChangeTracking`), then TopLink tracks changes accordingly when a client modifies the field (`phone`) using the getter/setter methods.

Similarly, if you enable change tracking on a field (for example, you decorate field `phone` with `@ChangeTracking`), then TopLink tracks changes accordingly when a client modifies the field (`phone`) directly.

However, if you enable change tracking on a getter/setter method (for example, you decorate method `setPhone` with `@ChangeTracking`) and a client accesses the field (`phone`) directly, TopLink does not detect the change. If you choose to code in this style—field access within a class for performance and method access outside of a class—be aware of this limitation.

For more information, see the following:

- [Section 117.4, "Configuring Method or Direct Field Access at the Project Level"](#)
- [Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"](#)
- [Section 113.2.3, "Unit of Work and Change Policy"](#)
- "How to Use the `@ChangeTracking` Annotation" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#How\\_to\\_Use\\_the\\_.40ChangeTracking\\_Annotation](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#How_to_Use_the_.40ChangeTracking_Annotation)
- [Section 119.30, "Configuring Change Policy"](#)

#### 2.4.1.5 Using Weaving

Weaving is a technique of manipulating the byte-code of compiled Java classes.

Weaving is used to enhance both JPA entities and Plain Old Java Object (POJO) classes for such things as lazy loading, change tracking, fetch groups, and internal optimizations.

For more information, see [Section 2.10, "Using Weaving"](#).

## 2.4.2 Persistent Class Requirements

The following requirements apply to plain Java objects:

- You can use direct access on private or protected attributes. For more information on direct and method access, see [Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"](#).

- When using *nontransparent* indirection, the attributes must be of the type `ValueHolderInterface` rather than the original attribute type. The value holder does not instantiate a referenced object until it is needed.
- `TopLink` provides *transparent* indirection for `Collection`, `List`, `Set`, and `Map` attribute types for any collection mappings. Using transparent indirection does not require the use of the `ValueHolderInterface` or any other object model requirements.

If you are using weaving, the `ValueHolderInterface` is not required. For more information, see [Section 2.4.1.5, "Using Weaving"](#).

---

---

**Note:** For EJB 2.0 entity beans with container-managed persistence, the bean requirements are defined by the EJB 2.0 specification; for EJB 2.1 entity beans with container-managed persistence, the bean requirements are defined by the EJB 2.1 specification.

---

---

See [Section 17.2.4, "Indirection \(Lazy Loading\)"](#) for more information on indirection and transparent indirection.

## 2.4.3 Persistence Layer Components

Typically, the `TopLink` persistence layer contains the following components:

- [Mapping Metadata](#)
- [Session](#)
- [Cache](#)
- [Queries and Expressions](#)
- [Transactions](#)

### 2.4.3.1 Mapping Metadata

The `TopLink` application metadata model is based on the `TopLink` project. The project includes descriptors, mappings, and various policies that customize the run-time capabilities. You associate this mapping and configuration information with a particular data source and application by referencing the project from a session.

For more information, see the following:

- [Section 2.9.2, "Creating Project Metadata"](#)
- [Chapter 15, "Introduction to Projects"](#)
- [Chapter 16, "Introduction to Descriptors"](#)
- [Chapter 17, "Introduction to Mappings"](#)

### 2.4.3.2 Session

A session is the primary interface between the client application and `TopLink`, and represents the connection to the underlying data source.

`TopLink` offers several different session types (see [Chapter 87, "Introduction to `TopLink` Sessions"](#)), each optimized for different design requirements and architectures. The most commonly used session is the server session, a session that clients access on the server through a client session. The server session provides a

shared cache and shared connection resources. You define a session with session metadata.

For CMP projects, the TopLink run-time creates and uses a session internally, but your application does not acquire or use this session directly. Depending on the application server you use, you can specify some of the parameters for this internal session.

For EclipseLink JPA projects, the `EntityManager` represents (wraps) the session.

For more information, see the following:

- [Section 2.9.3, "Creating Session Metadata"](#)
- [Section 2.4.4, "How to Use the Persistence Layer"](#)

### 2.4.3.3 Cache

By default, a TopLink session provides an object-level cache that guarantees object identity and enhances performance by reducing the number of times the application needs to access the data source. TopLink provides a variety of cache options, including locking, refresh, invalidation, isolation, and coordination. Using cache coordination, you can configure TopLink to synchronize changes with other instances of the deployed application. You configure most cache options at the session level. You can also configure cache options on a per-query basis or on a descriptor to apply to all queries on the reference class.

For more information, see [Chapter 102, "Introduction to Cache"](#).

### 2.4.3.4 Queries and Expressions

TopLink provides several object and data query types, and offers flexible options for query selection criteria, including the following:

- TopLink expressions
- JPQL (Java Persistence Query Language)
- SQL
- Stored procedures
- Query by example

With these options, you can build any type of query. Oracle recommends using predefined queries to define application queries. Predefined queries are held in the project metadata and referenced by name. This simplifies application development and encapsulates the queries to reduce maintenance costs.

When using entity beans, you can code finders completely using EJB QL (in addition to any of the other TopLink query options), enabling the application to comply with the Java EE specification.

Regardless of the architecture or persistent entity type, you are free to use any of the query options. Oracle JDeveloper and TopLink Workbench provide the simplest way to define queries. Alternatively, you can build queries in code, using the TopLink API.

For more information, see [Chapter 108, "Introduction to TopLink Queries"](#) and [Chapter 110, "Introduction to TopLink Expressions"](#).

### 2.4.3.5 Transactions

TopLink provides the ability to write transactional code isolated from the underlying database and schema by using a **unit of work**, a specific transactional session.

The unit of work isolates changes in a transaction from other threads until it successfully commits the changes to the database. Unlike other transaction mechanisms, the unit of work automatically manages changes to the objects in the transaction, the order of the changes, and changes that might invalidate other TopLink caches. The unit of work manages these issues by calculating a minimal change set, ordering the database calls to comply with referential integrity rules and deadlock avoidance, and merging changed objects into the shared cache. In a clustered environment, the unit of work also synchronizes changes with the other servers in the coordinated cache.

If an application uses entity beans, you do not access the unit of work API directly, but you still benefit from its features: the integration between the TopLink runtime and the Java EE container automatically uses the unit of work to the application's best advantage.

For more information, see [Chapter 108, "Introduction to TopLink Queries"](#).

## 2.4.4 How to Use the Persistence Layer

At run time, your application uses the TopLink metadata (see [Section 2.9, "Working with TopLink Metadata"](#)).

For a POJO (non-CMP) project, your application loads a `session.xml` file at run time using the session manager (see [Chapter 90, "Acquiring and Using Sessions at Run Time"](#)). The `session.xml` file contains a reference to the mapping metadata `project.xml` file. Using the session, your application accesses the TopLink runtime and the `project.xml` mapping metadata.

For a CMP project, the metadata required is dependent upon the Java EE application server you deploy your application to (see [Chapter 9, "Creating TopLink Files for Deployment"](#)). All application servers require an `ejb-jar.xml` and a TopLink project XML file. The session configuration is dependent on the specific Java EE application server.

## 2.5 Deploying the Application

Application packaging (for deployment in the host Java or Java EE environment) influences TopLink use and configuration. For example, you package a Java EE application in an EAR file. Within the EAR file, there are several ways to package persistent entities within WAR and JAR. How you configure TopLink depends, in part, on how you package the application and how you use the class loader of the host application server.

This section discusses packaging and deployment from a TopLink perspective. However, if you deploy your application to a Java EE container, you must configure elements of your application to enable TopLink container support.

This section includes the following information:

- [About Deployments](#)
- [How to Use TopLink in a Java EE Application](#)

For more information, see [Part III, "TopLink Application Deployment"](#).

## 2.5.1 About Deployments

The TopLink approach to deployment involves packaging application files into a single file, such as a JAR file, or an EAR file. This approach lets you create clean and self-contained deployments that do not require significant file management.

After creating these files, deploy the project.

## 2.5.2 How to Use TopLink in a Java EE Application

Although TopLink is an integral part of a Java EE application, in most cases the client does not interact with TopLink directly. Instead, TopLink features are invoked indirectly by way of EJB container callbacks.

As a result, the typical deployment process involves the following steps:

1. Build the project elements, including beans, classes, and data sources.
2. Define the application mappings in Oracle JDeveloper TopLink Editor or TopLink Workbench.
3. Build the application deployment files. Use Oracle JDeveloper or TopLink Workbench to create the files.
4. Package and deploy the application.
5. Add code to the client application to enable it to access the TopLink application.

## 2.6 Optimizing and Customizing the Application

TopLink provides a diverse set of features to optimize performance including the following:

- Enhancing queries
- Tuning the cache
- Scaling to multiple server configuration

You enable or disable most features in the descriptors or session, making any resulting performance gains global.

Using TopLink EIS (see [Section 2.2.1.4, "Enterprise Information System \(EIS\) Usage"](#)), you can integrate a TopLink application with legacy data sources using a JCA adapter. This is the most efficient way to customize a TopLink application to accommodate unusual or nonstandard systems.

Using TopLink XML (see [Section 2.2.1.5, "XML Usage"](#)), you can integrate a TopLink application with legacy data sources using a Web service.

See [Part IV, "Optimization and Customization of a TopLink Application"](#) for details on optimizing and customizing TopLink.

## 2.7 Troubleshooting the Application

See [Appendix A, "Troubleshooting a TopLink Application"](#) for information on troubleshooting all aspects of a TopLink application including development and deployment.

## 2.8 Persisting Objects

This section includes a brief description of relational mapping and provides important information and restrictions to guide object and relational modeling. This information is useful when building TopLink applications.

This section includes information on the following:

- [Application Object Model](#)
- [Data Storage Schema](#)
- [Primary Keys and Object Identity](#)
- [Mappings](#)
- [Foreign Keys and Object Relationships](#)
- [Inheritance](#)
- [Concurrency](#)
- [Caching](#)
- [Nonintrusive Persistence](#)
- [Indirection](#)
- [Mutability](#)

These sections contain additional detail on these features, and explain how to implement and use them with TopLink.

### 2.8.1 Application Object Model

Object modeling refers to the design of the Java classes that represent your application objects. With TopLink, you can use your favorite integrated development environment (IDE) or Unified Modeling Language (UML) modeling tool to define and create your application object model.

Any class that registers a descriptor with a TopLink database session is called a persistent class. TopLink does not require that persistent classes provide public accessor methods for any private or protected attributes stored in the database. Refer to [Section 2.4.2, "Persistent Class Requirements"](#) for more information.

### 2.8.2 Data Storage Schema

Your data storage schema refers to the design that you implement to organize the persistent data in your application. This schema refers to the data itself—not the actual data source (such as a relational database or nonrelational legacy system).

During the design phase of the TopLink application development process (see [Section 2.1.1, "Typical Development Stages"](#)), you should decide how to implement the classes in the data source. When integrating existing data source information, you must determine how the classes relate to the existing data. If no legacy information exists to integrate, decide how you will store each class, then create the necessary schema.

You can also use Oracle JDeveloper (see [Chapter 4, "Using Oracle JDeveloper TopLink Editor"](#)), TopLink Workbench (see [Chapter 5, "Using TopLink Workbench"](#)) or database schema manager (see [Chapter 6, "Using the Schema Manager"](#)) to create the necessary information.

### 2.8.3 Primary Keys and Object Identity

When making objects persistent, each object requires an *identity* to uniquely identify it for storage and retrieval. Object identity is typically implemented using a unique primary key. This key is used internally by TopLink to identify each object, and to create and manage references. Violating object identity can corrupt the object model.

In a Java application, object identity is preserved if each object in memory is represented by one, and only one, object instance. Multiple retrievals of the same object return references to the same object instance—not multiple copies of the same object.

TopLink supports multiple identity maps to maintain object identity (including composite primary keys). Refer to [Section 102.2.1, "Cache Type and Object Identity"](#) for additional information.

### 2.8.4 Mappings

TopLink uses the metadata produced by Oracle JDeveloper or TopLink Workbench (see [Section 2.9, "Working with TopLink Metadata"](#)) to describe how objects and beans map to the data source. This approach isolates persistence information from the object model—you are free to design their ideal object model, and DBAs are free to design their ideal schema.

You use Oracle JDeveloper or TopLink Workbench to create and manage the mapping information. At run time, TopLink uses the metadata to seamlessly and dynamically interact with the data source, as required by the application.

TopLink provides an extensive mapping hierarchy that supports the wide variety of data types and references that an object model might contain. For more information, see [Chapter 17, "Introduction to Mappings"](#).

### 2.8.5 Foreign Keys and Object Relationships

A **foreign key** is a combination of columns that reference a unique key, usually the primary key, in another table. Foreign keys can be any number of fields (similar to primary key), all of which are treated as a unit. A foreign key and the primary parent key it references must have the same number and type of fields.

Foreign keys represents relationships from a column or columns in one table to a column or columns in another table. For example, if every `Employee` has an attribute `address` that contains an instance of `Address` (which has its own descriptor and table), the one-to-one mapping for the `address` attribute would specify foreign key information to find an address for a particular `Employee`.

Refer to [Section 28.7, "Configuring Table and Field References \(Foreign and Target Foreign Keys\)"](#) for more information.

### 2.8.6 Inheritance

Object-oriented systems allow classes to be defined in terms of other classes. For example: motorcycles, sedans, and vans are all *kinds of vehicles*. Each of the vehicle types is a *subclass* of the `Vehicle` class. Similarly, the `Vehicle` class is the *superclass* of each specific vehicle type. Each subclass inherits attributes and methods from its superclass (in addition to having its own attributes and methods).

Inheritance provides several application benefits, including the following:

- Using subclasses to provide specialized behaviors from the basis of common elements provided by the superclass. By using inheritance, you can reuse the code in the superclass many times.
- Implementing *abstract* superclasses that define generic behaviors. This abstract superclass may define and partially implement behavior, while allowing you to complete the details with specialized subclasses.

Refer to [Section 119.20, "Configuring Inheritance for a Child \(Branch or Leaf\) Class Descriptor"](#) and [Section 119.23, "Configuring Inherited Attribute Mapping in a Subclass"](#) for detailed information on using inheritance with TopLink.

## 2.8.7 Concurrency

To have concurrent clients logged in at the same time, the server must spawn a dedicated thread of execution for each client. Java EE application servers do this automatically. Dedicated threads enable each client to work without having to wait for the completion of other clients. TopLink ensures that these threads do not interfere with each other when they make changes to the identity map or perform database transactions.

Using the TopLink `UnitOfWork` class, your client can make transactional changes in an isolated and thread safe manner. The unit of work manages clones for the objects you modify to isolate each client's work from other concurrent clients and threads. The unit of work is essentially an object-level transaction mechanism that maintains all of the ACID (Atomicity, Consistency, Isolation, Durability) transaction principles as a database transaction. For more information on the unit of work, see [Chapter 113, "Introduction to TopLink Transactions"](#).

TopLink supports configurable optimistic and pessimistic locking strategies to let you customize the type of locking that the TopLink concurrency manager uses. For more information, see [Section 16.4, "Descriptors and Locking"](#).

## 2.8.8 Caching

TopLink caching improves application performance by automatically storing data returned as objects from the database for future use. This caching provides several advantages:

- Reusing Java objects that have been previously read from the database minimizes database access
- Minimizing SQL calls to the database when objects already exist in the cache
- Minimizing network access to the database
- Setting caching policies a class-by-class and bean-by-bean basis
- Basing caching options and behavior on Java garbage collection

TopLink supports several caching policies to provide extensive flexibility. You can fine-tune the cache for maximum performance, based on individual application performance. Refer to [Part XXIII, "Cache"](#) for complete information.

## 2.8.9 Nonintrusive Persistence

The TopLink nonintrusive approach of achieving persistence through a metadata architecture (see [Section 2.9, "Working with TopLink Metadata"](#)) means that there are almost no object model intrusions.

To persist Java objects, TopLink does not require any of the following:



- Persistent superclass or implementation of persistent interfaces
- Store, delete, or load methods required in the object model
- Special persistence methods
- Generating source code into or wrapping the object model

When using entity beans with container-managed persistence, TopLink does not require any additional intrusion to the object model, other than the CMP specification requirements.

See [Section 2.4, "Building and Using the Persistence Layer"](#) for additional information on this nonintrusive approach.

## 2.8.10 Indirection

An indirection object takes the place of an application object so the application object is not read from the database until it is needed. Using indirection, or lazy loading in JPA, allows TopLink to create *stand-ins* for related objects. This results in significant performance improvements, especially when the application requires the contents of only the retrieved object rather than all related objects.

Without indirection, each time the application retrieves a persistent object, it also retrieves *all* the objects referenced by that object. This may result in lower performance for some applications.

---

---

**Note:** Oracle strongly recommends that you use indirection in all situations.

---

---

TopLink provides several indirection models, such as proxy indirection, transparent indirection, and value holder indirection. TopLink also provides indirection support for EJB (see [Section 17.2.4.6, "Indirection and EJB 2.n CMP"](#)).

See [Section 17.2.4, "Indirection \(Lazy Loading\)"](#) for more information.

## 2.8.11 Mutability

Mutability is a property of a complex field that specifies whether or not the field value may be changed or not changed as opposed to replaced.

An immutable mapping is one in which the mapped object value cannot change unless the object ID of the object changes: that is, unless the object value is replaced by another object value altogether.

A mutable mapping is one in which the mapped object value can change without changing the object ID of the object.

By default, TopLink assumes the following:

- all `TransformationMapping` instances are mutable;
- all JPA `@Basic` mapping types, except `Serializable` types, are immutable (including `Date` and `Calendar` types);
- all JPA `@Basic` mapping `Serializable` types are mutable.

Whether a value is immutable or mutable largely depends on how your application uses your persistent classes. For example, by default, TopLink assumes that a persistent field of type `Date` is immutable: this means that as long as the value of the field has the same object ID, TopLink assumes that the value has not changed. If your

application uses the set methods of the `Date` class, you can change the state of the `Date` object value without changing its object ID. This prevents TopLink from detecting the change. To avoid this, you can configure a mapping as mutable: this tells TopLink to examine the state of the persistent value, not just its object ID.

You can configure the mutability of the following:

- TransformationMapping instances;
- any JPA `@Basic` mapping type (including `Date` and `Calendar` types) individually;
- all `Date` and `Calendar` types.

Mutability can affect change tracking performance. For example, if a transformation mapping maps a mutable value, TopLink must clone and compare the value in a unit of work (see [Section 119.29, "Configuring Copy Policy"](#)). If the mapping maps a simple immutable value, you can improve unit of work performance by configuring mapping as immutable.

Mutability also affects weaving. TopLink can only weave an attribute change tracking policy for immutable mappings.

For more information, see the following:

- [Section 113.2.3, "Unit of Work and Change Policy"](#)
- [Section 2.10, "Using Weaving"](#)
- "How to Use the `@Mutable` Annotation" of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#How\\_to\\_Use\\_the\\_.40Mutable\\_Annotation](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#How_to_Use_the_.40Mutable_Annotation)
- [Section 121.17, "Configuring Mutable Mappings"](#)

## 2.9 Working with TopLink Metadata

The TopLink metadata is the bridge between the development of an application and its deployed run-time environment. Capture the metadata using the following:

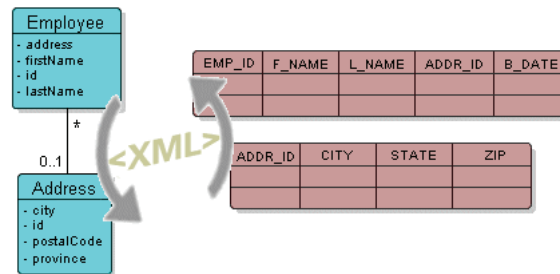
- JPA annotations, `persistence.xml`, `orm.xml`, and EclipseLink JPA annotation and `persistence.xml` property extensions: the EclipseLink JPA persistence provider interprets all these sources of metadata to create an in-memory session and project at run time.
- Oracle JDeveloper TopLink Editor or TopLink Workbench (see [Section 2.9.2, "Creating Project Metadata"](#) and [Section 2.9.3, "Creating Session Metadata"](#)) to create `sessions.xml` and `project.xml` files which you pass to the TopLink run-time environment.
- Java and the TopLink API (this approach is the most labor-intensive).

The metadata lets you pass configuration information into the run-time environment. The run-time environment uses the information in conjunction with the persistent classes (Java objects, JPA entities, or EJB entity beans) and the code written with the TopLink API, to complete the application.

Using EclipseLink JPA, you also have the option of specifying your metadata using `sessions.xml` and `project.xml` while accessing your persistent classes using JPA and an `EntityManager`. For more information, see "What You May Need to Know About EclipseLink JPA Overriding Mechanisms" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_)

[%28ELUG%29#What\\_You\\_May\\_Need\\_to\\_Know\\_About\\_EclipseLink\\_JPA\\_Overriding\\_Mechanisms](#)

**Figure 2–2 TopLink Metadata**



This section describes the following:

- [Advantages of the TopLink Metadata Architecture](#)
- [Creating Project Metadata](#)
- [Creating Session Metadata](#)
- [Deploying Metadata](#)

### 2.9.1 Advantages of the TopLink Metadata Architecture

The TopLink metadata architecture provides many important benefits, including the following:

- Stores mapping information in XML descriptors—not in the domain model objects
- By using the metadata, TopLink does not intrude in the object model or the database schema
- Allows you to design the object model as needed, without forcing any specific design
- Allows DBAs to design the database as needed, without forcing any specific design
- Does not rely on code-generation (which can cause serious design, implementation, and maintenance issues)
- Is unobtrusive: adapts to the object model and database schema, rather than requiring you to design their object model or database schema to suit TopLink

Using EclipseLink JPA, you have the flexibility of expressing persistence metadata using standard JPA annotations, deployment XML, or both and you can optionally take advantage of EclipseLink JPA annotation and `persistence.xml` property extensions.

### 2.9.2 Creating Project Metadata

A TopLink project contains the mapping metadata that the TopLink runtime uses to map objects to a data source. The project is the primary object used by the TopLink runtime.

This section describes the principal contents of project metadata, including the following:

- [Descriptors and Mappings](#)

### ■ Data Source Login Information

Using EclipseLink JPA, the TopLink runtime constructs an in-memory project based on any combination of JPA annotations, `persistence.xml`, `orm.xml`, and EclipseLink JPA annotation and `persistence.xml` property extensions. The use of a `project.xml` file is optional (see "What You May Need to Know About EclipseLink JPA Overriding Mechanisms" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#What\\_You\\_May\\_Need\\_to\\_Know\\_About\\_EclipseLink\\_JPA\\_Overriding\\_Mechanisms](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#What_You_May_Need_to_Know_About_EclipseLink_JPA_Overriding_Mechanisms)).

For more information about creating `project.xml` metadata, see [Section 9.1.1, "project.xml File"](#).

### 2.9.2.1 Descriptors and Mappings

TopLink maps persistent entities to the database in the application, using the descriptors and mappings you build with Oracle JDeveloper TopLink Editor or TopLink Workbench. These tools support several approaches to project development, including the following:

- Importing classes and tables for mapping
- Importing classes and generating tables and mappings
- Importing tables and generating classes and mappings
- Creating both class and table definitions

TopLink Workbench supports all these options. The most common solution is to develop the persistent entities using a development tool, such as an integrated development environment (IDE) like Oracle JDeveloper, or a modeling tool, and to develop the relational model through appropriate relational design tools. You then use Oracle JDeveloper TopLink Editor or TopLink Workbench to construct mappings that relate these two models.

Although Oracle JDeveloper TopLink Editor and TopLink Workbench do offer the ability to generate persistent entities or the relational model components for an application, these utilities are intended only to assist in rapid initial development strategies—not complete round-trip application development.

For more information, see [Chapter 16, "Introduction to Descriptors"](#) and [Chapter 17, "Introduction to Mappings"](#).

**2.9.2.1.1 Amending Descriptors** An amendment method lets you implement a TopLink feature that is not currently supported by Oracle JDeveloper TopLink Editor or TopLink Workbench. Simply write a Java method to amend the descriptor after it is loaded, and specify the method in Oracle JDeveloper TopLink Editor or TopLink Workbench for inclusion in the project metadata. See [Section 119.35, "Configuring Amendment Methods"](#) for detailed information on implementing an amendment method for a TopLink descriptor.

### 2.9.2.2 Data Source Login Information

For POJO projects, you configure a session login in the session metadata that specifies the information required to access the data source (see [Section 2.9.3, "Creating Session Metadata"](#)).

For CMP projects, the project contains a deployment login that specifies the information required to access the data source.

For more information, see [Section 15.2.4, "Projects and Login"](#).

### 2.9.3 Creating Session Metadata

A TopLink session contains a reference to a particular `project.xml` file, plus the information required to access the data source. The session is the primary object used by your application to access the features of the TopLink runtime.

The agent responsible for creating and accessing session metadata differs depending on whether or not you are creating a CMP project. In a POJO project, your application acquires and accesses a session directly (see [Section 9.1.2.2, "POJO Applications and Session Metadata"](#)). In a CMP project, your application indirectly accesses a session acquired internally by the TopLink runtime (see [Section 9.1.2.4, "CMP Applications and Session Metadata"](#)).

Using EclipseLink JPA, the TopLink runtime constructs an in-memory session based on any combination of JPA annotations, `persistence.xml`, `orm.xml`, and EclipseLink JPA annotation and `persistence.xml` property extensions. The use of a `sessions.xml` file is optional (see "What You May Need to Know About EclipseLink JPA Overriding Mechanisms" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#What\\_You\\_May\\_Need\\_to\\_Know\\_About\\_EclipseLink\\_JPA\\_Overriding\\_Mechanisms](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#What_You_May_Need_to_Know_About_EclipseLink_JPA_Overriding_Mechanisms)).

### 2.9.4 Deploying Metadata

The `project.xml` and `sessions.xml` file are packaged for deployment differently according to the type of application you are deploying.

For more information, see the following:

- [Chapter 9, "Creating TopLink Files for Deployment"](#)
- [Chapter 10, "Packaging a TopLink Application"](#)

Using EclipseLink JPA, you also have the option of specifying your metadata using `sessions.xml` and `project.xml` while accessing your persistent classes using JPA and an `EntityManager`. For more information, see "What You May Need to Know About EclipseLink JPA Overriding Mechanisms" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#What\\_You\\_May\\_Need\\_to\\_Know\\_About\\_EclipseLink\\_JPA\\_Overriding\\_Mechanisms](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#What_You_May_Need_to_Know_About_EclipseLink_JPA_Overriding_Mechanisms).

## 2.10 Using Weaving

Weaving is a technique of manipulating the byte-code of compiled Java classes. Weaving is used to enhance both JPA entities and Plain Old Java Object (POJO) classes for such things as lazy loading, change tracking, fetch groups, and internal optimizations.

This section describes the following:

- [Configuring Dynamic Weaving Using EclipseLink Agent](#)
- [Configuring Static Weaving](#)
- [Disabling Weaving Using TopLink Persistence Unit Properties](#)
- [Packaging a POJO Application for Weaving](#)
- [What You May Need to Know About Weaving and POJO Classes](#)
- [What You May Need to Know About Weaving and Java EE Application Servers](#)

## 2.10.1 Configuring Dynamic Weaving Using EclipseLink Agent

When using EclipseLink JPA outside of an EJB 3.0 container, consider dynamic weaving. For more information, see "How to Configure Dynamic Weaving for JPA Entities Using EclipseLink Agent" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#How\\_to\\_Configure\\_Dynamic\\_Weaving\\_for\\_JPA\\_Entities\\_Using\\_the\\_EclipseLink\\_Agent](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#How_to_Configure_Dynamic_Weaving_for_JPA_Entities_Using_the_EclipseLink_Agent)

### 2.10.1.1 To Configure Dynamic Weaving Using EclipseLink Agent

For information, see the following:

- "How to Configure Dynamic Weaving for JPA Entities Using EclipseLink Agent" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#How\\_to\\_Configure\\_Dynamic\\_Weaving\\_for\\_JPA\\_Entities\\_Using\\_the\\_EclipseLink\\_Agent](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#How_to_Configure_Dynamic_Weaving_for_JPA_Entities_Using_the_EclipseLink_Agent)
- Section 2.10.4, "Packaging a POJO Application for Weaving"

## 2.10.2 Configuring Static Weaving

Consider this option to weave all applicable class files at build time so that you can deliver prewoven class files. For more information, see "How to Configure Static Weaving for JPA Entities" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#How\\_to\\_Configure\\_Static\\_Weaving\\_for\\_JPA\\_Entities](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#How_to_Configure_Static_Weaving_for_JPA_Entities).

Note that for weaving, you use a `persistence.xml` file in both JPA and POJO applications.

For information on packaging and deployment of POJO applications, see Section 2.10.4, "Packaging a POJO Application for Weaving".

## 2.10.3 Disabling Weaving Using TopLink Persistence Unit Properties

To disable weaving, you use persistence unit properties in both JPA and POJO applications. For more information, see "How to Disable Weaving Using EclipseLink Persistence Unit Properties" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#How\\_to\\_Disable\\_Weaving\\_Using\\_EclipseLink\\_Persistence\\_Unit\\_Properties](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#How_to_Disable_Weaving_Using_EclipseLink_Persistence_Unit_Properties).

For information on packaging and deployment of POJO application, see Section 2.10.4, "Packaging a POJO Application for Weaving".

## 2.10.4 Packaging a POJO Application for Weaving

To package a POJO application for weaving, you create a JAR that contains a `sessions.xml` file and a `persistence.xml` file.

### 2.10.4.1 To Package a POJO Application for Weaving

1. Create a `sessions.xml` file for your application.  
For more information, [Chapter 87, "Introduction to TopLink Sessions"](#).
2. Create a `persistence.xml` file for your application and reference your `sessions.xml` file, as [Example 2-1](#) shows.

**Example 2–1 persistence.xml File for a EclipseLink JPA Application**

```

<persistence>
  <persistence-unit name="appname">
    <exclude-unlisted-classes>>false</exclude-unlisted-classes>
    <properties>
      <property
        name="eclipselink.session-name"
        value="appname-session"
      >
    </property>
      <property
        name="eclipselink.sessions-xml"
        value="sessions.xml"
      >
    </property>
    </properties>
  </persistence-unit>
</persistence>

```

3. Create a JAR file that contains your POJO classes, `sessions.xml` file, and `persistence.xml` file, as [Example 2–2](#) shows.

Put both the `persistence.xml` and `sessions.xml` file in a `META-INF` directory.

**Example 2–2 JAR File for a POJO Application**

```

appname.jar
  META-INF
    persistence.xml
    sessions.xml
  *.java

```

4. Weave the JAR.

For more information, see the following:

- [Section 2.10.1, "Configuring Dynamic Weaving Using EclipseLink Agent"](#)
- [Section 2.10.2, "Configuring Static Weaving"](#)

**2.10.5 What You May Need to Know About Weaving and POJO Classes**

TopLink uses weaving to enable the following for POJO classes:

- lazy loading (indirection): see [Section 121.3, "Configuring Indirection \(Lazy Loading\)"](#)
- change tracking: see [Section 119.30, "Configuring Change Policy"](#)
- fetch groups: see [Section 119.33, "Configuring Fetch Groups"](#)
- internal optimizations.

TopLink weaves all the POJO classes in the JAR you create when you package a POJO application for weaving. For more information, see [Section 2.10.4, "Packaging a POJO Application for Weaving"](#).

TopLink weaves all the classes defined in the `persistence.xml` file. That is the following:

- all the classes you list in the `persistence.xml` file;
- all classes relative to the JAR containing the `persistence.xml` file if element `<exclude-unlisted-classes>` is `false`.

## 2.10.6 What You May Need to Know About Weaving and Java EE Application Servers

The default TopLink weaving behavior applies in any Java EE JPA-compliant application server using the EclipseLink JPA persistence provider.

To change this behavior, modify your `persistence.xml` (for your JPA entities or POJO classes) to use EclipseLink JPA properties, EclipseLink JPA annotations, or both.

For lazy loading (indirection) differences between Java EE and Java SE applications, see "EclipseLink JPA Support for Lazy Loading by Mapping Type" table of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#Table\\_19-33](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#Table_19-33)

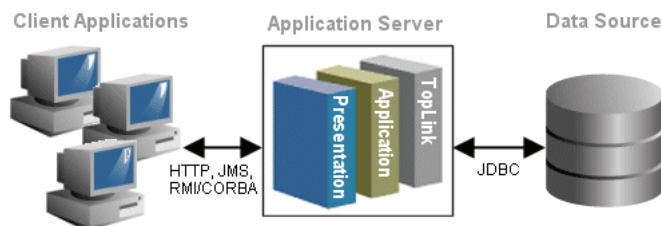
## 2.11 Considering Three-Tier Architecture

The three-tier Web application architecture generally includes the connection of a server-side Java application to the database through a JDBC connection (see [Figure 2-3](#)). In this pattern, TopLink resides within a Java server (a Java EE server or a custom server), with several possible server integration points. The application can support Web clients such as servlets, Java clients, and generic clients using XML or Common Object Request Broker Architecture (CORBA).

The three-tier application is a common architecture in which TopLink resides within a Java server (either a Java EE server or a custom server). In this architecture, the server session provides clients with shared access to JDBC connections and a shared object cache. Because it resides on a single JVM, this architecture is simple and easily scalable. The TopLink persistent entities in this architecture are generally Java objects.

This architecture often supports Web-based applications in which the client application is a Web client, a Java client, or a server component.

**Figure 2-3 Three Tier Architecture**



Although not all three-tier applications are Web-based, this architecture is ideally suited to distributed Web applications. In addition, although it is also common to use EJB in a Web application, this TopLink architecture does not.

### 2.11.1 Example Implementations

Examples of three-tier architecture implementation include the following:

- A Model-View-Controller Model 2 architectural design pattern that runs in a Java EE container with servlets and JSP that uses TopLink to access data without EJB.
- A Swing or Abstract Window Toolkit (AWT) client that connects to a server-side Java application through RMI, without an application server or container.



## 2.11.2 Advantages and Disadvantages

The three-tier Web application architecture offers the following advantages:

- High performance, lightweight persistent objects
- High degree of flexibility in deployment platform and configuration

The disadvantage of this architecture is it is less standard than EJB.

## 2.11.3 Variation Using Remote Sessions

TopLink includes a session type called remote session. The session offers the full session API and contains a cache of its own, but exists on the client system rather than on the TopLink server. Communications can be configured to use RMI or RMI-Internet Inter-Object Request Broker Protocol (IIOP).

Remote session operations require a corresponding client session on the server.

Although this is an excellent option for you if you wish to simplify the access from the client tier to the server tier, it is less scalable than using a client session and does not easily allow changes to server-side behavior.

For more information, see [Section 87.9, "Remote Sessions"](#).

## 2.11.4 Technical Challenges

The three-tier application with a stateless client presents several technical challenges, including the following:

- Transaction management in a stateless environment

A common design practice is to delimit client requests within a single unit of work (transactional session). In a stateless environment, this may affect how you design the presentation layer. For example, if a client requires multiple pages to collect information for a transaction, then the presentation layer must retain the information from page to page until the application accumulates the full set of changes or requests. At that point, the presentation layer invokes the unit of work to modify the database.

- Optimistic locking in a stateless environment

In a stateless environment, take care to avoid processing out-of-date (stale) data. A common strategy for avoiding stale data is to implement optimistic locking, and store the optimistic lock values in the object.

This solution requires careful implementation if the stateless application serializes the objects, or sends the contents of the object to the client in an alternative format. In this case, transport the optimistic lock values to the client in the HTTP contents of an edit page. You must then use the returned values in any write transaction to ensure that the data did not change while the client was performing its work.

For more information about locking, see [Section 119.26, "Configuring Locking Policy"](#).

- External JDBC pools

By default, TopLink manages its own connection pools. You can also configure TopLink to use connection pooling offered by the host application server. This feature is useful for shared connection pools and is required for JTA/JTS integration (see [Section 97.4, "Configuring External Connection Pooling"](#)).

- JTA/JTS Integration

JTA and JTS are standard Java components that enable sessions to participate in distributed transactions. You must configure TopLink to use JTA/JTS to use session beans in the architecture (see [Section 115.13, "Integrating the Unit of Work with an External Transaction Service"](#)).

- Cache coordination

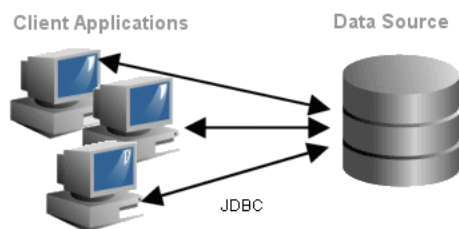
If you choose to use multiple servers to scale your application, you may require TopLink cache coordination (see [Chapter 102, "Introduction to Cache"](#)).

## 2.12 Considering Two-Tier Architecture

A two-tier application generally includes a Java client that connects directly to the database through TopLink. The two-tier architecture is most common in complex user interfaces with limited deployment. The database session provides TopLink support for two-tier applications.

For more information, see [Chapter 87, "Introduction to TopLink Sessions"](#).

**Figure 2–4 Two-Tier Architecture**



Although the two-tier architecture is the simplest TopLink application pattern, it is also the most restrictive, because each client application requires its own session. As a result, two-tier applications do not scale as easily as other architectures.

Two-tier applications are often implemented as user interfaces that directly access the database (see [Figure 2–4](#)). They can also be non-interface processing engines. In either case, the two-tier model is not as common as the three-tier model.

The following are key elements of an efficient two-tier (client-server) architecture with TopLink:

- Minimal dedicated connections from the client to the database
- An isolated object cache

### 2.12.1 Example Implementations

An example of a two-tier architecture implementation is a Java user interface (Swing/AWT) and batch data processing.

### 2.12.2 Advantages and Disadvantages

The advantage of the two-tier design is its simplicity. The TopLink database session that builds the two-tier architecture provides all the TopLink features in a single session type, thereby making the two-tier architecture simple to build and use.

The most important limitation of the two-tier architecture is that it is not scalable, because each client requires its own database session.

### 2.12.3 Technical Challenges

The current trend toward multitiered Web applications makes the two-tier architecture less common in production systems, but no less viable. Because there is no shared cache in a two-tier system, you risk encountering stale data if you run multiple instances of the application. This risk increases as the number of individual database sessions increases.

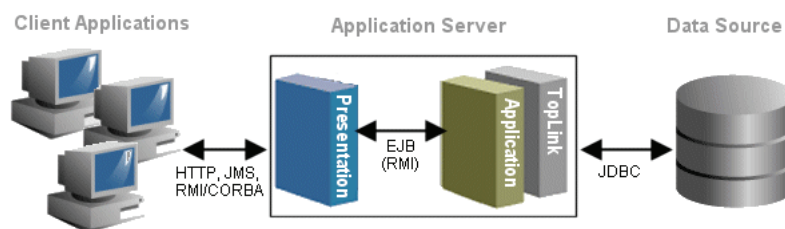
To minimize this problem, TopLink offers support for several data locking strategies. These include pessimistic locking and several variations of optimistic locking. For more information, see [Section 119.26, "Configuring Locking Policy"](#).

## 2.13 Considering EJB Session Bean Facade Architecture

This architecture is an extension of the three-tier pattern, with the addition of EJB session beans wrapping the access to the application tier. Session beans provide public API access to application operations, enabling you to separate the presentation tier from the application tier. The architecture also lets you use session beans within a Java EE container.

This type of architecture generally includes JTA integration, and serialization of data to the client.

**Figure 2-5 Three-Tier Architecture Using Session Beans and Java Objects**



A common extension to the three-tier architecture is to combine session beans and persistent Java objects managed by TopLink. The resulting application includes session beans and Java objects on a TopLink three-tier architecture (see [Figure 2-5](#)).

The three-tier architecture creates a server session and shares it between the session beans in the application. When a session bean needs to access a TopLink session, the bean obtains a client session from the shared server session. This architecture has the following key features:

- Session beans delimit transactions.
  - Configure TopLink to work with a JTA system and its associated connection pool.
- Accessing the persistent objects on the client side causes them to be serialized.
  - Ensure that when the objects re-emerge on the server-side, they properly merge into the cache to maintain identity.

### 2.13.1 Example Implementation

An example of the EJB session bean facade architecture implementation is a Model-View-Controller Model 2 architectural design pattern that runs in a Java EE container with servlets and JSP and uses the session bean enabled by TopLink to access data without EJB.

## 2.13.2 Advantages and Disadvantages

The EJB session bean facade architecture is a popular and effective compromise between the performance of persistent Java objects, and the benefits of EJB for standardized client development and server scalability. It offers the following advantages:

- Less overhead than an EJB CMP application  
TopLink shares access to the project, descriptor, and login information across the beans in the application.
- Future compatibility with other servers  
This design isolates login and EJB server-specific information from the beans, which lets you migrate the application from one application server to another without major recoding or rebuilding.
- Shared read cache  
This design offers increased efficiency by providing a shared cache for reading objects.

The key disadvantage of this model is the need to transport the persistent model to the client. If the model involves complex object graphs in conjunction with indirection (lazy loading), this can present many challenges with inheritance, indirection, and relationships.

For more information about managing inheritance, indirection and relationships, see [Part VIII, "Mappings"](#).

## 2.13.3 What Are Session Beans

Session beans model a process, operation, or service and as such, are not persistent entities. However, session beans can use persistence mechanisms to perform the services they model.

Under the session bean model, a client application invokes methods on a session bean that, in turn, performs operations on Java objects enabled by TopLink. Session beans execute all operations related to TopLink on behalf of the client.

The EJB specifications describe session beans as either stateless or stateful.

**Stateful beans** maintain a conversational state with a client; that is, they retain information between method calls issued by a particular client. This enables the client to use multiple method calls to manipulate persistent objects.

**Stateless beans** do not retain data between method calls. When the client interacts with stateless session beans, it must complete any object manipulations within a single method call.

## 2.13.4 Technical Challenges

Your application can use both stateful and stateless session beans with a TopLink client session or database session. When you use session beans with a TopLink session, the type of bean used affects how it interacts with the session.

- Stateless session beans and the TopLink session  
Stateless beans store no information between method calls from the client. As a result, reestablish the connection of the bean to the session for each client method call. Each method call through TopLink obtains a client session, makes the appropriate calls, and releases the reference to the client session.

- Stateful session beans and the TopLink session

Your EJB server configuration includes settings that affect the way it manages beans—settings designed to increase performance, limit memory footprint, or set a maximum number of beans. When you use stateful beans, the server may deactivate a stateful session bean enabled by TopLink out of the JVM memory space between calls to satisfy one of these settings. The server then reactivates the bean when required, and brings it back into memory.

This behavior is important, because a TopLink session instance does not survive passivation. To maintain the session between method calls, release the session during the passivation process and re-obtain it when you reactivate the bean.

- External JDBC pools

By default, TopLink manages its own connection pools. For the session bean architecture, you must configure TopLink to use connection pooling offered by the host application server. This feature is useful for shared connection pools and is required for JTA/JTS integration (see [Section 97.4, "Configuring External Connection Pooling"](#)).

- JTA/JTS integration

JTA and JTS are standard Java components that enable sessions to participate in distributed transactions. You must configure TopLink to use JTA/JTS to use session beans in the architecture (see [Section 115.13, "Integrating the Unit of Work with an External Transaction Service"](#)).

- Cache coordination

If you choose to use multiple servers to scale your application, you may require TopLink cache coordination (see [Section 102.3, "Cache Coordination"](#)).

### 2.13.5 What Is a Unit of Work Merge

You can use a unit of work to enable your client application to modify objects on the database. The unit of work merge functions employ mappings to copy the values from the serialized object into the unit of work, and to calculate changes.

For more information, see [Section 115.5, "Merging Changes in Working Copy Clones"](#).

## 2.14 Considering EJB Entity Beans with CMP Architecture

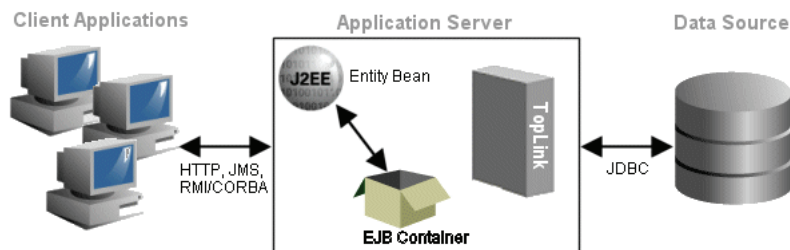
CMP is the part of the Java EE component model that provides an object persistence service that an EJB container uses to persist entity beans. CMP provides distributed, transactional, secure access to persistent data, with a guaranteed portable interface.

This architecture is an extension of the three-tier architecture, in which the implementation of persistence methods is handled by the container at runtime. As a bean provider, you only need to specify in a deployment descriptor those persistent fields and relationships for which the container must handle data access and, optionally, an abstract representation of the database schema.

TopLink CMP is an extension of the TopLink persistence framework that provides custom integration to EJB containers of various application servers (see [Section 8.1, "Introduction to the Application Server Support"](#)). For more information about choosing an application server, see [Section 2.2.2, "Target Platforms"](#). TopLink integrates with the EJB container in this architecture to augment the container's persistence manager.

TopLink CMP integration is nonintrusive (see [Figure 2–6](#)). Through a combination of run-time integration and code generation, the container uses TopLink internally and the bean user interacts with entity beans with container-managed persistence according to their standard API. This lets you combine the standard interfaces and power of CMP and a container with TopLink flexibility, performance, and productivity.

**Figure 2–6 Three-Tier CMP Architecture**



For more information, see the following:

- [Chapter 8, "Integrating TopLink with an Application Server"](#)
- [Chapter 9, "Creating TopLink Files for Deployment"](#)
- [Chapter 10, "Packaging a TopLink Application"](#)
- [Chapter 11, "Deploying a TopLink Application"](#)
- [Section 117.5, "Configuring Persistence Type"](#)

### 2.14.1 Example Implementation

An example of the entity beans with container-managed persistence implementation is a Model-View-Controller Model 2 architectural design pattern that runs in a Java EE container, with servlets and JSP that access either session beans or entity beans with container-managed persistence enhanced by TopLink.

### 2.14.2 Advantages and Disadvantages

A three-tier architecture using entity beans with container-managed persistence offers the following advantages:

- It allows for entity beans with container-managed persistence supplied with sophisticated TopLink features such as caching and mapping support, storing bean data across more than one table, composite primary keys, and data conversion.
- It presents a standard method to access data, which lets you create standardized, reusable business objects.
- It is well-suited to create coarse-grained objects, which TopLink relates to dependent, lightweight, regular Java objects (TopLink can also manage container-managed relationships to lightweight dependent Java objects).
- TopLink provides for lazy initialization of referenced objects and beans (see [Section 17.2.4, "Indirection \(Lazy Loading\)"](#)).
- TopLink provides functionality for transactional copies of beans, allowing concurrent access by several clients, rather than relying on individual serialization.

- TopLink provides advanced query capabilities, as well as dynamic querying, including the ability to define queries at the bean-level rather than the data source level and to use a rich set of querying and finder options.
- TopLink maintains bean and object identity.

The disadvantage of this architecture is that pure entity bean with container-managed persistence architectures can impose a high overhead cost. This is especially true when a data model has a large number of fine-grained classes with complex relationships.

### 2.14.3 Technical Challenges

The key technical challenge in this architecture lies in integrating components into a cohesive system. For example, this architecture requires a specific TopLink integration with the application server or Java EE container.

Other issues include the following:

- [External JDBC Pools](#)
- [JTA/JTS Integration](#)
- [Cache Coordination](#)
- [Maintaining Bidirectional Relationships](#)
- [Managing Dependent Objects](#)
- [Managing Collections of EJBObject Objects](#)

#### 2.14.3.1 External JDBC Pools

By default, TopLink manages its own connection pools. You can also configure TopLink to use connection pooling offered by the host application server. This feature is useful for shared connection pools and is required for JTA/JTS integration (see [Section 97.4, "Configuring External Connection Pooling"](#)).

#### 2.14.3.2 JTA/JTS Integration

JTA and JTS are standard Java components that enable sessions to participate in distributed transactions. You must configure TopLink to use JTA/JTS to use session beans in the architecture (see [Section 115.13, "Integrating the Unit of Work with an External Transaction Service"](#)).

#### 2.14.3.3 Cache Coordination

If you choose to use multiple servers to scale your application, you may require TopLink cache coordination (see [Section 102.3, "Cache Coordination"](#)).

#### 2.14.3.4 Maintaining Bidirectional Relationships

When one-to-one or many-to-many relationship is bidirectional, you must maintain the back pointers as the relationships change.

TopLink automatically maintains the relationship between two entity beans.

To set the back pointer manually, do one of the following:

- Code the entity bean to maintain the back pointer when the relationship is established or modified (recommended).
- Code the client to explicitly set the back pointer.

If you code the entity bean to set back pointers, the client is freed of this responsibility. This has the advantage of encapsulating this maintenance implementation in the bean.

In a one-to-many relationship, a source bean might have several dependent target objects. For example, an `EmployeeBean` might own several dependent `PhoneNumber` instances. When you add a new dependent object (a `PhoneNumber`, in this example) to an employee, you must set the `PhoneNumber` instance's back pointer to its owner (the employee). Maintaining a one-to-many relationship in the entity bean involves getting the local object reference from the context of the `EmployeeBean`, and then updating the back pointer, as [Example 2-3](#) shows.

### **Example 2-3 Setting the Back-Pointer in the Entity Bean**

```
// obtain owner and phoneNumber
owner = empHome.findByPrimaryKey(ownerId);
phoneNumber = new PhoneNumber("cell", "613", "5551212");
// add phoneNumber to the phoneNumbers of the owner
owner.addPhoneNumber(phoneNumber);

// Maintain the relationship in the Employee's addPhoneNumber method
public void addPhoneNumber(PhoneNumber newPhoneNumber) {
    // get, then set the back pointer to the owner
    Employee owner = (Employee)this.getEntityContext().getEJBLocalObject();
    newPhoneNumber.setOwner(owner);
    // add new phone
    getPhoneNumbers().add(newPhoneNumber);
}
```

For more information, see the following:

- [Section 121.18, "Configuring Bidirectional Relationship"](#)
- [Section 27.2.1, "Directionality"](#)

### **2.14.3.5 Managing Dependent Objects**

Unlike EJB, `TopLink` dependent persistent objects can be sent back and forth between a client and a server. When objects are serialized, the risk exists the objects can cause the cache to lose the identity of the objects or attempt to cache duplicate identical objects. To avoid potential problems, use the bean setter methods when adding dependent objects to relationship collections, as [Example 2-4](#) shows. This enables `TopLink` to handle merging of objects in the cache.

### **Example 2-4 Managing Dependent Objects**

```
addPhoneNumber(PhoneNumber phone) {
    Collection phones = this.getPhoneNumbers();
    Vector newCollection = new Vector();
    newCollection.addAll(phones);
    newCollection.add(phone);
    this.setPhones(newCollection);
}
```

### **2.14.3.6 Managing Collections of EJBObject Objects**

Collections generally use the `equals` method to compare objects. This is not a problem in the case of an object that contains a collection of `EJBObject` objects, because the EJB container collection handles equality appropriately.



## 2.15 Considering EJB Entity Beans with BMP Architecture

BMP is the part of the Java EE component model that lets you, the bean provider, implement the entity bean's persistence directly in the entity bean class or in one or more helper classes that you provide.

This architecture is an extension of the three-tier architecture, in which the persistent data is bean managed within an entity bean using code that you implement. The client code accesses the data through the entity bean interface.

TopLink BMP is an extension of the TopLink persistence framework that provides base class `BMPEntityBase` as a starting point for your BMP development. This class provides an implementation for all methods (except `ejbPassivate`) required by the EJB specifications prior to 3.0. Subclass `BMPEntityBase` to create a TopLink-enabled entity bean with bean-managed persistence.

To use the `BMPEntityBase` class, perform the following:

1. Create a TopLink session (see [Chapter 87, "Introduction to TopLink Sessions"](#)) for your application.
2. Add a `BMPWrapperPolicy` to each descriptor that represents an entity bean with bean-managed persistence.

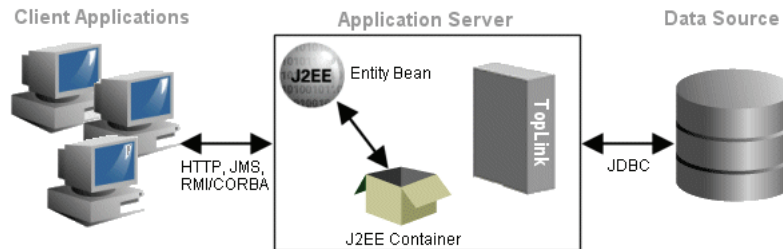
The `BMPWrapperPolicy` provides TopLink with the information to create remote objects for entity beans and to extract the data out of a remote object.

3. Create the home and remote interfaces.
4. Create deployment descriptors (see [Chapter 8, "Integrating TopLink with an Application Server"](#) and [Chapter 9, "Creating TopLink Files for Deployment"](#)).
5. Package your application (see [Chapter 10, "Packaging a TopLink Application"](#)).
6. Deploy the beans (see [Chapter 11, "Deploying a TopLink Application"](#)).

To make full use of TopLink session and unit of work features, TopLink provides a hook into its functionality through the `BMPDataStore` class. Use this class to translate EJB-required functionality into simple calls.

The `BMPDataStore` class provides implementations of `LOAD` and `STORE`, multiple finders, and `REMOVE` functionality. The `BMPDataStore` class requires a TopLink session. A single instance of `BMPDataStore` must exist for each bean type deployed within a session. When creating a `BMPDataStore`, pass in the session name of the session that the `BMPDataStore` must use to persist the beans and the class of the bean type being persisted. Store the `BMPDataStore` in a global location so that each instance of a bean type uses the correct `store` method.

TopLink BMP support (see [Figure 2-7](#)) lets you combine the standard interfaces of entity beans with bean-managed persistence with TopLink flexibility, performance, and productivity.

**Figure 2-7 Three-Tier BMP Architecture**

TopLink supports BMP. To use BMP support, the home interface must inherit from the `oracle.toplink.ejb.EJB20Home`. To make calls to the `BMPEntityBase`, the `findAll` method must call the EJB 2.0 version of the methods. These methods are prefixed with `ejb20`. For example, in the EJB 2.0 version, the `findAll` method appears as `ejb20findAll`.

To use local beans, use the `oracle.toplink.ejb.EJB20LocalHome` setting instead of the default `oracle.toplink.ejb.EJB20Home`. Instead of the `oracle.toplink.ejb.BMPWrapperPolicy` setting, use the `oracle.toplink.ejb.bmp.BMPLocalWrapperPolicy` setting.

To accommodate both local and remote configurations, ensure the following:

- For a bean that has a single interface, use the corresponding wrapper policy (local or remote) for the descriptor.
- Beans can only participate in relationships as either local or remote interfaces, not both.

For more information, see the following:

- [Chapter 8, "Integrating TopLink with an Application Server"](#)
- [Chapter 9, "Creating TopLink Files for Deployment"](#)
- [Chapter 10, "Packaging a TopLink Application"](#)
- [Chapter 11, "Deploying a TopLink Application"](#)
- [Section 117.5, "Configuring Persistence Type"](#)
- [Section 89.9, "Configuring the Server Platform"](#)

### 2.15.1 Example Implementations

An example of the entity beans with bean-managed persistence implementation is a Model-View-Controller Model 2 architectural design pattern that runs in a Java EE container, with servlets and JSP that access session beans and entity beans with bean-managed persistence enhanced by TopLink.

### 2.15.2 Advantages and Disadvantages

Using BMP with a TopLink three-tier architecture offers the following advantages:

- It simplifies the BMP method calls. These can be inherited from an abstract bean class, rather than being generated.
- TopLink makes BMP easier to implement.
- It enables you to implement database-independent code in the bean methods.

- The architecture supports features such as complex relationships, caching, object-level and dynamic queries, and the unit of work.

The main disadvantages of BMP include the following:

- You must create the persistence mechanisms in the bean code.
- It is not as transparent or efficient as CMP.
- TopLink-only Java object applications offer the same degree of independence from the application server.

### 2.15.3 Technical Challenges

The key technical challenge in this architecture lies in integrating components into a cohesive system. For example, this architecture requires a specific TopLink integration with the application server or Java EE container.

Other issues include the following:

- [External JDBC Pools](#)
- [JTA/JTS Integration](#)
- [Cache Coordination](#)

#### 2.15.3.1 External JDBC Pools

By default, TopLink manages its own connection pools. You can also configure TopLink to use connection pooling offered by the host application server. This feature is useful for shared connection pools and is required for JTA/JTS integration (see [Section 97.4, "Configuring External Connection Pooling"](#)).

#### 2.15.3.2 JTA/JTS Integration

JTA and JTS are standard Java components that enable sessions to participate in distributed transactions. You must configure TopLink to use JTA/JTS to use session beans in the architecture (see [Section 115.13, "Integrating the Unit of Work with an External Transaction Service"](#)).

#### 2.15.3.3 Cache Coordination

If you choose to use multiple servers to scale your application, you may require TopLink cache coordination (see [Section 102.3, "Cache Coordination"](#)).

## 2.16 Considering JPA Entity Architecture

A part of the EJB 3.0 specification, the Java Persistence API (JPA) is a lightweight, POJO-based framework for Java persistence. JPA focuses on object relational mapping and contains a full object relational mapping specification supporting the use of Java language metadata annotations and/or XML descriptors to define the mapping between Java objects and a relational database. Object relational mapping with the JPA is completely metadata-driven. JPA supports a SQL-like query language for both static and dynamic queries. It also supports the use of pluggable persistence providers.

JPA includes the following concepts:

- Entity—any application-defined object with the following characteristics can be an entity:
  - it can be made persistent;

- it has a persistent identity (a key that uniquely identifies an entity instance and distinguishes it from other instances of the same entity type. An entity has a persistent identity when there is a representation of it in a data store);
- it is partially transactional in a sense that a persistence view of an entity is transactional (an entity is created, updated and deleted within a transaction, and a transaction is required for the changes to be committed in the database). However, in-memory entities can be changed without the changes being persisted.
- it is not a primitive, a primitive wrapper, or built-in object. An entity is a fine-grained object that has a set of aggregated state that is typically stored in a single place (such as a row in a table), and have relationships to other entities.
- Entity metadata—describes every entity. Metadata could be expressed as annotations (specifically defined types that may be attached to or place in front of Java programming elements) or XML (descriptors).
- Entity manager—enables API calls to perform operations on an entity. Until an entity manager is used to create, read, or write an entity, the entity is just a regular nonpersistent Java object. When an entity manager obtains a reference to an entity, that entity becomes managed by the entity manager. The set of managed entity instances within an entity manager at any given time is called its persistence context—only one Java instance with the same persistent identity may exist in a persistence context at any time.

You can configure an entity manager to be able to persist or manage certain types of objects, read or write to a particular database, and be implemented by a specific persistence provider. The persistence provider supplies the backing implementation engine for JPA, including the `EntityManager` interface implementation, the `Query` implementation, and the SQL generation.

Entity managers are provided by an `EntityManagerFactory`. The configuration for an entity manager is bound to the `EntityManagerFactory`, but it is defined separately as a persistence unit. You name persistence units to allow differentiation between `EntityManagerFactory` objects. This way your application obtains control over which configuration to use for operations on a specific entity. The configuration that describes the persistence unit is defined in a `persistence.xml` file.

The following description expresses relationships between JPA concepts:

- Persistence creates one or more `EntityManagerFactory` objects;
- each `EntityManagerFactory` is configured by one persistence unit;
- `EntityManagerFactory` creates one or more `EntityManager` objects;
- one or more `EntityManager` manages one `PersistenceContext`.

TopLink implementation of JPA is provided by EclipseLink.

For more information, see the following:

- JPA sections of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/EclipseLink/UserGuide/Developing\\_JPA\\_Projects\\_%28ELUG%29](http://wiki.eclipse.org/EclipseLink/UserGuide/Developing_JPA_Projects_%28ELUG%29)
- *EclipseLink API* at <http://www.eclipse.org/eclipselink/api/1.0/index.html>

## 2.16.1 Example Implementations

An example of the entity beans with bean-managed persistence implementation is a Model-View-Controller Model 2 architectural design pattern that runs in a Java EE container, with servlets and JSP that access session beans and EJB 3.0-compliant entities using the EclipseLink JPA persistence provider.

## 2.16.2 Advantages and Disadvantages

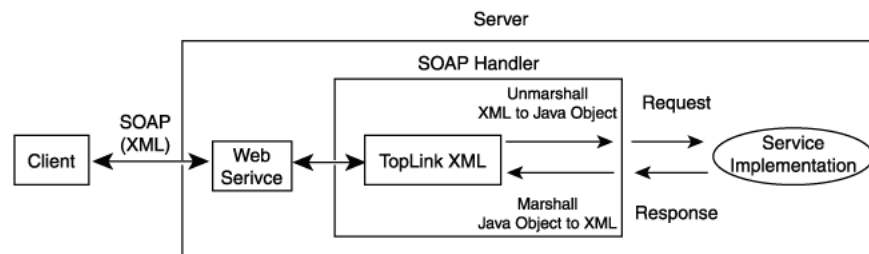
The use of EclipseLink JPA entities offers the following advantages:

- POJO persistence—in JPA, persistent objects are POJOs.
- Object relational mapping is completely metadata-driven.
- The persistence API exists as a separate layer from the persistent objects and does not intrude upon them.
- Using the query framework you can query across entities and their relationships without having to use concrete foreign keys or database columns. Also, you can define queries statically in metadata or create them dynamically by passing query criteria on construction. Queries can return entities as results.
- Entities are mobile: objects are able to move from one JVM to another and back, and at the same time be usable by the application.
- You can configure persistence features through the use of Java SE 5 annotations, or XML, or a combination of both. You may also rely on defaults.
- If your application is running inside a container, the container provides support and ease of use; you can configure the same application to run outside a container.

## 2.17 Considering Web Services Architecture

A Web services architecture is similar to the three-tier (see [Section 2.11, "Considering Three-Tier Architecture"](#)) or session bean (see [Section 2.13, "Considering EJB Session Bean Facade Architecture"](#)) architecture, however, in a Web services architecture, you encapsulate business logic (the service) in a Web service instead of (or in addition to) using session beans. In a Web services architecture, clients communicate with your application using SOAP messages (XML over HTTP).

**Figure 2–8 Web Services Architecture**



As in any architecture, you can use TopLink to persist objects to relational or EIS data sources. However, in a Web services architecture, you can also use TopLink to map your object model to an XML schema for use with the Web service or as the Web service XML serializer.

### 2.17.1 Example Implementations

An example of a Web services architecture implementation is the use of a Web service to expose parts of an existing application to a remote client (typically another application) by way of SOAP messages. In this application, you can use TopLink XML to unmarshall XML messages to Java objects to facilitate requests and marshall Java object responses back into XML for transmission to the client.

### 2.17.2 Advantages and Disadvantages

Using TopLink in Web services architecture has many advantages, including, but not limited to, the following:

- you can map XML messages to an existing Java object model;
- you can achieve a high level of complexity of mapping support;
- compliance with the JAXB standards;
- providing a scalable, high-performing solution.

One debatable disadvantage is this solution's complexity over a simple RMI session bean service.

### 2.17.3 Technical Challenges

As with any technology, there are technical challenges associated with the use of TopLink in Web services architecture. These technical challenges are mostly related to special-case scenarios, such as when you need to implement a custom serializer because you have both the Java objects and the schema.

For more information, see the following:

- *Oracle TopLink as a Custom Serializer in a JAX-RPC 1.1 Web service* at <http://www.oracle.com/technology/products/ias/toplink/technical/tips/jaxRpc11/index.htm>
- Part XVI, "XML Mappings"

## 2.18 Considering EclipseLink Service Data Objects (SDO) Architecture

An EclipseLink SDO architecture uses the SDO 2.1 framework for data application and development. For more information, see "Considering EclipseLink Service Data Objects (SDO) Architecture" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Introduction\\_to\\_EclipseLink\\_Application\\_Development\\_%28ELUG%29#Considering\\_EclipseLink\\_Service\\_Data\\_Objects\\_.28SDO.29\\_Architecture](http://wiki.eclipse.org/Introduction_to_EclipseLink_Application_Development_%28ELUG%29#Considering_EclipseLink_Service_Data_Objects_.28SDO.29_Architecture).

# Part II

---

## TopLink Development Tools Overview

This part describes the development tools and tool support TopLink provides. It contains the following chapters:

- [Chapter 3, "Introduction to TopLink Development Tools"](#)  
This chapter describes the development tools and tool support TopLink provides.
- [Chapter 4, "Using Oracle JDeveloper TopLink Editor"](#)  
This chapter describes how to use Oracle JDeveloper TopLink editor.
- [Chapter 5, "Using TopLink Workbench"](#)  
This chapter describes how to use TopLink Workbench including working with databases, generating data from database tables, and creating and editing a `sessions.xml` file.
- [Chapter 6, "Using the Schema Manager"](#)  
This chapter explains how to use the TopLink schema manager to create databases, tables, stored procedures, and to populate database tables.
- [Chapter 7, "Using an Integrated Development Environment"](#)  
This chapter explains how to integrate TopLink with an IDE.





## Introduction to TopLink Development Tools

The TopLink runtime provides Java or Java EE applications with access to persistent entities stored in a data source. In addition to run-time capabilities, the TopLink Foundation Library includes the TopLink Application Programming Interface (API). This API enables applications to access TopLink run-time features.

TopLink includes additional development tools that simplify application development. These tools capture mapping and run-time configuration information in metadata files that TopLink passes to the application at run time.

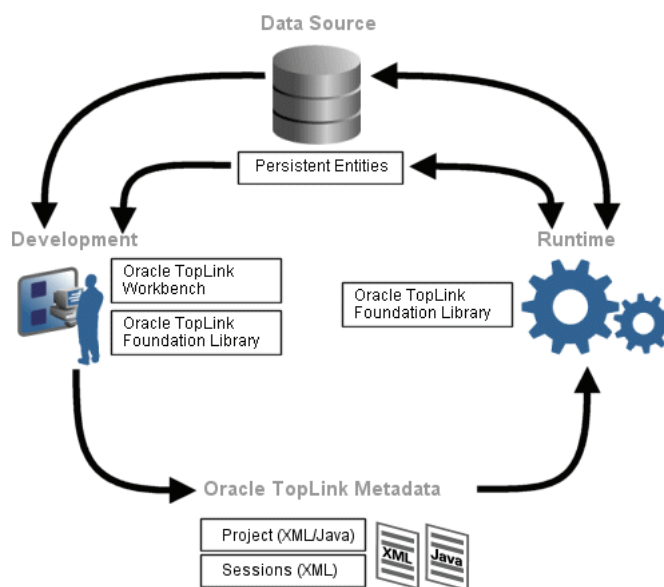
This chapter includes the following sections:

- [Development Environment](#)
- [TopLink Run-Time Environment](#)

TopLink metadata is the link between the two (see [Section 2.9, "Working with TopLink Metadata"](#)).

Figure 3-1 illustrates how these elements interact with the data source.

**Figure 3-1 TopLink Components in Development Lifecycle**

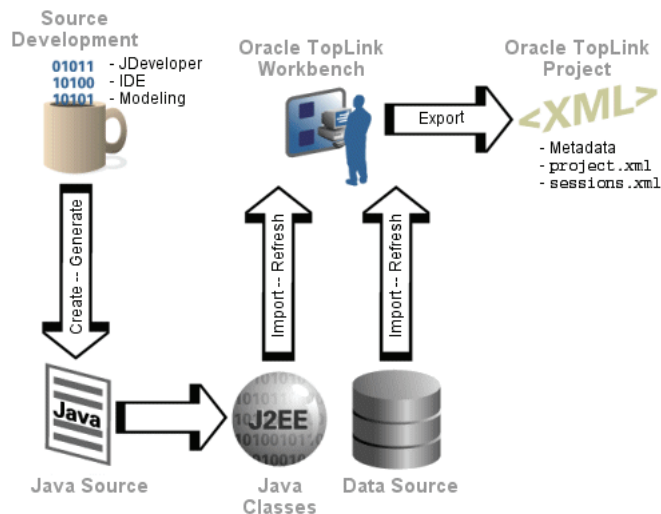


## 3.1 Development Environment

To create a TopLink application, use Oracle JDeveloper TopLink Editor or TopLink Workbench to map objects to data sources using relational and nonrelational models. Capture the resulting mappings and additional run-time configurations in the TopLink project file (`project.xml`) and build a session configuration file (`sessions.xml`). These files together represent your entire TopLink project, as shown in [Figure 3–2](#).

During development, you can use the TopLink API to define query and transaction logic. When you use entity beans, there is generally little or no direct use of the TopLink API and there is no session or `sessions.xml` file.

**Figure 3–2 TopLink Workbench in Development Environment**



TopLink Workbench can import compiled entity classes (Java objects or EJB entity beans), as well as relational or nonrelational schemas through a JDBC driver (configured by you). Because TopLink imports the object and relational models for mapping, you can develop the two models relatively independently from the mapping phase of a project development.

## 3.2 TopLink Run-Time Environment

The TopLink Foundation Library provides the TopLink run-time component. Access the run-time component either directly through the TopLink API or indirectly through a Java EE container when using entity beans with container-managed persistence. The run-time environment is not a separate or external process—it is embedded within the application. Application calls `invoke TopLink` to provide persistence behavior. This function allows for transactional and thread-safe access to shared database connections and cached objects.

In addition to Java EE environments, TopLink fully supports non-Java EE environments as well. See [Section 2.3, "Selecting an Architecture with TopLink"](#) for more information.

---

## Using Oracle JDeveloper TopLink Editor

This chapter provides general information about Oracle JDeveloper TopLink Editor, as well as detailed information on using and customizing it.

This chapter includes the following sections:

- [Introduction to Oracle JDeveloper TopLink Editor](#)
- [Configuring the Oracle JDeveloper TopLink Editor](#)
- [Using the Oracle JDeveloper TopLink Editor](#)

### 4.1 Introduction to Oracle JDeveloper TopLink Editor

Using Oracle JDeveloper's TopLink editor, you can quickly and easily configure and map your Java classes, EJB, JPA entities to different data sources, including relational databases, enterprise information systems (EIS), and XML schemas *without using code*. The TopLink editor supports multiple standards, including JPA and EJB 3.0.

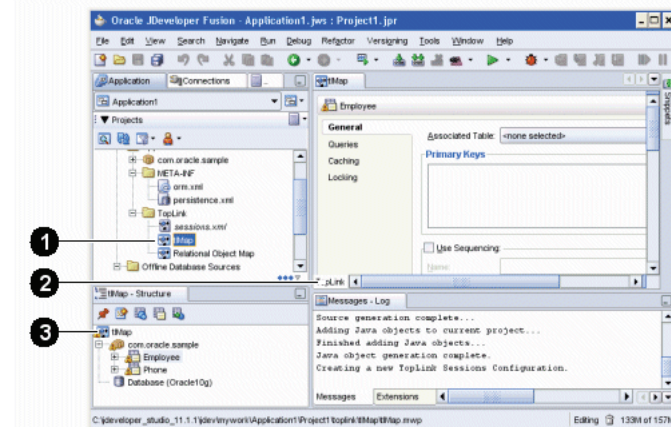
For more information, see the "Developing TopLink Mappings" section of the Oracle JDeveloper online help.

### 4.2 Configuring the Oracle JDeveloper TopLink Editor

For more information, see "Configuring TopLink Preferences" in the "Developing TopLink Mappings" section of the Oracle JDeveloper online help.

### 4.3 Using the Oracle JDeveloper TopLink Editor

[Figure 4–1](#) shows the primary parts of the TopLink editor:

**Figure 4–1 Parts of the TopLink Editor**

1. TopLink Project Elements in the Application Navigator
2. TopLink Editor Tabs in the Editor Window
3. TopLink Project Elements in the Structure Window

For more information, see "About the TopLink Editor" in the "Developing TopLink Mappings" section of the Oracle JDeveloper online help.

### 4.3.1 TopLink Project Elements in the Application Navigator

The Application Navigator displays each element of with your TopLink mappings including the mapping project, deployment descriptor, and sessions information.

Figure 4–2 shows sample TopLink project elements in the Application Navigator.

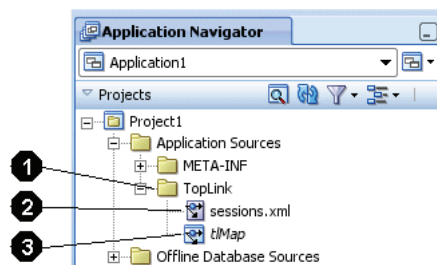
**Figure 4–2 Sample TopLink Editor – Application Navigator**

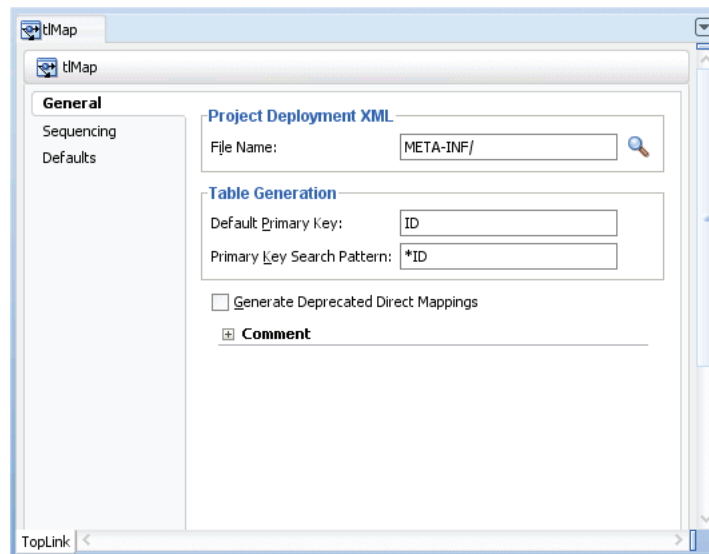
Figure 4–2 identifies the following user interface components:

1. TopLink folder
2. Sessions configuration file (`sessions.xml`)
3. TopLink map

### 4.3.2 TopLink Editor Tabs in the Editor Window

The TopLink Editor displays your TopLink mapping information. The information in the editor will vary, depending on the TopLink element you selected in the Application Navigator or Structure window.

Figure 4–3 shows the TopLink Editor tabs for a TopLink map.

**Figure 4–3 Sample TopLink Editor Tabs**

### 4.3.3 TopLink Project Elements in the Structure Window

The Structure window displays detailed information about the TopLink element selected in Application Navigator or TopLink Editor.

- When working with an EJB or Java class, the Structure window displays the related TopLink descriptor and its mapping attributes.
- When working with a TopLink sessions configuration file, the Structure window displays your sessions and session brokers.
- When working with a persistence configuration, the Structure window displays your JPA descriptors and persistence units.

When you select an item in the Structure window, its properties appear in the TopLink Editor.

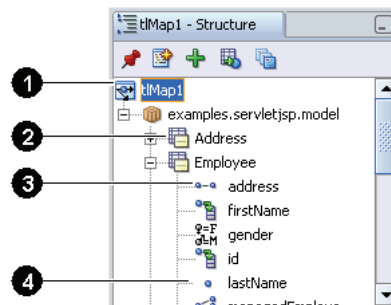
**Figure 4–4 Sample TopLink Editor – Structure Window**

Figure 4–4 identifies the following user interface components:

1. TopLink map
2. Descriptor
3. Mapped Java attribute (one-to-one mapping)
4. Unmapped attribute

You can perform specific functions for an item by selecting the item in the Navigator and doing the following:

- Right-clicking the object and selecting the function from the pop-up menu.
- Selecting the object by clicking a button in the Structure toolbar.

---

---

## Using TopLink Workbench

This chapter provides information about understanding, using, and customizing TopLink Workbench.

This chapter includes the following sections:

- [Introduction to TopLink Workbench](#)
- [Configuring the TopLink Workbench Environment](#)
- [Using TopLink Workbench](#)
- [Using TopLink Workbench Preferences](#)
- [Using Databases](#)
- [Using XML Schemas](#)
- [Using Classes](#)
- [Integrating TopLink Workbench with Apache Ant](#)

For information on using TopLink Workbench to configure sessions XML, refer to Part XXI, "TopLink Sessions".

### 5.1 Introduction to TopLink Workbench

TopLink Workbench is a separate component from the TopLink runtime—it lets you graphically configure descriptors and map your project. TopLink Workbench can verify the descriptor options, access the data source (either a database or an XML schema), and create the database schema. Using TopLink Workbench, you can define TopLink descriptors and configurations *without using code*.

You can use TopLink Workbench during the *development* phase of the development process (see [Section 2.1, "Introduction to TopLink Application Development"](#)). Typically, this phase includes the following:

1. Defining an object model (a set of Java classes) to describe and solve your problem.
2. Creating a TopLink Workbench project, importing your Java classes and data sources, and using descriptors to describe how the Java classes map to your data source model.
3. Creating a TopLink session and registering your descriptors. In your application, use the session to retrieve and store objects from and to the data source.

TopLink Workbench creates a `<projectName>.mwp` file to store all TopLink project information, including object model, descriptor, and session information.

The `<projectName>.mwp` file is used only by TopLink Workbench. Typically, the only time you need to modify the `<projectName>.mwp` file is to merge changes during application development by a team of developers (Section 7.2.2, "How to Merge Files").

Using TopLink Workbench, you export this information into a `project.xml` file that your TopLink enabled application reads at run time.

For more information on using TopLink Workbench as the development environment, see Figure 3-2.

## 5.2 Configuring the TopLink Workbench Environment

TopLink Workbench reads its environment variables from the `setenv` script in the `<TOPLINK_HOME>/bin` directory.

Before you launch TopLink Workbench, you must configure its environment as follows:

1. Use a text editor to open the `<TOPLINK_HOME>/bin/setenv` script.
  - For Windows, open the `setenv.cmd` file.
  - For UNIX, open the `setenv.sh` file.
2. Ensure that the `JAVA_HOME` environment variable is set:
  - For Windows: `set JAVA_HOME=C:/j2sdk1.5.0_04`
  - For UNIX: `JAVA_HOME=/usr/local/packages/java; export JAVA_HOME`
3. Update the `DRIVER_CLASSPATH` environment variable to add the location of the following (if necessary):

---

---

**Note:** Do not include any Java classes for your persistent business objects in the `DRIVER_CLASSPATH` variable. Instead, add these persistent business objects in your TopLink Workbench project classpath (see Section 117.3, "Configuring Project Classpath").

---

---

- JDBC drivers—if you are using relational projects (see Section 18.1, "Building Relational Projects").
- Java EE Connector Architecture (JCA) adapters—if you are using EIS projects (see Section 71.1, "EIS Project Concepts").
- JCA `connector.jar` file—if you are using EIS projects (see Section 71.1, "EIS Project Concepts").

The `connector.jar` file contains `javax.resource.cci` and `javax.resource.spi` interfaces that TopLink EIS uses. By default, TopLink Workbench updates its classpath to include the Java 1.5.*n* `connector.jar` file from `<ORACLE_HOME>/lib/java/api`. If this version of the `connector.jar` file is incompatible with your environment, edit the `workbench.cmd` or `workbench.sh` file in `<TOPLINK_HOME>/utils/workbench` to change the path to this file.

At run time, this `connector.jar` file (or its equivalent) must be on your application or application server classpath.



- Oracle Database ORACLE\_HOME/rdbms/jlib/xdm.jar file—if you are using direct-to-XMLType mappings with an Oracle9i Database or later (see Section 27.4, "Direct-to-XMLType Mapping").
- Custom Collection class that you use to override the default Collection class that TopLink uses with a mapping container policy (see Section 121.14, "Configuring Container Policy").

Example 5–1 shows how to set the DRIVER\_CLASSPATH variable for Windows system, and Example 5–2—for UNIX.

#### Example 5–1 Setting DRIVER\_CLASSPATH on Windows

```
set DRIVER_
CLASSPATH=C:\OraHome2\jdbc\lib\ojdbc14.jar;C:\Attunity\Connect\Java\lib\attunityResourceAdapt
er.jar;C:\OraHome2\rdbms\jlib\xdm.jar
```

---

**Note:** If the path to your driver(s) contains spaces, you must enclose the path in double-quotes in the setenv.cmd file. For example:

```
set DRIVER_CLASSPATH="C:\Program Files\some directory\driver.jar"
```

---

#### Example 5–2 Setting DRIVER\_CLASSPATH on UNIX

```
DRIVER_
CLASSPATH=/OraHome2/jdbc/lib/ojdbc14.jar;/attunity/connect/java/lib/attunityResourceAdapter.j
ar;/OraHome2/rdbms/jlib/xdm.jar; export JDBC_CLASSPATH
```

4. Save and close the setenv script.

To launch TopLink Workbench, double-click the workbench.cmd file located in <TOPLINK\_HOME>/utils/workbench directory.

## 5.2.1 How to Configure the Language Preference

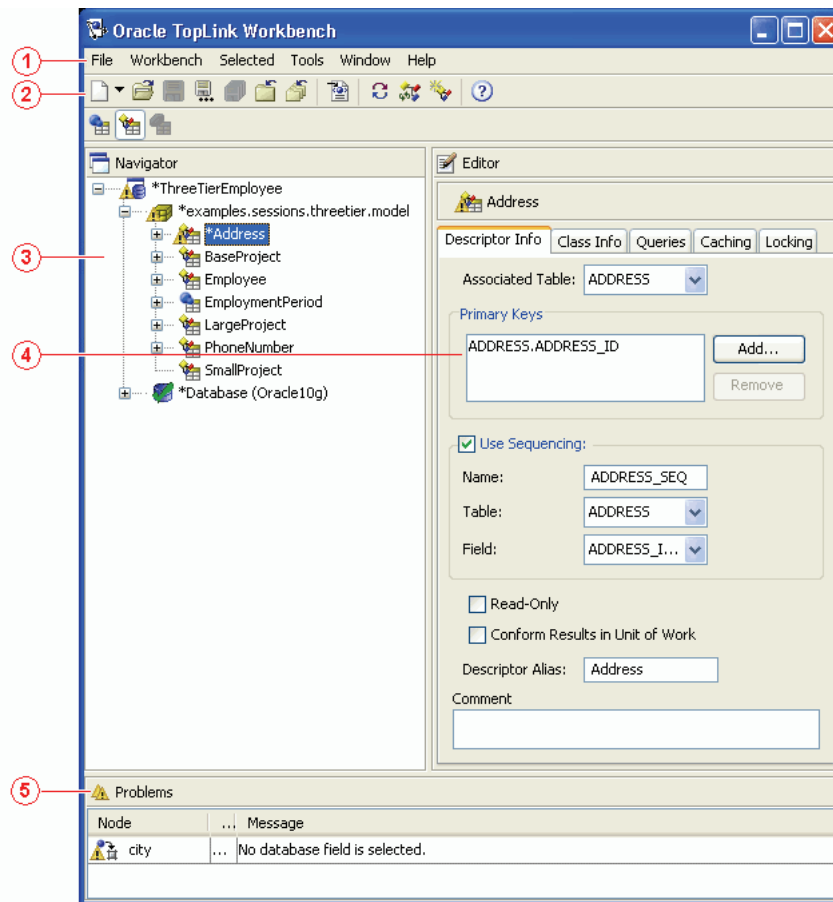
To use TopLink Workbench in a language different than your default, add the -Duser.language and -Duser.country options to the JVM\_ARGS variable in the workbench.cmd or .sh file. For example, the following arguments will start TopLink Workbench in US English, regardless of default language of your operating system:

```
JVM_ARGS="-Duser.language=en -Duser.country=en_US"
```

## 5.3 Using TopLink Workbench

Figure 5–1 shows the primary parts of TopLink Workbench window.

Figure 5–1 TopLink Workbench Window



The numbered callouts in [Figure 5–1](#) identify the following user interface components:

1. Menu bar

The menu bar contains menus for each TopLink Workbench function. Some objects also contain context-sensitive menus. See [Section 5.3.1, "How to Use Menus"](#) for more information.

2. Toolbars

The toolbars contain shortcuts to specific functions. See [Section 5.3.2, "How to Use Toolbars"](#) for more information.

3. Navigator window section

The Navigator window section shows the project navigation tree for all open projects (see [Section 5.3.3, "How to Use the Navigator"](#)). Click the plus (+) or minus (–) sign next to an object (or double-click the object) to expand or collapse the tree. When you select an object in the **Navigator** window section, its properties appear in the **Editor** window.

4. Editor window section

The **Editor** window section contains specific property sheets and option tabs for the currently selected object. See [Section 5.3.4, "How to Use the Editor"](#) for more information.

5. Problems window section

The **Problems** window section shows messages and errors for the currently selected object in the Navigator window section (see Section 5.3.5, "How to Use the Problems Window"). Section A.3, "TopLink Workbench Error Reference" contains detailed information on each error message.

## 5.3.1 How to Use Menus

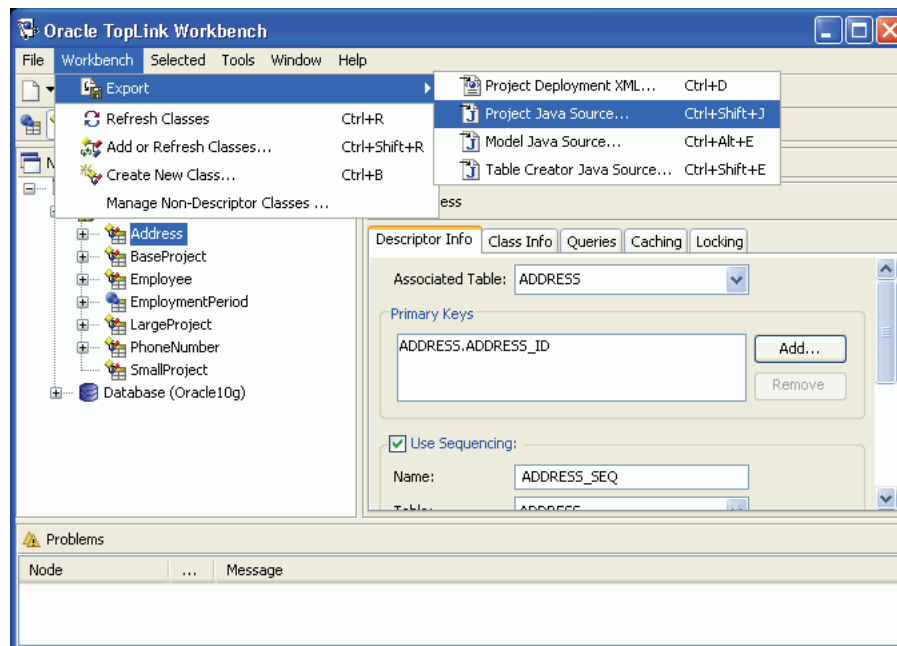
TopLink Workbench contains the following two types of menus:

- Menu bar menus (see Section 5.3.1.1, "Using Menu Bar Menus")
- Context menus (see Section 5.3.1.2, "Using Context Menus")

### 5.3.1.1 Using Menu Bar Menus

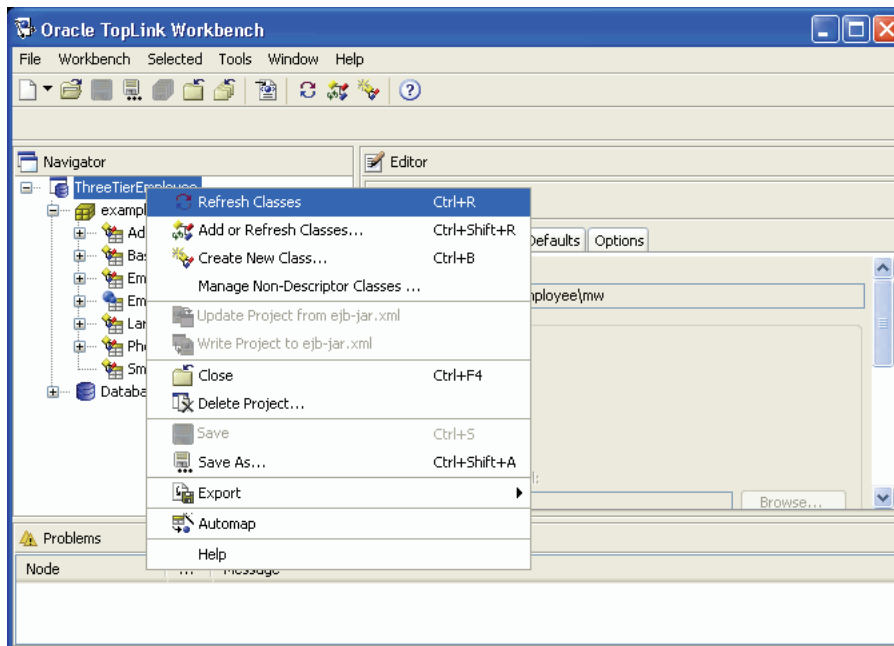
The menu bar, located at the top of the TopLink Workbench window, provides menus for each TopLink Workbench function. Some menus (such as **Selected**) are context-sensitive; the available options may vary, depending on the currently selected object.

**Figure 5–2 Sample Menu Bar Menu**



### 5.3.1.2 Using Context Menus

When you right-click objects in the **Navigator** window, a context menu appears with functions specific to the selected object.

**Figure 5–3 Sample Context Menu**

## 5.3.2 How to Use Toolbars

TopLink Workbench contains the following toolbars at the top of the window:









- Standard toolbar (see [Section 5.3.2.1, "Using Standard Toolbar"](#))
- Context toolbar (see [Section 5.3.2.2, "Using Context Toolbar"](#))

Toolbars provide tool tips: each toolbar button provides a brief description when you position the mouse pointer over it.





### 5.3.2.1 Using Standard Toolbar

The standard toolbar furnishes quick access to the standard menu options (**F**ile, **E**dit, **S**electe**d**, and so on).

**Table 5–1 Standard Toolbar Buttons**

Button	Description	Available for...
	New	All
	Open	All
	Save	All
	Save as	All
	Save all	All
	Close	All
	Close all	All
	Help topics	All

**Table 5–1 (Cont.) Standard Toolbar Buttons**



















Button	Description	Available for...
	Export deployment XML for the selected projects	Projects
	Refreshes selected classes	Projects
	Add or refresh classes	Projects
	Create new class	Projects

### 5.3.2.2 Using Context Toolbar





























The context toolbar provides quick access to functions for the currently selected object in the **Navigator** (see [Section 5.3.3, "How to Use the Navigator"](#)). The available buttons will vary, depending on which item you have selected.

You can also right-click the item and choose the appropriate option from the context menu.



**Table 5–2 Context Toolbar Buttons**

Button	Description	Available for...
	Login to database	Databases
	Logout of database	Databases
	Add new table	Databases
	Add or update existing tables from database	Databases
	Refresh from database	Database tables
	Remove table or selected item	Database tables
	Rename table or selected item	Database tables
	Import schema	Schemas
	Relational aggregate descriptor	Descriptors
	Relational class descriptor	Descriptors
	Relational EJB descriptor	Descriptors
	EIS composite descriptor	Descriptors
	EIS root descriptor	Descriptors
	EIS EJB descriptor	Descriptors
	XML descriptor	Descriptors
	Direct-to-field mapping	Attributes in relational descriptors
	Object type mapping <sup>1</sup>	Attributes in relational descriptors
	Type conversion mapping <sup>1</sup>	Attributes in relational descriptors

**Table 5–2 (Cont.) Context Toolbar Buttons**

<b>Button</b>	<b>Description</b>	<b>Available for...</b>
	Serialized object mapping <sup>1</sup>	Attributes in relational descriptors
	Direct-to-XMLType mapping	Attributes in relational descriptors
	Direct collection mapping	Attributes in relational descriptors
	Direct map mapping	Attributes in relational descriptors
	Aggregate mapping	Attributes in relational descriptors
	One-to-one mapping	Attributes in relational descriptors
	Variable one-to-one mapping	Attributes in relational descriptors
	One-to-many mapping	Attributes in relational descriptors
	Many-to-many mapping	Attributes in relational descriptors
	Direct mapping	Attributes in EIS descriptors
	Direct collection mapping	Attributes in EIS descriptors
	Composite object mapping	Attributes in EIS descriptors
	Composite collection mapping	Attributes in EIS descriptors
	One-to-one mapping	Attributes in EIS descriptors
	One-to-many mapping	Attributes in EIS descriptors
	Direct-to-XML mapping	Attributes in XML descriptors
	Direct collection mapping	Attributes in XML descriptors
	Composite object mapping	Attributes in XML descriptors
	Composite collection mapping	Attributes in XML descriptors
	Any object mapping	Attributes in XML descriptors
	Any collection mapping	Attributes in XML descriptors
	Transformation mapping	Attributes in all descriptors
	Unmap	Attributes in all descriptors
	Session	Sessions configurations
	Session Broker	Sessions configurations
	Named connection pool	Server sessions
	Sequence connection pool	Server sessions
	Write connection pool	Server sessions

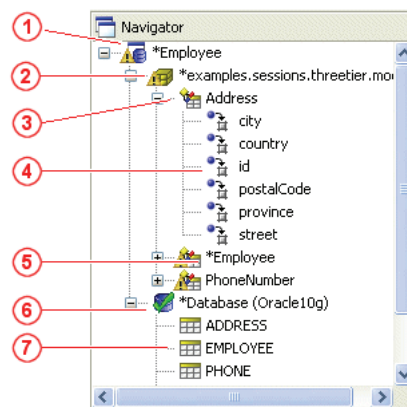
**Table 5–2 (Cont.) Context Toolbar Buttons**

Button	Description	Available for...
	Rename	Database sessions, session brokers
	Delete session	Database sessions, session brokers

<sup>1</sup> Deprecated. For more information, see [Section 27.2.2.2, "Using a Converter Mapping"](#)

### 5.3.3 How to Use the Navigator

TopLink displays the items included in each project (descriptors, mappings, data source, and so on) in the **Navigator** on the left side of the TopLink Workbench window, as [Figure 5–4](#) shows.

**Figure 5–4 Sample Navigator**

The numbered callouts on [Figure 5–4](#) identify the following user interface components:

1. Project (relational project)
2. Package
3. TopLink Descriptor (relational descriptor)
4. Attribute/mapping (direct to field mapping)
5. Unsaved/changed item
6. Database
7. Database table

Click the plus (+) or minus (–) sign next to the item, or double-click the item name to expand or collapse the item.

TopLink Workbench identifies items that have been changed but not yet saved by adding an asterisk (\*) in front of the item name.

When you select an item in the **Navigator**, its properties appear in the **Editor** (see [Section 5.3.4, "How to Use the Editor"](#)).

To perform specific functions for an item, select the item in the **Navigator** and do one of the following:

- Right-click on the object and select the function from the context menu (see [Section 5.3.1.2, "Using Context Menus"](#)).

- Choose a function from the **Selected** menu (see [Section 5.3.1.1, "Using Menu Bar Menus"](#)).

For information on using the Navigator with a database in relational projects, see [Section 5.5.1, "How to Use Database Tables in the Navigator Window"](#).

For information on using the Navigator with an XML schema in EIS projects (using XML records) and XML projects, see [Section 5.6.1, "How to Use XML Schemas in the Navigator"](#).

### Active and Inactive Descriptors

Inactive descriptors appear dimmed in the **Navigator**. Inactive descriptors are not registered with the session when the project is loaded into Java. This feature lets you define and test subsets of descriptors. To activate or deactivate a descriptor, right-click the descriptor and select **Activate/Deactivate Descriptor** from the context menu.

**Figure 5–5 Sample Active and Inactive Descriptors**

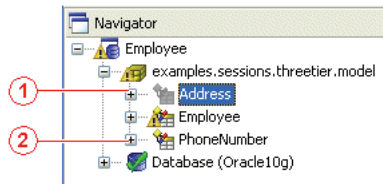


Figure 5–5 numbered callouts show the following user interface components:

- Inactive descriptor
- Active descriptor

### Errors and Missing Information



If an element in the project (such as a descriptor or mapping) contains an error or some deficiency (sometimes called *neediness*), a warning icon appears beside the element icon in the **Navigator**, and TopLink Workbench displays a message in the Problems window (see [Section 5.3.5, "How to Use the Problems Window"](#)).

[Section A.3, "TopLink Workbench Error Reference"](#) contains more information on each TopLink Workbench error message.

## 5.3.4 How to Use the Editor

The **Editor**, on the right side of the TopLink Workbench window, displays the property sheet associated with the currently selected item in the **Navigator**, as [Figure 5–6](#) shows.



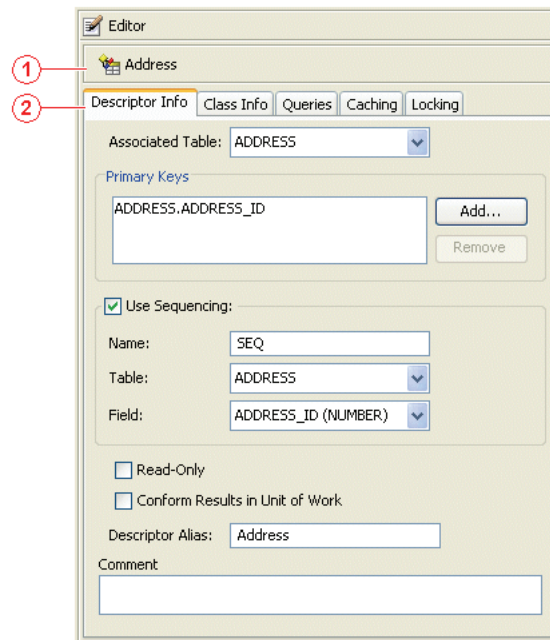
**Figure 5–6 Sample Editor**

Figure 5–6 numbered callouts identify the following user interface components:

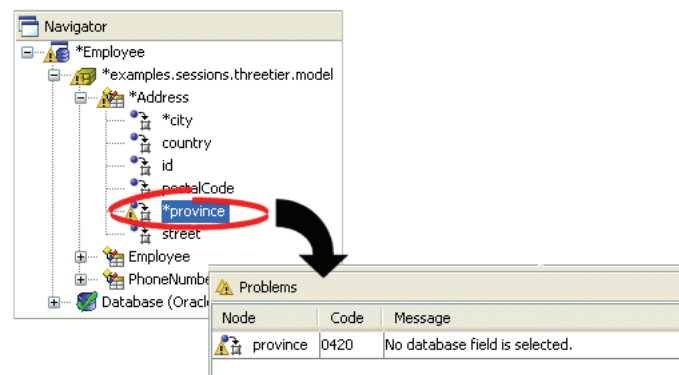
1. Selected element (from the **Navigator**)
2. Editor property tabs

### 5.3.5 How to Use the Problems Window



If an element in the project (such as a descriptor or mapping) contains an error or some deficiency (sometimes called *neediness*), the TopLink Workbench displays a caution icon (represented by a yellow triangle with a black exclamation point in the middle) to the left of the deficient element in the **Navigator** (see [Section 5.3.3, "How to Use the Navigator"](#)) and displays a message in the Problems window as [Figure 5–7](#) shows.

If you select the error, then TopLink Workbench displays the complete error message in the **Problems** window. [Section A.3, "TopLink Workbench Error Reference"](#) contains detailed information on each error message.

**Figure 5–7 Sample Deficient Mapping**

Double-click any error message in the **Problems** window to automatically highlight the specific node in the **Navigator**. To display or hide the **Problems** window, select **Window > Show Problems** from the menu.

You can also create a status report (see [Section 116.2.3, "How to Generate the Project Status Report"](#)) that includes all errors in a selected project.

### 5.3.6 How to Use the Online Help

TopLink Workbench contains an extensive online Help system to assist you in developing TopLink applications.

To receive help on any field, tab, or element in TopLink Workbench, right-click the element and select **Help** from the context menu or press **F1**.

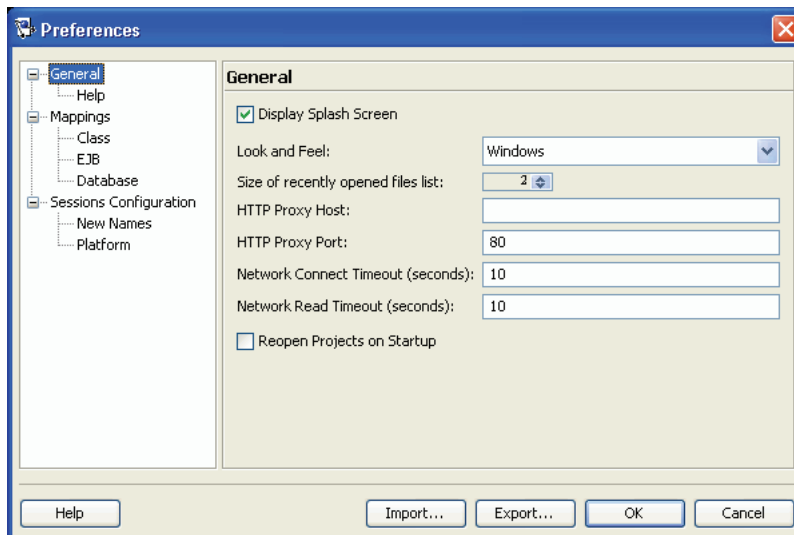


To review the complete TopLink documentation and Quick Start, click **Help**.

## 5.4 Using TopLink Workbench Preferences

To customize TopLink Workbench, select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.

**Figure 5–8 Preferences Dialog Box**



TopLink Workbench provides the following preferences:

- [How to Use General Preferences](#)
  - [How to Use Help Preferences](#)
- [How to Use Mappings Preferences](#)
  - [How to Use Class Preferences](#)
  - [How to Use EJB Preferences](#)
  - [How to Use Database Preferences](#)
- [How to Use Sessions Configuration Preferences](#)
  - [How to Use New Names Preferences](#)
  - [How to Use Session Platform Preferences](#)

Use this dialog box to configure TopLink Workbench preferences. After changing preferences, you must restart TopLink Workbench.

To import your preferences from an existing file, click **Import** and select the file.

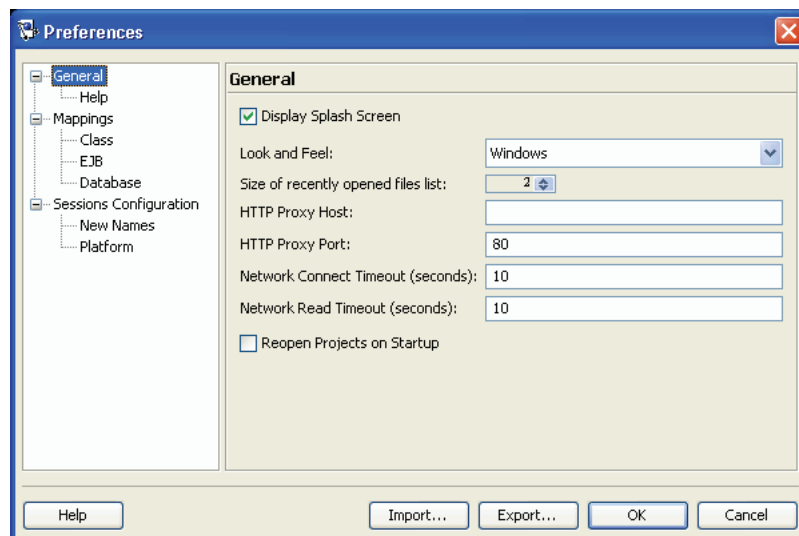
To export your preferences, click **Export** and select a directory location and filename.

## 5.4.1 How to Use General Preferences

Use the General preferences to customize the look and feel (the graphical user interface) of TopLink Workbench as well as to specify any proxy information required to access the Internet (for example, to allow TopLink to access XML schemas hosted on Internet sites). Follow these steps to customize the General preferences:

1. Select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.
2. Select **General** in the **Category** window.

**Figure 5–9 Preferences–General Dialog Box**



Use the following information to enter data in each field of the dialog box:

Field	Description
<b>Display Splash Screen</b>	Specify if TopLink Workbench should show the graphical splash screen when starting.
<b>Look and Feel</b>	Select the look and feel to use for TopLink Workbench.
<b>Size of recently opened files list</b>	Select the number of projects to maintain in the <b>File</b> menu. See <a href="#">Section 116.2.1, "How to Open Existing Projects"</a> for more information.
<b>HTTP Proxy Host</b>	Specify if your PC requires a proxy server to access the internet.
<b>HTTP Proxy Port</b>	Specify the port used by your proxy host.
<b>Network Connect Timeout</b>	Specify the timeout (in seconds) to establish a network or internet connection.
<b>Network Read Timeout</b>	Specify the timeout (in seconds) when accessing data from a network or internet connection.
<b>Reopen Projects on Startup</b>	Select to reopen the projects that were open the last time you exited the TopLink Workbench.

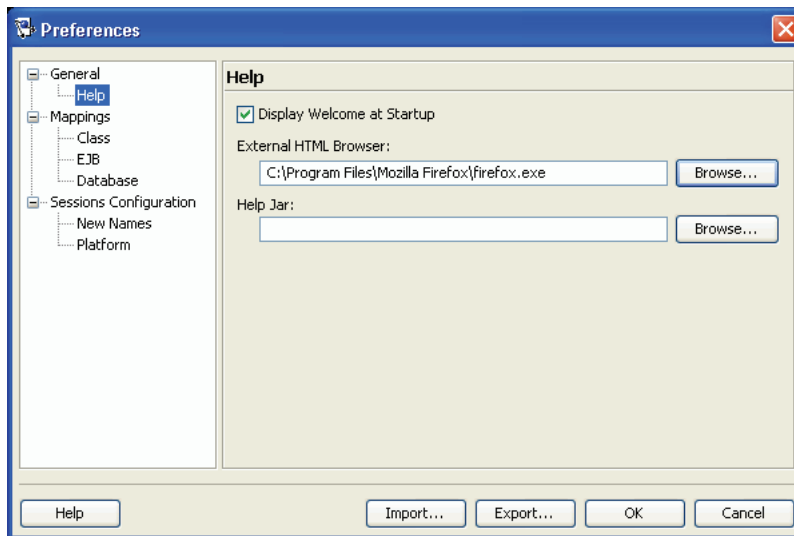
You must restart TopLink Workbench to apply the changes.

## 5.4.2 How to Use Help Preferences

Use the Help preferences to select the Help system preferences.

1. Select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.
2. Expand **General** in the **Category** window and select **Help**. The **Preferences–Help** dialog box appears.

**Figure 5–10 Preferences–General–Help Dialog Box**



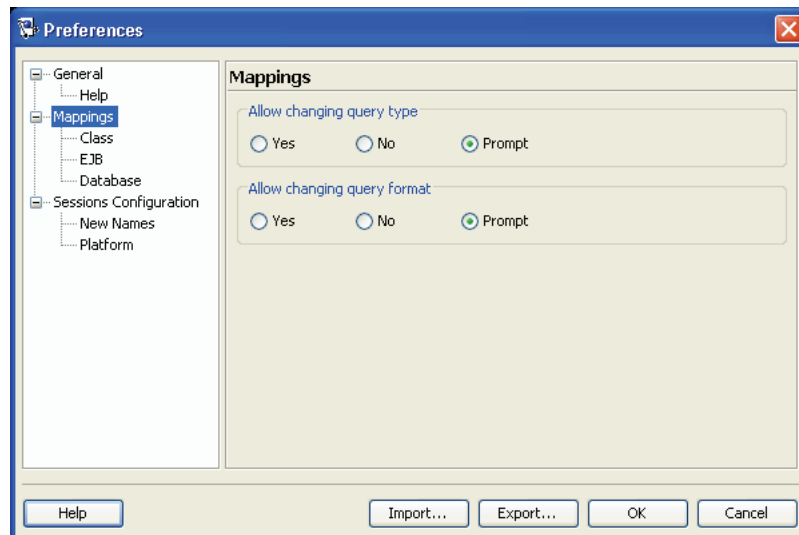
Use the following information to enter data in each field:

Field	Description
<b>Display Welcome at Startup</b>	Specify if TopLink should show the <b>Welcome</b> screen each time you start TopLink Workbench.
<b>External HTML Browser</b>	Click <b>Browse</b> and select the location of your default Web browser. You must specify a Web browser to access the Quick Tour, Javadoc (API), and other Web-based material.
<b>Help Jar</b>	Specify the location of the JAR file that contains help topics. Click <b>Browse</b> and select the help JAR file location. The default filename is <TOPLINK_HOME>/utils/jlib/help/tlmwhelp.jar To use the help content in a language other than the default language for your local, specify a different help JAR file. <b>Note:</b> You must restart TopLink Workbench to apply changes to this option.

## 5.4.3 How to Use Mappings Preferences

Use the Mappings preferences to specify general mapping preferences. Follow these steps to set the Mapping preferences:

1. Select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.
2. Select **Mappings** in the **Category** window. The **Mappings** dialog box appears.

**Figure 5–11 Preferences–Mappings Dialog Box**

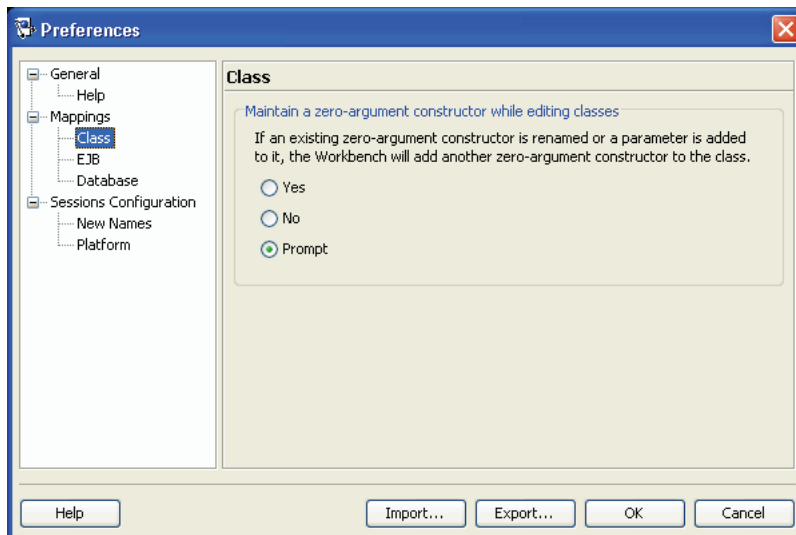
Use the following information to enter data in each field:

Field	Description
<b>Allow changing query type</b>	Configure whether or not TopLink Workbench always allows, never allows, or prompts before allowing you to change the query type associated with a descriptor.
<b>Allow changing query format</b>	Configure whether or not TopLink Workbench always allows, never allows, or prompts before allowing you to change the configuration of a query associated with a descriptor.

#### 5.4.4 How to Use Class Preferences

Use the Class preferences to specify how TopLink Workbench maintains classes when renaming or editing a zero-argument constructor. Follow these steps to set the Class preferences:

1. Select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.
2. Expand **Mappings** in the **Category** window and select **Class**.

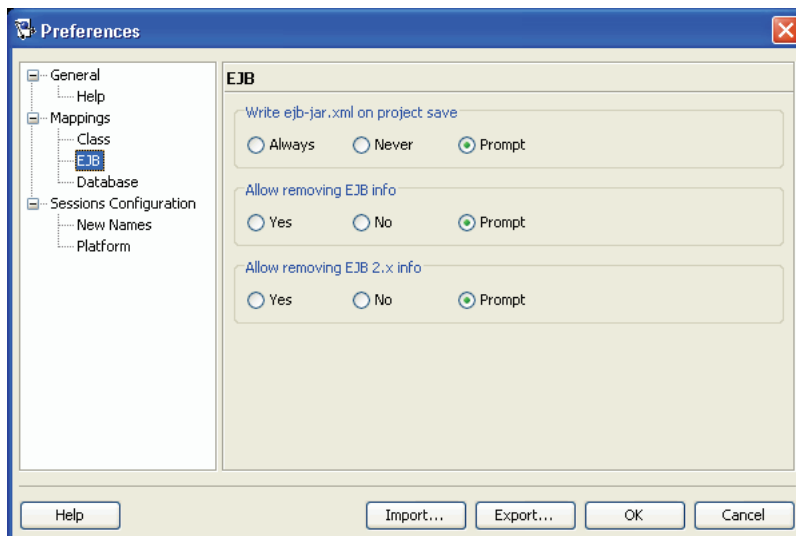
**Figure 5–12 Preferences – Mappings – Class Dialog Box**

On the **Preferences–Mappings–Class** dialog box, specify how TopLink Workbench maintains classes when renaming or editing a zero-argument constructor.

### 5.4.5 How to Use EJB Preferences

Use the EJB preferences to specify how TopLink Workbench updates the `ejb-jar.xml` file when saving EJB projects. Follow these steps to set the EJB preferences:

1. Select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.
2. Expand **Mappings** in the **Category** window and select **EJB**.

**Figure 5–13 Preferences–Mappings–EJB Preferences Dialog Box**

Use the following information to select how TopLink Workbench will update the `ejb-jar.xml` file:

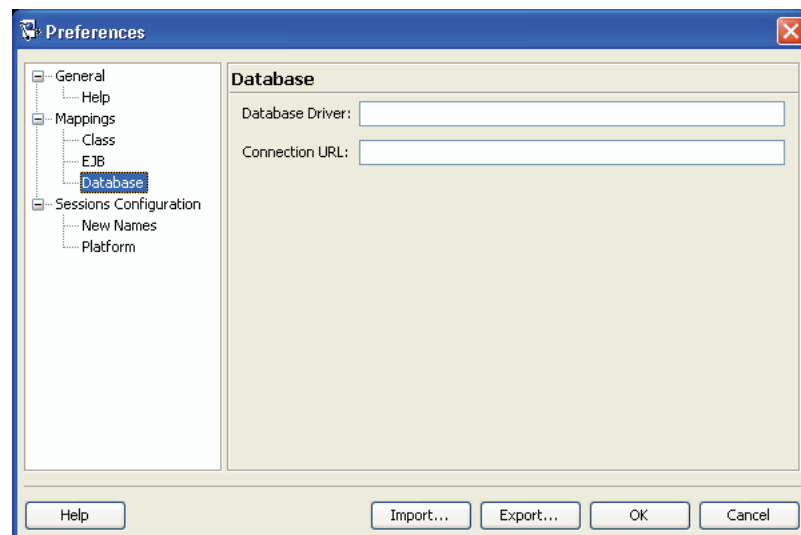
Field	Description
Write ejb-jar.xml on project save	Configure whether or not TopLink Workbench always updates, never updates, or prompts before updating the <code>ejb-jar.xml</code> file each time you save the project.
Allow removing EJB info	Configure whether or not TopLink Workbench always allows, never allows, or prompts before allowing you to remove the EJB information associated with a descriptor. See <a href="#">Section 119.18, "Configuring a Descriptor with EJB CMP and BMP Information"</a> for more information.
Allow removing EJB 2.x info	Configure whether or not TopLink Workbench always allows, never allows, or prompts before allowing you to remove the EJB 2.0 or 2.1 information associated with a descriptor. See <a href="#">Section 119.18, "Configuring a Descriptor with EJB CMP and BMP Information"</a> for more information.

## 5.4.6 How to Use Database Preferences

Use the Database preferences to specify custom database drivers and connection URLs for TopLink Workbench. These drivers and URLs can then be used when defining database logins. Follow these steps to set the Database preferences:

1. Select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.
2. Expand **Mappings** in the **Category** window and select **Database**.

**Figure 5–14 Preferences–Mappings–Database Preferences Dialog Box**



Use the following information to enter data in each field:

Field	Description
Database Driver	Enter the custom database driver class name.
Connection URL	Enter the custom database connection URL.

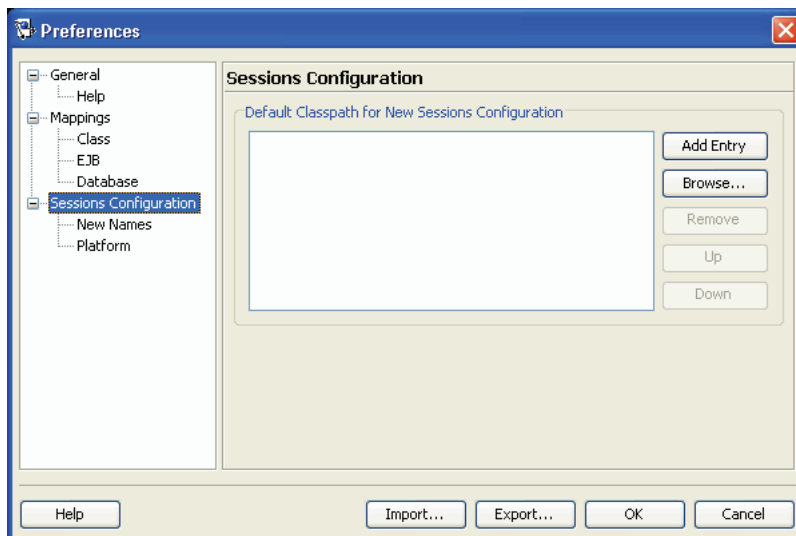
## 5.4.7 How to Use Sessions Configuration Preferences

Use the Sessions preferences to specify default classpaths to be added to each newly created TopLink sessions configuration for features that require an external Java class

(for example, session event listeners). The entries added here will automatically appear on the Sessions Configuration property sheet (see [Section 88.3, "Configuring a Sessions Configuration"](#)). Follow these steps to set the Sessions Configuration preferences:

1. Select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.
2. Select **Sessions Configuration** in the **Category** window.

**Figure 5–15 Preferences–Sessions Configuration Dialog Box**



To add a JAR or ZIP file, click **Add Entry** or **Browse** and add the JAR or ZIP files that contain the default compiled Java classes for this sessions configuration.

To remove a JAR or ZIP file, select the file and click **Remove**.

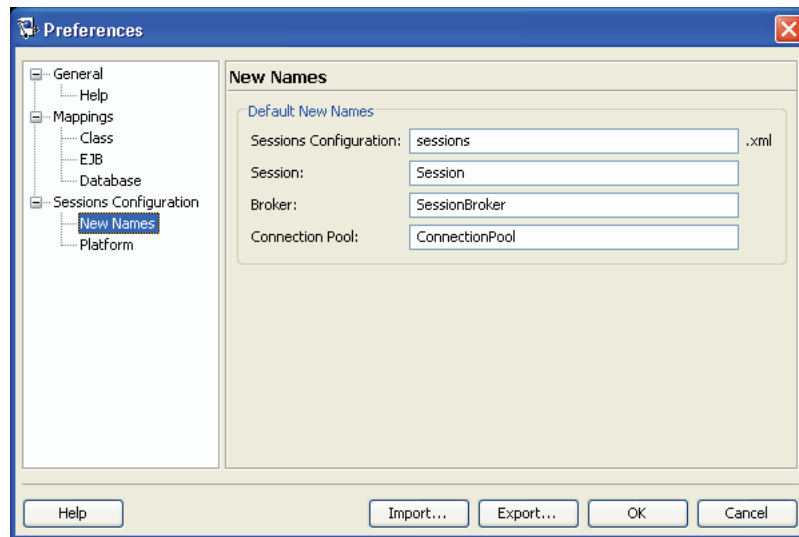
To change the order in which TopLink searches these JAR or ZIP files, select a file and click **Up** to move it up, or click **Down** to move it down in the list.

## 5.4.8 How to Use New Names Preferences

Use the New Names preferences to specify the default values and names of newly created sessions, session brokers, and connection pools. Follow these steps to set the New Names preferences:

1. Select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.
2. Expand **Sessions Configuration** in the **Category** window and select **New Names**.



**Figure 5–16 Preferences–Sessions Configuration–New Names Dialog Box**

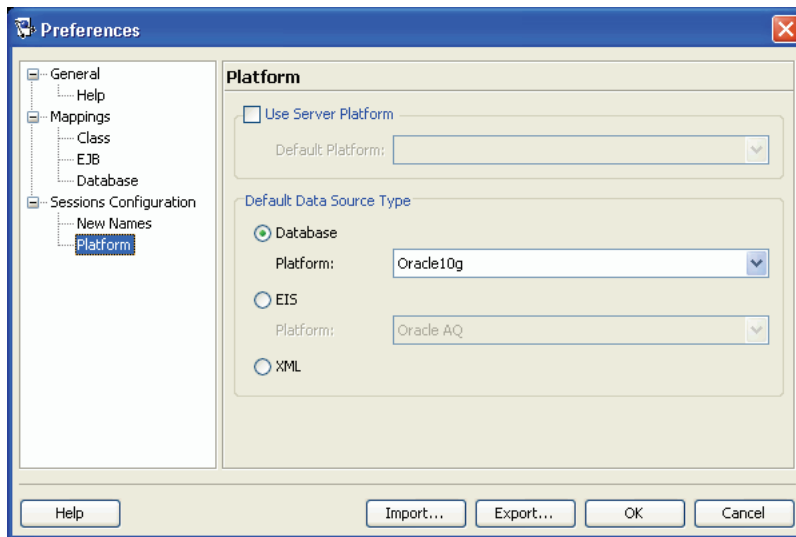
Use the following information to enter data in each field:

Field	Description
<b>Sessions Configuration</b>	Specify the default name for newly created sessions configuration files (default, <code>sessions.xml</code> ). See <a href="#">Section 88.2, "Creating a Sessions Configuration"</a> for more information.
<b>Session</b>	Specify the default name for newly created sessions (default, <code>Session</code> ). See <a href="#">Section 88.1, "Introduction to the Session Creation"</a> for more information.
<b>Broker</b>	Specify the default name for newly created session brokers (default, <code>SessionBroker</code> ). See <a href="#">Section 88.5, "Creating Session Broker and Client Sessions"</a> for more information.
<b>Connection Pool</b>	Specify the default name for newly created connection pools (default, <code>ConnectionPool</code> ). See <a href="#">Chapter 100, "Creating an Internal Connection Pool"</a> for more information.

### 5.4.9 How to Use Session Platform Preferences

Use the Platform preferences to specify the default data source type for newly created sessions. The type selected here will automatically appear on the Create New Session dialog box. Follow these steps to set the Platform preferences:

1. Select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.
2. Expand **Sessions Configuration** in the **Category** window and select **Platform**.

**Figure 5–17 Preferences–Sessions Configuration–Platform Preferences Dialog Box**

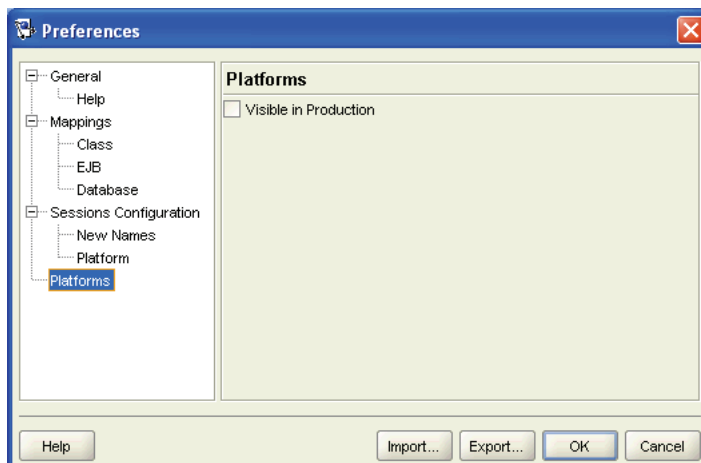
Use the following information to enter data in each field:

Field	Description
Use Server Platform	Specify the default application server platform for newly created sessions configuration files (default, <code>sessions.xml</code> ). See <a href="#">Section 88.2, "Creating a Sessions Configuration"</a> for more information.
Default Data Source Type	Select the default data source type ( <b>Database</b> , <b>EIS</b> , or <b>XML</b> ) and platform for newly created sessions. See <a href="#">Section 89.9, "Configuring the Server Platform"</a> for more information.

### 5.4.10 How to Use Platforms Preferences

Use the Platforms preferences to specify if platforms are to be visible in production. Follow these steps to set Platforms preferences:

1. Select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.
2. Select **Platforms** in the **Category** window.

**Figure 5–18 Preferences – Platforms Preferences**

Configure whether or not TopLink Workbench allows platforms to be visible in production.

## 5.5 Using Databases

In relational projects, when you expand the database object in the **Navigator**, TopLink Workbench displays the database tables associated with the project. You can associate tables by importing them from the database, or by creating them within TopLink Workbench.

**Figure 5–19 Sample Database Tables**

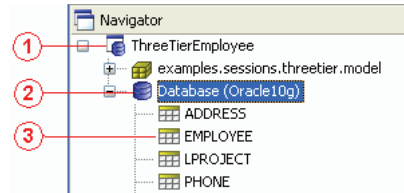


Figure 5–19 numbered callouts identify the following database icons.

1. Project
2. Database
3. Database table

Each database table property sheet contains the following tabs in the **Editor**:

- **Columns**—Add or modify the table's fields, and specify each field's properties.
- **References**—Specify references between tables.

This section includes information on the following topics:

- [How to Use Database Tables in the Navigator Window](#)
- [How to Use Database Tables in the Editor Window](#)
- [How to Generate Data from Database Tables](#)

### 5.5.1 How to Use Database Tables in the Navigator Window

This section describes the following options:

- [Logging In and Out of a Database](#)
- [Creating New Tables](#)
- [Importing Tables from a Database](#)
- [Removing Tables](#)
- [Renaming Tables](#)
- [Refreshing Tables from the Database](#)

See [Section 5.5.2, "How to Use Database Tables in the Editor Window"](#) for more information.

#### 5.5.1.1 Logging In and Out of a Database

To log in or out of a relational database, do the following:

1. Create a database login (see [Section 98.1, "Introduction to Database Login Configuration"](#)).
2. To log in to a relational database, right-click the database object in the **Navigator**, and choose **Log In to Database** from the context menu or choose **Selected > Log In to Database** from the menu.
3. To log out of a relational database, right-click the database object in the **Navigator** and choose **Log Out of Database** from the context menu or choose **Selected > Log Out of Database** from the menu.

### 5.5.1.2 Creating New Tables

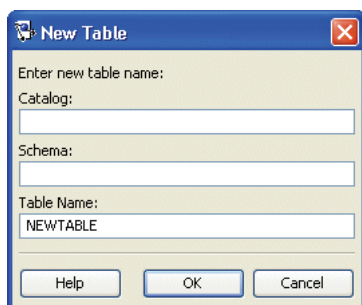
To create a new database table within TopLink Workbench, use the following procedure:



1. Select the database object in the **Navigator** window and click **Add New Table**. The **New Table** dialog box appears.

You can also right-click the database object and choose **Add New Table** from the context menu, or choose **Selected > Add New Table** from the menu.

**Figure 5–20 New Table Dialog Box**



Use the following information to enter data in each field:

Field	Description
<b>Catalog</b>	Use to identify specific database information for the table. Consult your database administrator for more information.
<b>Schema</b>	Use to identify specific database information for the table. Consult your database administrator for more information.
<b>Table Name</b>	Specify the name of this database table.

TopLink Workbench adds the database table to the project.

Although the database table has been added to the project, it has not been written to the actual database. See [Section 5.5.3.4, "Generating Tables on the Database"](#) for more information on creating the table in the database.

Continue with [Section 5.5.2, "How to Use Database Tables in the Editor Window"](#) to use these tables in your project.

### 5.5.1.3 Importing Tables from a Database

TopLink Workbench can automatically read the schema for a relational database and import the table data into the project as long as your JDBC driver supports the following JDBC methods:

- getTables
- getTableTypes
- getImportedKeys
- getCatalogs
- getPrimaryKeys

The JDBC driver must be on the TopLink Workbench classpath (see [Section 5.2, "Configuring the TopLink Workbench Environment"](#)).

To import tables from the database, use the following procedure:

1. Select the database object in the **Navigator**, and click **Add or Update Existing Tables from Database**. The **Import Tables from Database** dialog box appears.

You can also right-click on the database object in the **Navigator** and choose **Add or Update Existing Tables from Database** from the context menu or choose **Selected > Add or Update Existing Tables from Database** from the menu.

**Figure 5–21 Import Tables from Database Dialog Box**

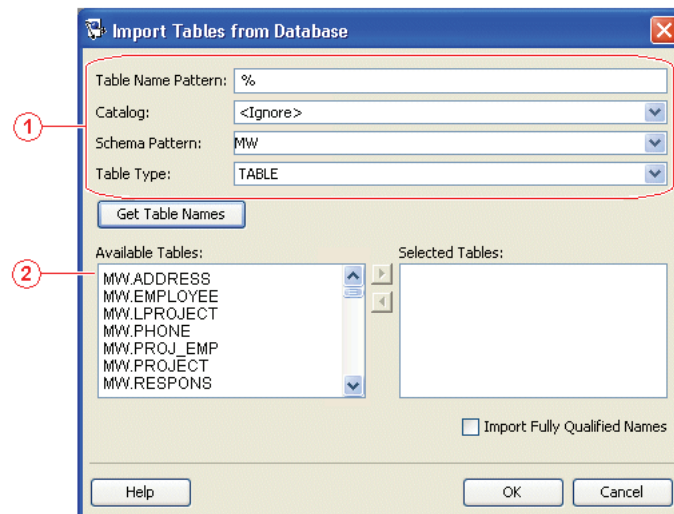


Figure 5–21 numbered callouts identify the following user interface components:

1. Filters
2. Database tables that match the filters

Use the following information to enter data in each field of the dialog box:

Field	Description
<b>Table Name Pattern</b>	Specify the name of database table(s) to import. Use percent character ( %) as a wildcard. Tables that match the <b>Table Name Pattern</b> can be imported.
<b>Catalog</b>	Specify the catalog of database table(s) to import.
<b>Schema Pattern</b>	Specify the schema of database table(s) to import.
<b>Table Type</b>	Specify the type of database table(s) to import.

Field	Description
Available Tables	Click <b>Get Table Names</b> to make TopLink display tables that match <b>Table Name Pattern</b> , <b>Catalog</b> , <b>Schema Pattern</b> , and <b>Table Type</b> settings.
Selected Tables	Select the tables in the <b>Available Tables</b> area to import, and click the right-arrow button. TopLink adds the table to the <b>Selected Tables</b> field.  Click <b>OK</b> to import the tables from the database into the TopLink Workbench project.
Import Fully Qualified Names	Specify whether or not the tables' names are fully qualified against the schema and catalog.

Examine each table's properties to verify that the imported tables contain the correct information. See [Section 5.5.2, "How to Use Database Tables in the Editor Window"](#) for more information.

#### 5.5.1.4 Removing Tables

To remove a database table from the project, use the following procedure:



1. Select a database table in the **Navigator**, and click **Remove Selected Item** on the toolbar. TopLink Workbench prompts for confirmation.

You can also right-click on the database object and choose **Remove** from the context menu or choose **Selected > Remove** from the menu.

2. Click **OK**. TopLink Workbench removes the table from the project.

---

**Note:** Although you have removed the table from the TopLink Workbench project, the table remains in the database.

---

#### 5.5.1.5 Renaming Tables

To rename a database table in the TopLink Workbench project, use the following procedure:

1. Select a database table in the **Navigator**, and click **Rename** on the toolbar. The Rename dialog box appears.

You can also right-click on the table and choose **Rename** from the context menu or choose **Selected > Rename** from the menu.

2. Enter a new name and click **OK**. TopLink Workbench renames the table.

---

**Note:** Although you have renamed the table in the TopLink Workbench project, the original table name remains in the database.

---

#### 5.5.1.6 Refreshing Tables from the Database

To refresh (that is, reload) the database tables in the TopLink Workbench project, use this procedure:

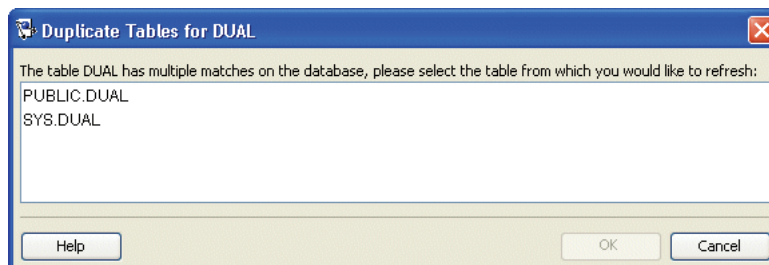


Select a database table in the **Navigator**, and click **Refresh from Database** on the toolbar.

You can also right-click on the table and choose Refresh from Database from the context menu or choose **Selected > Refresh from Database** from the menu. TopLink Workbench reloads the database table.

When refreshing tables from the database, if there are multiple database tables with similar names, the **Duplicate Tables** dialog box appears.

**Figure 5–22 Duplicate Table Dialog Box**



Select the specific database table to update, and then click **OK**.

## 5.5.2 How to Use Database Tables in the Editor Window

When you select a database table in the **Navigator**, its properties appear in the **Editor**. Each database table contains the following property tabs:

- **Columns**—Add or modify the table fields, and specify each field properties.
- **References**—Specify references between tables.

This section describes how to use these tabs to configure the following:

- [Working with Column Properties](#)
- [Setting a Primary Key for Database Tables](#)
- [Creating Table References](#)
- [Creating Field Associations](#)

### 5.5.2.1 Working with Column Properties

Use the database table's **Column** tab to specify properties for the database table's fields.

To specify a table's column properties, use this procedure:

1. Select a database table in the **Navigator**. The table's property sheet displays in the **Editor**.
2. Click the Columns tab.

**Figure 5–23 Fields Properties**

Name	Type	Size	Sub-Size	Allows Null	Unique	Primary Key	Identity
DESCRIP	VARCHAR2	200		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
LEADER_ID	NUMBER	15	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
PROJ_ID	NUMBER	15	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
PROJ_NAME	VARCHAR2	30		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
PROJ_TYPE	VARCHAR2	1		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
VERSION	NUMBER	15	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

3. Enter data in each field on the Columns tab. Use the scroll bar to display the additional field.

Use the following information to fill each column on the Columns tab:

Field	Description
<b>Name</b>	Specify the name of the field.
<b>Type</b>	Use the drop-down list to select the field's type. Note: The valid values will vary, depending on the database.
<b>Size</b>	Specify the size of the field.
<b>Sub-Size</b>	Specify the sub-size of the field.
<b>Allows Null</b>	Specify if this field can be null.
<b>Unique</b>	Specify whether the value must be unique within the table.
<b>Primary Key</b>	Specify whether or not this field is a primary key for the table (see <a href="#">Section 5.5.2.2, "Setting a Primary Key for Database Tables"</a> ).
<b>Identity</b>	Use to indicate a Sybase, SQL Server or Informix identity field.

---

**Note:** Some properties may be unavailable, depending on your database type.

---

To add a new field, click **Add**.

To remove a field, select the field and click **Remove**.

To rename a field, select the field and click **Rename**.

### 5.5.2.2 Setting a Primary Key for Database Tables

To set a primary key(s) for a database table, use this procedure:

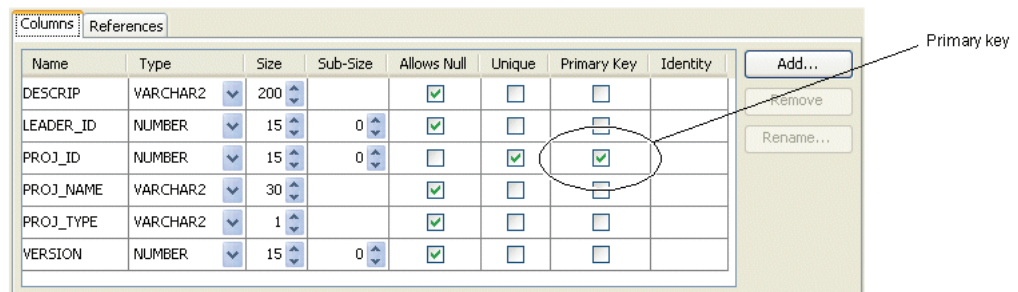
---

**Note:** TopLink Workbench can automatically import primary key information if supported by the JDBC driver.

---

1. Select a database table in the **Navigator**. Its property sheet appears in the **Editor**.
2. Click the **Columns** tab.



**Figure 5–24 Setting Primary Key for a Database Table**

3. Select the **Primary Key** field(s) for the table.

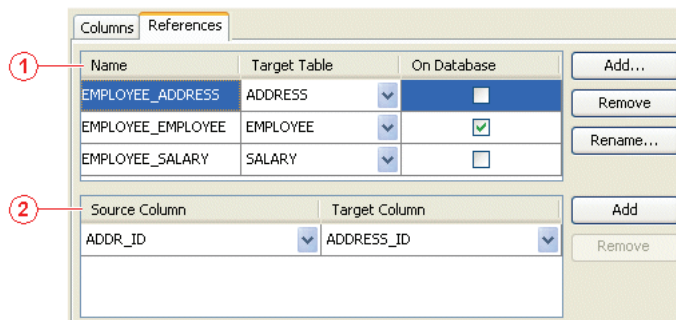
### 5.5.2.3 Creating Table References

References are table properties that contain the foreign key; they may or may not correspond to an actual constraint that exists on the database. TopLink Workbench uses these references when you define relationship mappings and multiple table associations.

When importing tables from the database, TopLink Workbench can automatically create references (if the driver supports this), or you can define references from the workbench. See [Section 5.5.1.3, "Importing Tables from a Database"](#).

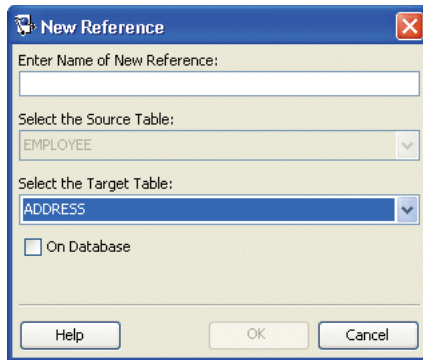
To create a new table reference, use this procedure:

1. Select a database table in the **Navigator**. The table's properties display in the **Editor**.
2. Click the **References** tab.

**Figure 5–25 References Tab**

[Figure 5–27](#) numbered callouts identify the following user interface components:

1. Table References area
2. Key Pairs area
3. In the **References** area, click **Add**. The **New Reference** dialog box appears.

**Figure 5–26 New Reference Dialog Box**

Use the following information to enter data in each field of the dialog box:

Field	Description
<b>Enter Name of New Reference</b>	Specify the name of the reference table. If you leave this field blank, TopLink Workbench automatically creates a name based on the format: SOURCETABLE_TARGETTABLE.
<b>Select the Source Table</b>	Specify the name of the source database table (the currently selected table in the Navigator).
<b>Select the Target Table</b>	Use the list to specify the target table for this reference.
<b>On Database</b>	Specify if you want to create the reference on the database when you create the table. Not all database drivers support this option.

Continue with [Section 5.5.2.4, "Creating Field Associations"](#).

#### 5.5.2.4 Creating Field Associations

For each table reference, you can specify one or more field associations that define how fields in the source table relate to fields in the target table. See [Section 5.5.2.3, "Creating Table References"](#).

To create new field references, use this procedure:

1. Select a database table in the Navigator. The table's properties display in the **Editor**.
2. Click the **References** tab.

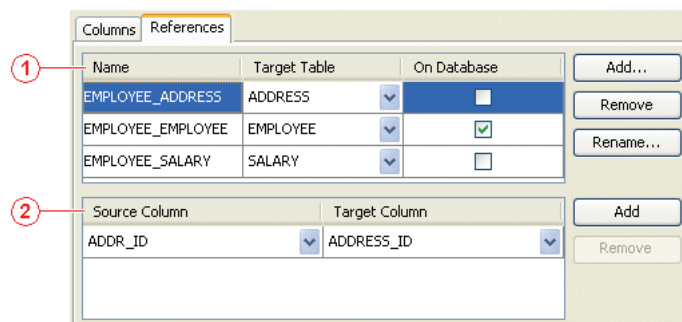
**Figure 5–27 References Tab**

Figure 5–27 numbered callouts identify the following user interface components:

1. Table references area
2. Key pairs area
3. Select a table reference from the references area.
4. To create a new key pair, click **Add** in the key pairs area and complete each field in the key pairs area using the following information:

To create a new key pair, click **Add** in the key pairs area and complete each field in the key pairs area using the following information.

Field	Description
<b>Table References Area</b>	
<b>Name</b>	Specify the name of this table reference
<b>Target Table</b>	Specify the database table that is the <i>target</i> of this reference.
<b>On Database</b>	Specify if the reference exists on the database.
<b>Key Pairs Area</b>	
<b>Source Column</b>	Select the database field from the source table.
<b>Target Column</b>	Select the database field from the target table.

### 5.5.3 How to Generate Data from Database Tables

TopLink Workbench can automatically generate a variety of information from the database tables. This section describes the following:

- [Generating SQL Creation Scripts](#)
- [Generating Classes and Descriptors from Database Tables](#)
- [Generating EJB Entity Beans and Descriptors from Database Tables](#)
- [Generating Tables on the Database](#)

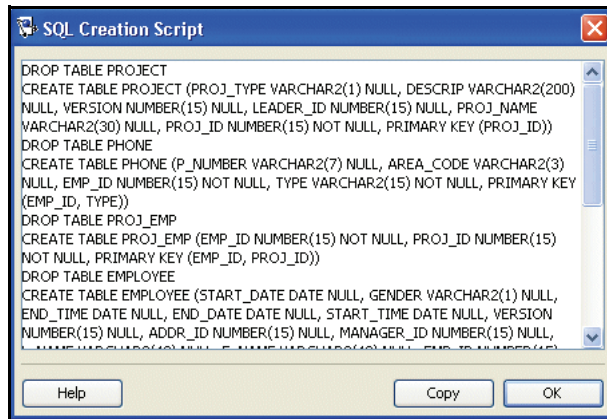
#### 5.5.3.1 Generating SQL Creation Scripts

Using the TopLink Workbench, you can generate SQL scripts that you can use to create tables in a relational database.

To automatically generate SQL scripts to create the tables in a project, use this procedure:

1. Select the database table(s) in the **Navigator**.
2. Right-click the table(s) and choose **Generate Creation Script for > Selected Tables** or **All Tables** from the context menu. The SQL Creation Script dialog box appears.

You can also choose **Selected > Generate Creation Script for > Selected Tables** or **All Tables** from the menu.

**Figure 5–28 SQL Creation Script Dialog Box**

Copy the script and paste it into a file. You may need to edit the file to include additional SQL information that TopLink Workbench could not generate. If the database table or column name is an SQL reserved word, you must edit the SQL script and enclose the database table or column in quotes. See "Oracle Database Reserved Words" in the *Oracle Database SQL Reference Guide* for more information.

---

**Note:** If TopLink cannot determine how a particular table feature should be implemented in SQL, it generates a descriptive message in the script.

---

### 5.5.3.2 Generating Classes and Descriptors from Database Tables

TopLink Workbench can automatically generate Java class definitions, descriptor definitions, and associated mappings from the information in database tables. You can later edit the generated information if necessary.

For each table, TopLink Workbench does the following:

- Creates a class definition and a descriptor definition.
- Adds attributes to the class for each column in the table.
- Automatically generates access methods, if specified.
- Creates direct-to-field mappings for all direct (nonforeign key) fields in the table.
- Creates relationship mappings (one-to-one and one-to-many) if there is sufficient foreign key information. You may be required to determine the exact mapping type.

---

**Note:** Class and attribute names are generated based on the table and column names. You can edit the class properties to change their names.

---

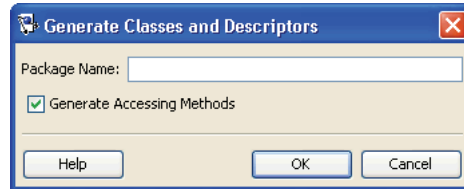
**To generate classes and descriptors from database tables, use the following procedure:**

1. Select the database table(s) in the **Navigator**.
2. Right-click the table(s) and choose **Generate Classes and Descriptors from > Selected Tables** or **All Tables** from the context menu. TopLink Workbench prompts you to save your project.

You can also choose **Selected > Generate Classes and Descriptors from > Selected Tables** or **All Tables** from the menu.

3. Click **Yes**. The Generate Classes and Descriptors dialog box appears.

**Figure 5–29 Generate Classes and Descriptors Dialog Box**

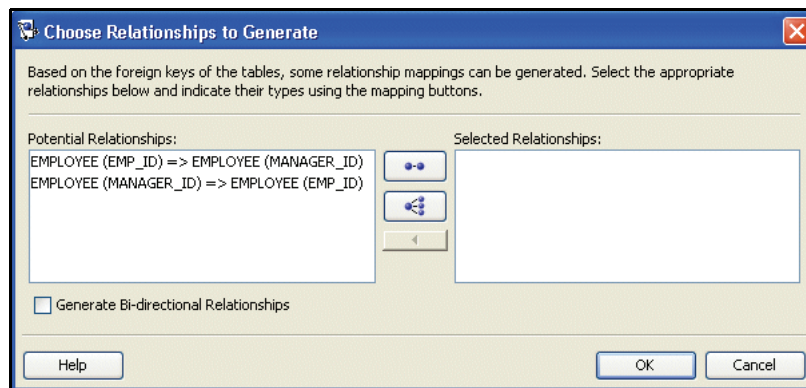


Use the following information to enter data in each field:

Field	Description
Package Name	Specify the name of package to generate. The package name must comply with Java naming standards.
Generate Accessing Methods	Specify if TopLink Workbench generates accessing methods for each class and descriptor.

If the table contains foreign key fields that may represent relationship mappings, then the Choose Relationships to Generate dialog box appears.

**Figure 5–30 Choose Relationships to Generate Dialog Box**



Select an entry from **Potential Relationships** and click the **1:1 Mapping** or **1:M Mapping** button, located between the Potential Relationships and Selected Relationships windows. See [Chapter 27, "Introduction to Relational Mappings"](#) for more information on mappings.

You can also specify whether the relationships are bidirectional. See [Section 121.18, "Configuring Bidirectional Relationship"](#) for more information.

Click **OK** to automatically create the relationships.

The newly created descriptors appear in the **Navigator** of TopLink Workbench.

### 5.5.3.3 Generating EJB Entity Beans and Descriptors from Database Tables

Using TopLink Workbench, you can automatically generate EJB entity beans and descriptors for each database table, including the following:

- One EJB descriptor that implements the `<javax.ejb.EntityBean>` and entity bean classes
- Bean relation attributes (CMP or BMP)
- Java source for each class
- EJB-compliant method stubs

---

**Note:** This option is available only for projects with container-managed or bean-managed persistence. See [Section 117.5, "Configuring Persistence Type"](#) for more information.

---

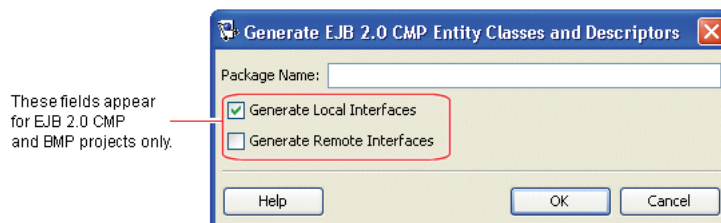
To automatically generate EJB entity beans and descriptors for each database table, use this procedure:

1. Select the database table(s) in the **Navigator**.
2. Right-click the table(s) and choose **Generate EJB Entities and Descriptors from > Selected Table** or **All Tables** from the context menu. TopLink Workbench prompts you to save your project.

You can also choose **Selected > Generate EJB Entities and Descriptors from > Selected Table** or **All Tables** from the menu.

3. Click **Yes** to save your project before generating EJB entities. The Generate EJB Entity Classes and Descriptors dialog box appears.

**Figure 5–31** *Generate EJB Entity Classes and Descriptors Dialog Box*



Use the following information to enter data in each field on the Generate EJB Entity Classes and Descriptors dialog box:

Field	Description
Package Name	Name of the package to contain the generated entity beans and descriptors.
Generate Local Interfaces <sup>1</sup>	Specify if TopLink creates local interfaces for the EJB entity beans.
Generate Remote Interfaces <sup>1</sup>	Specify if TopLink creates remote interfaces for the EJB entity beans.

<sup>1</sup> For CMP and BMP projects only. See [Section 119.18, "Configuring a Descriptor with EJB CMP and BMP Information"](#) for more information.





If the table contains foreign key fields that may represent relationship mappings, then the Choose Relationships to Generate dialog box appears. Select a potential relationship and click the **1:1 Mapping** or **1:M Mapping** button, located between the Potential Relationships and **Selected Relationships** windows.

You can also specify if the relationships are bidirectional (see [Section 121.18, "Configuring Bidirectional Relationship"](#)).

Repeat for all appropriate sets of tables.

Click **OK** to generate the relationship mappings.

The system creates the remote primary key, home, and bean classes for each bean and adds this information to the project.

#### 5.5.3.4 Generating Tables on the Database

To create a table in the database, based on the information in TopLink Workbench, use this procedure:

---



---

**Note:** You must log in the database before creating tables. See [Section 20.6, "Logging In to the Database"](#) for more information.

---



---

1. Select the database table(s) in the **Navigator**.
2. Right-click the table(s) and choose **Create on Database > Selected Table** or **All Tables** from the context menu.

You can also create tables by selecting **Selected > Create on Database > Selected Table** or **All Tables** from the menu.

TopLink Workbench creates the tables on the database.

Alternatively, you can generate tables at run time by exporting the information in TopLink Workbench to a `TableCreator` class (see [Section 6.1, "Introduction to the Schema Manager"](#)).

## 5.6 Using XML Schemas

For XML and EIS projects, TopLink Workbench maps each TopLink descriptor to your XML schema.

This section includes information on the following topics:

- [How to Use XML Schemas in the Navigator](#)
- [How to Use an XML Schema Structure](#)
- [How to Import an XML Schema](#)
- [How to Configure an XML Schema Reference](#)
- [How to Configure XML Schema Namespace](#)

### 5.6.1 How to Use XML Schemas in the Navigator

After you import one or more XML schemas into your project (see [Section 5.6.3, "How to Import an XML Schema"](#)) and you expand the schema object in the **Navigator**, TopLink Workbench displays the schemas associated with the project.

Figure 5–32 Sample XML Schemas

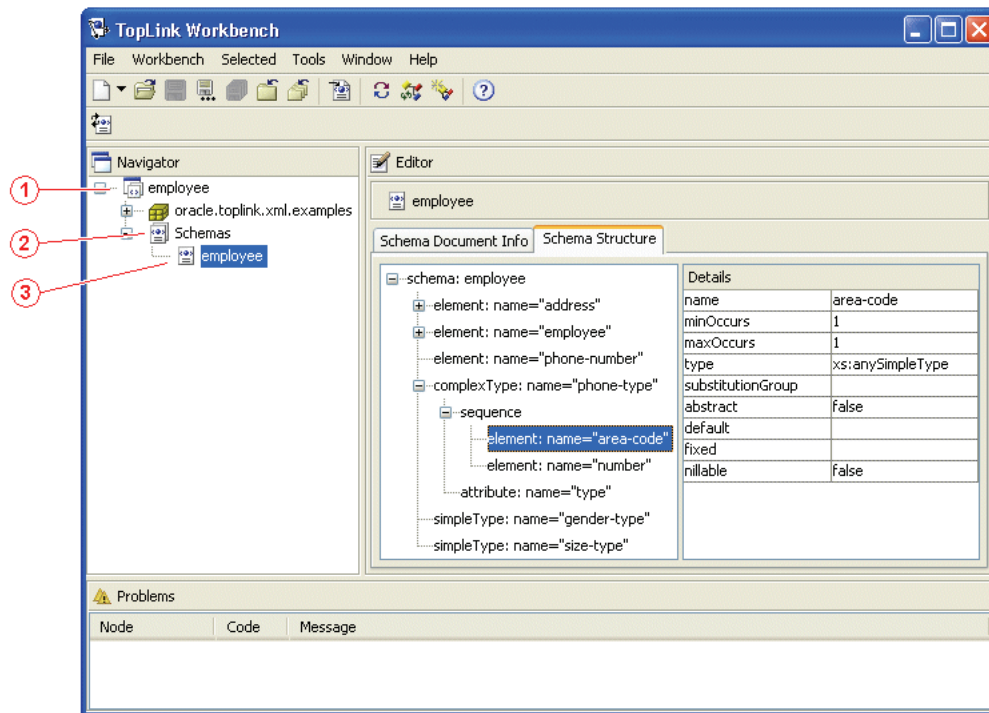


Figure 5–32 numbered callouts identify the following schema icons:

1. Project
2. Schemas object
3. Specific schema

For more information, see the following:

- [Section 5.6.2, "How to Use an XML Schema Structure"](#)
- [Section 5.6.4, "How to Configure an XML Schema Reference"](#)
- [Section 5.6.5, "How to Configure XML Schema Namespace"](#)

## 5.6.2 How to Use an XML Schema Structure

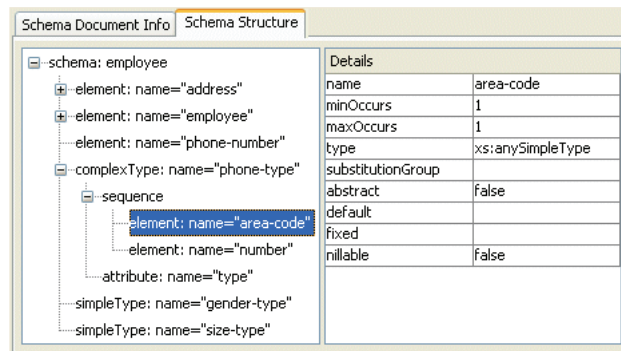
When you select a specific XML schema in the **Navigator**, you can display the structure and details of the schema using the Schema Structure tab.

To display the structure and details of a schema, use this procedure:

1. Select a schema element in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Schema Structure** tab. The Schema Structure tab appears.
3. Select an element in the schema. The element's details appear.



Figure 5–33 Schema Structure Tab



Use the following information to verify data in each field in the Schema Document Info tab:

Field	Description
Schema Structure	Displays the elements of the schema, listed in alphabetical order, in an expandable or collapsible tree structure.
Details	Displays detailed information (such as name and type) for the currently selected element in the <b>Schema Structure</b> area.

These fields are for display only and cannot be changed in TopLink Workbench.

### 5.6.3 How to Import an XML Schema

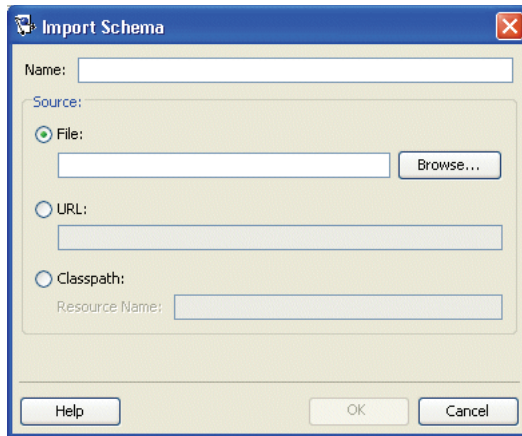
The first step in configuring an EIS project (using XML records) or XML project is importing the XML schema(s) that your project uses.

When you import a schema, you define a schema reference that gives TopLink the information it needs to locate the schema itself. Anytime after you import an XML schema, you can update the schema reference (see [Section 5.6.4, "How to Configure an XML Schema Reference"](#)) if necessary.

After importing an XML schema, you can configure XML schema namespaces (see [Section 5.6.5, "How to Configure XML Schema Namespace"](#)).

To import an XML schema into an EIS project (using XML records) or an EIS project, use this procedure:

1. Right-click the schemas element in the **Navigator** and select **Import Schema** from the context menu. The Import Schema dialog box appears.

**Figure 5–34 Import Schema Dialog Box**

Use the following information to enter data in each field in the Import Schema dialog box:

Field	Description
<b>Name</b>	Specify the name of this schema. This is the display name that TopLink Workbench uses. It can be different than the name you specify when you configure <b>Source</b> .
<b>Source</b>	Select how TopLink Workbench should import the schema.
<b>File</b>	Specify that TopLink Workbench should import the schema from a file. Enter the fully qualified directory path and filename of the schema file.
<b>URL</b>	Specify that TopLink Workbench should import the schema using a URL. Enter the complete URL of the schema file. Note: When importing schemas by URL, ensure you have set your proxy information correctly. See <a href="#">Section 5.4.1, "How to Use General Preferences"</a> for more information.
<b>Classpath</b>	Specify that TopLink Workbench should import the schema from the project classpath.
<b>Resource Name</b>	Enter the fully qualified name of the XML schema file including the name of the package of which it is a part. For example, if your XML schema <code>mySchema.xsd</code> is in <code>C:\project\config</code> and you add this directory to your project classpath (see <a href="#">Table 117–4, "Project Support for Project Classpath"</a> ), specify a resource name of <code>project.config.mySchema.xsd</code> .

To reimport *a specific schema*, right-click on the specific schema in the **Navigator** and select **Reimport Schema** from the context menu.

To reimport *all schemas in a project*, right-click on **Schemas** in the **Navigator** and select **Reimport All Schemas** from the context menu.

To change a schema's source, right-click on the specific schema in the Navigator window and select **Properties** from the context menu. The Schema Properties dialog appears.

## 5.6.4 How to Configure an XML Schema Reference

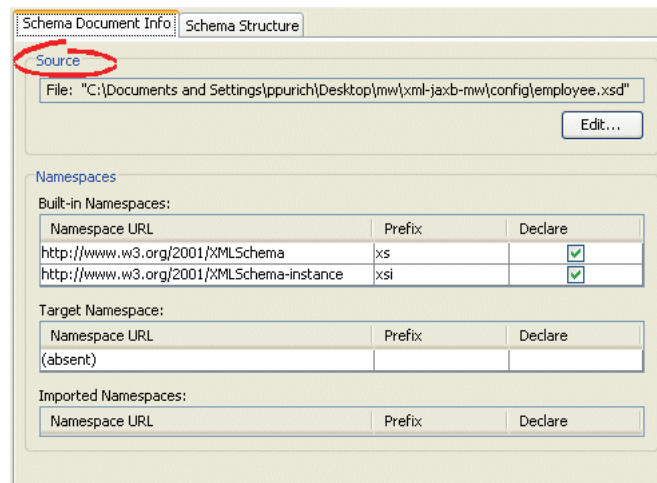
After you import an XML schema (see [Section 5.6.3, "How to Import an XML Schema"](#)), you can update its source by configuring the schema reference.

### 5.6.4.1 How to Configure an XML Schema Reference Using TopLink Workbench

To specify the source of a schema, use this procedure:

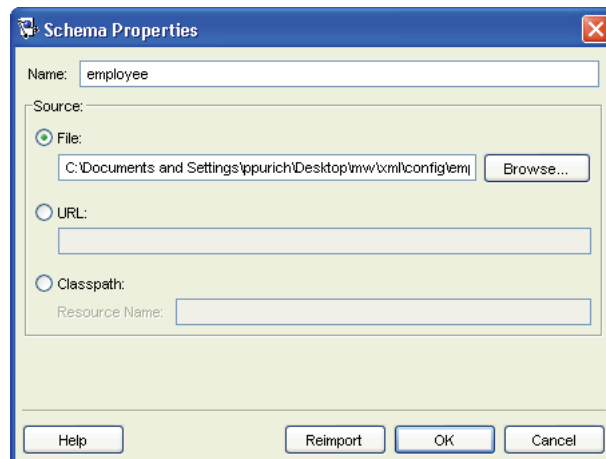
1. Select a schema element in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Schema Document Info** tab. The Schema Document Info tab appears.

**Figure 5–35 Schema Document Info Tab–Source Field**



3. Click **Edit** to select a new source for the selected schema. The Schema Properties dialog box appears.

**Figure 5–36 Schema Properties Dialog Box**



Use the following information to complete each field in the Schema Properties dialog box:

Field	Description
<b>Name</b>	Specify the name of this schema. This is the display name that TopLink Workbench uses. It can be different than the name you specify when you configure <b>Source</b> .
<b>Source</b>	Select how TopLink Workbench should import the schema.
<b>File</b>	Specify that TopLink Workbench should import the schema from a file. Enter the fully qualified directory path and filename of the schema file.
<b>URL</b>	Specify that TopLink Workbench should import the schema using a URL. Enter the complete URL of the schema file. <b>Note:</b> When importing schemas by URL, ensure you have set your proxy information correctly. See <a href="#">Section 5.4.1, "How to Use General Preferences"</a> for more information.
<b>Classpath</b>	Specify that TopLink Workbench should import the schema from the project classpath.
<b>Resource Name</b>	Enter the fully qualified name of the XML schema file including the name of the package of which it is a part. For example, if your XML schema mySchema.xsd is in C:\project\config and you add this directory to your project classpath (see <a href="#">Section 117.3, "Configuring Project Classpath"</a> ), specify a resource name of project.config.mySchema.xsd.

#### 5.6.4.2 How to Configure an XML Schema Reference Using Java

Use Java to configure schema reference. Create a descriptor amendment method (see [Section 119.35, "Configuring Amendment Methods"](#)) that instantiates the appropriate type of XMLSchemaReference (XMLSchemaClassPathReference, XMLSchemaFileReference, or XMLSchemaURLReference) and configures the descriptor with it, as follows:

- If you are using EISDescriptors, the TopLink runtime does not use the schema reference; no further configuration is required.
- If you are using XMLDescriptors, configure the descriptor with the XMLSchemaReference using XMLDescriptor method setSchemaReference.

#### 5.6.5 How to Configure XML Schema Namespace

As defined in <http://www.w3.org/TR/REC-xml-names/>, an **XML namespace** is a collection of names, identified by a URI reference, which are used in XML documents as element types and attribute names. To promote reusability and modularity, XML document constructs should have universal names, whose scope extends beyond their containing document. XML namespaces are the mechanism which accomplishes this.

When you import an XML schema (see [Section 5.6.3, "How to Import an XML Schema"](#)) such as the one that [Example 5–3](#) shows, TopLink Workbench organizes the various namespaces that the XML schema identifies, as [Table 5–3](#) shows.

##### **Example 5–3 XML Schema with Namespace Options**

```
<xsd:schema
  xmlns:<prefix>=<URI>"          <!-- TopLink Workbench Built-in Namespace -->
  targetNamespace=<URI>"         <!-- TopLink Workbench Target Namespace -->
  elementFormDefault="qualified"
```

```

attributeFormDefault="unqualified"
version="10.1.3">
<xsd:import
                                <!-- TopLink Workbench Imported Namespace -->
    namespace="http://xmlns.oracle.com/ias/xsds/opm"
    schemaLocation="object-persistence_1_0.xsd"
  />
...
</xsd:schema>

```

**Table 5–3 TopLink Workbench XML Schema Categories**

TopLink Workbench Category	Defined By	Purpose	When Needed
Built-in	<code>xmlns:&lt;prefix&gt;=&lt;URI&gt;</code>	Provides access to types defined in other XML schemas for use as is.	If your project uses more than one XML schema or if you want to use <code>xsi</code> or <code>xsd</code> types.
Target	<code>targetNamespace=&lt;URI&gt;</code>	The namespace you use to qualify the types you define for your application. If set, all XML documents that use these types must use this namespace qualifier.	You may need to specify a target namespace depending on how element and attribute form options are set (see <a href="#">Section 15.4.2, "Element and Attribute Form Options"</a> ).
Imported	<code>xsd:import</code>	Provides access to types defined in the corresponding built-in XML schema so that you can extend the built-in types. Extended types must be qualified by the target namespace.	If your project uses more than one XML schema and you want to extend one or more built-in types.

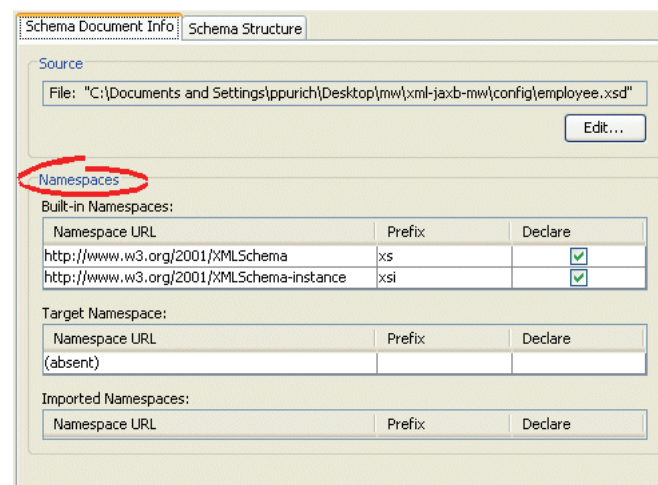
For more information, see [Section 15.4, "XML Namespaces Overview"](#).

### 5.6.5.1 How to Configure XML Schema Namespace Using TopLink Workbench

To specify the namespaces of a schema, use this procedure:

1. Select a schema element in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Schema Document Info** tab. The Schema Document Info tab appears.

**Figure 5–37 Schema Document Info Tab—Namespaces Field**



Use the following information to complete each Namespaces field in the tab:

Field	Description
<b>Built-in Namespaces</b>	All namespaces defined by <code>xmlns:&lt;prefix&gt;="&lt;URI&gt;"</code> . Note that when a schema is imported to the TopLink Workbench (see <a href="#">Section 5.6.3, "How to Import an XML Schema"</a> ), none of the built-in namespaces' URLs are selected. If you are using inheritance, declare the built-in namespace with <code>xsi</code> prefix. Otherwise, TopLink will throw exceptions.
<b>Target Namespaces</b>	All namespaces defined by <code>targetNamespace="&lt;URI&gt;"</code> .
<b>Imported Namespaces</b>	All namespaces defined by <code>xsd:import</code> .
<b>Prefix</b>	Double-click in the <b>Prefix</b> field to specify the prefix that corresponds to the given namespace.  When the TopLink runtime marshalls (writes) an object to an XML document, it uses the namespace prefixes you specify here.  When the TopLink runtime unmarshalls (reads) an XML document, the document may use any prefix value as long as it corresponds to the appropriate namespace. For more information, see <a href="#">Section 15.4.3, "TopLink Runtime Namespace Resolution"</a> .
<b>Declare</b>	When selected, XML documents must use the corresponding URI qualifier when referring to types from this namespace. XML documents may use a different prefix value as long as that value is associated with the appropriate namespace URI. For more information, see <a href="#">Section 15.4.3, "TopLink Runtime Namespace Resolution"</a> .

### 5.6.5.2 How to Configure XML Schema Namespace Using Java

Using Java, to configure XML schema namespaces for an EIS descriptor (with XML records) or an XML descriptor, create a descriptor amendment method (see [Section 119.35, "Configuring Amendment Methods"](#)) that uses `EISDescriptor` or `XMLDescriptor` method `getNamespaceResolver` to configure the descriptor's `NamespaceResolver` accordingly, as [Example 5-4](#) shows.

#### **Example 5-4** Configuring Namespaces

```
public void addToDescriptor(ClassDescriptor descriptor) {
    descriptor.getNamespaceResolver.put(
        prefix,
        namespaceURI
    );
}
```

## 5.7 Using Classes

Using TopLink Workbench, you can create Java classes and packages. This section includes information on the following:

- [How to Create Classes](#)
- [How to Configure Classes](#)
- [How to Import and Update Classes](#)
- [How to Manage Nondesoriptor Classes](#)
- [How to Rename Packages](#)

## 5.7.1 How to Create Classes

Oracle recommends that you develop your Java classes using an IDE, such as Oracle Oracle JDeveloper, and import these existing classes into TopLink Workbench (see [Section 5.7.3, "How to Import and Update Classes"](#))

However, it is sometimes convenient to create and configure classes in TopLink Workbench: for example, when generating an object model from a database schema.

This section includes information on using TopLink Workbench to create Java classes.

For more information on using TopLink Workbench to edit classes, see [Section 5.7.1, "How to Create Classes"](#).

### 5.7.1.1 How to Create Classes Using TopLink Workbench

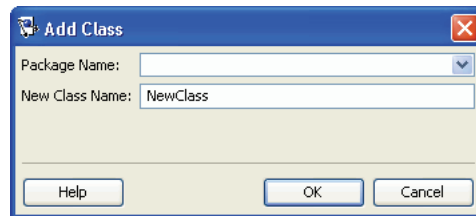
To create new classes and packages from within TopLink Workbench, use this procedure:



1. Select the project in the **Navigator** and click **Create New Class**.

You can also right-click the project in the **Navigator** and choose **Create New Class** from the context menu or choose **Selected > Create New Class** from the menu.

**Figure 5–38 Add Class Dialog Box**



Use the following information to enter data in each field on the Add Class dialog box:

Field	Description
Package Name	Choose an existing package or enter a new package name. If blank, TopLink Workbench uses the default package name.
New Class Name	Enter a class name. The <b>New Class Name</b> must be unique within the package.

For more information on using TopLink Workbench to edit classes, see [Section 5.7.2, "How to Configure Classes"](#).

## 5.7.2 How to Configure Classes

Oracle recommends that you develop your Java classes using an IDE, such as Oracle Oracle JDeveloper, and import these existing classes into TopLink Workbench (see [Section 5.7.3, "How to Import and Update Classes"](#))

However, it is sometimes convenient to create (see [Section 5.7.1, "How to Create Classes"](#)) and configure classes in TopLink Workbench: for example, when generating an object model from a database schema.

This section describes using TopLink Workbench to edit classes, including the following:

- [Configuring Class Information](#)

- [Configuring Class Modifiers](#)
- [Configuring Class Interfaces](#)
- [Adding Attributes](#)
- [Configuring Attribute Modifiers](#)
- [Configuring Attribute Type Declaration](#)
- [Configuring Attribute Accessing Methods](#)
- [Adding Methods](#)
- [Configuring Method Modifiers](#)
- [Configuring Method Return Type](#)
- [Configuring Method Parameters](#)

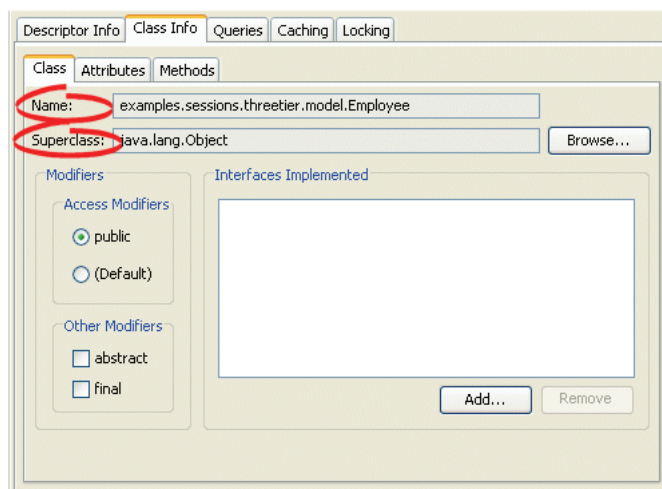
### 5.7.2.1 Configuring Class Information

This section includes information on [Using TopLink Workbench](#) to configure class information.

**5.7.2.1.1 Using TopLink Workbench** To configure class and superclass information, use this procedure:

1. Select a class in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Class Info** tab in the **Editor**.
3. Click the **Class** tab.

**Figure 5–39 Class Tab, Class Information Fields**



Use the following information to enter data in each field on the tab:

Field	Description
Name	The name of the class. This field is for display only.
Superclass	Click <b>Browse</b> and select a class and package that contains the class (that is, the superclass).



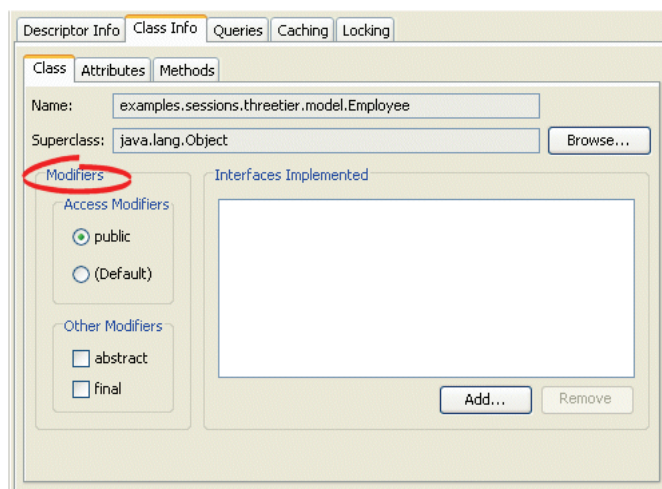
### 5.7.2.2 Configuring Class Modifiers

This section includes information on [Using TopLink Workbench](#) to configure class modifiers.

**5.7.2.2.1 Using TopLink Workbench** To configure class modifiers, use this procedure:

1. Select a class in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Class Info** tab in the **Editor**.
3. Click the **Class** tab.

**Figure 5–40 Class Tab, Class Modifiers Fields**



Use the following information to enter data in each field on the tab:

Field	Description
<b>Access Modifiers</b>	Use to specify whether the class is accessible publicly or not. Only public classes are visible to the Oracle TopLink Workbench.
<b>Other Modifiers</b>	Specify if the class is <b>Final</b> or <b>Abstract</b> , or both. Final classes are not included in the superclass selection lists for other classes to extend.

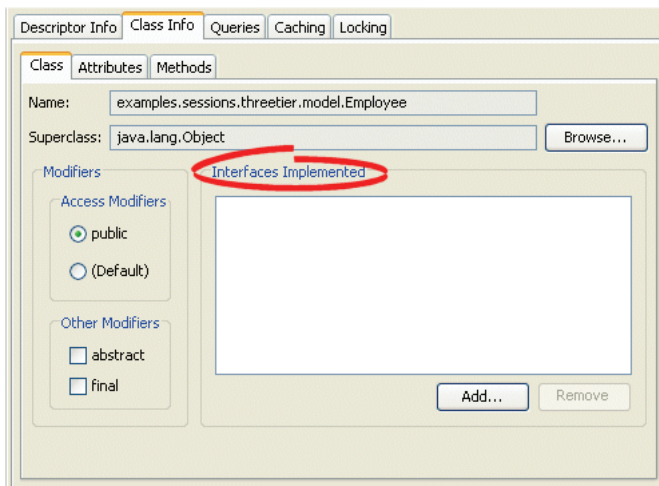
### 5.7.2.3 Configuring Class Interfaces

This section includes information on [Using TopLink Workbench](#) to specify the interfaces implemented by a class. You can choose any interface in the TopLink Workbench classpath (see [Section 117.3, "Configuring Project Classpath"](#)).

Although you may add interfaces to a project directly (see [Section 5.7.3, "How to Import and Update Classes"](#)), you do not need to do so in order to configure a class to implement an interface.

**5.7.2.3.1 Using TopLink Workbench** To implement interfaces, use this procedure:

1. Select a class in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Class Info** tab in the **Editor**.
3. Click the **Class** tab.

**Figure 5–41 Class Tab, Interfaces Implemented Fields**

Use the following information to enter data in the Interfaces Implemented field on the Class tab:

Field	Description
<b>Interfaces Implemented</b>	To add an interface, click <b>Add</b> . The Choose Class dialog box appears. In the dialog box, select the interface and package. To remove an interface, select the interface and click <b>Remove</b>

### 5.7.2.4 Adding Attributes

This section includes information on [Using TopLink Workbench](#) to add an attribute to a class.

**5.7.2.4.1 Using TopLink Workbench** To add a new attribute (field) to the descriptor, click **Add**.

To delete an existing attribute, select the attribute and click **Remove**.

To rename an existing attribute, select the attribute and click **Rename**.

The **Attributes** tab contains the following tabs:

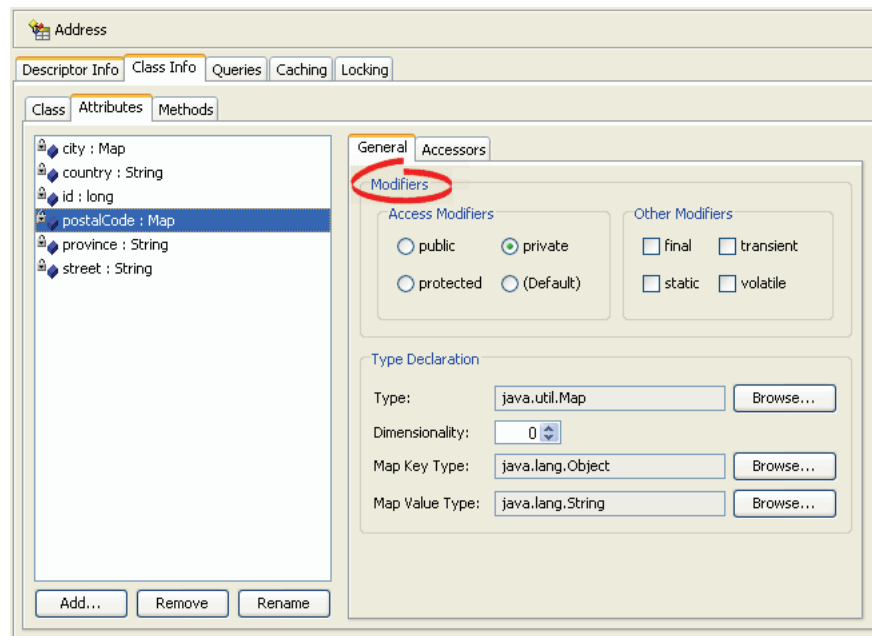
- General
- Accessors

### 5.7.2.5 Configuring Attribute Modifiers

This section includes information on [Using TopLink Workbench](#) to configure attribute modifiers.

**5.7.2.5.1 Using TopLink Workbench** To specify access modifiers, use this procedure:

1. Select a class in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Class Info** tab in the **Editor**.
3. Click the **Attributes** tab. The Attributes tab contains two sub-tabs.
4. Click the **General** tab.

**Figure 5–42** *Attributes Tab, Modifiers Fields*

Use the following information to enter data in the Modifiers fields on the Attributes tab:

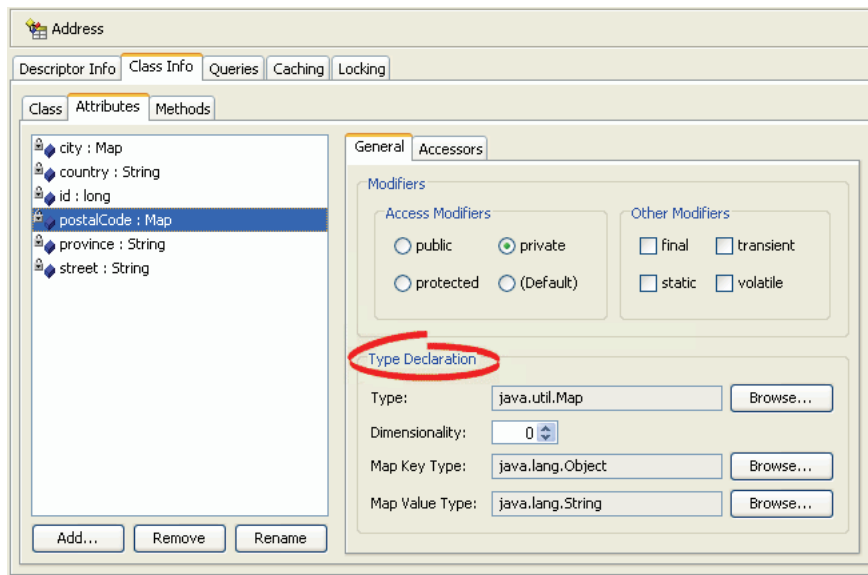
Field	Description
<b>Access Modifiers</b>	Specify how the attribute is accessible: <ul style="list-style-type: none"> <li> <b>Public</b></li> <li> <b>Protected</b>—only visible within its own package and subclasses.</li> <li> <b>Private</b>—not visible for subclasses</li> <li> <b>Default</b>—only visible within its own package</li> </ul>
<b>Other Modifiers</b>	Specify whether the attribute is <b>Final</b> , <b>Static</b> , <b>Transient</b> , or <b>Volatile</b> . Note: Selecting some modifiers may disable others.

### 5.7.2.6 Configuring Attribute Type Declaration

This section includes information on [Using TopLink Workbench](#) to configure attribute type declaration.

**5.7.2.6.1 Using TopLink Workbench** To specify attribute type declaration, use this procedure:

1. Select a class in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Class Info** tab in the **Editor**.
3. Click the **Attributes** tab. The Attributes tab contains two sub-tabs.
4. Click the **General** tab.

**Figure 5–43 Attributes Tab, Type Declaration Fields**

Use the following information to enter data in Type Declaration fields on the Attributes tab:

Field	Description
<b>Type</b>	Click <b>Browse</b> and select a class and package for the attribute.
<b>Dimensionality</b>	Specify the length of an array. This field applies only if <b>Type</b> is an array.
<b>Value Type</b>	Click <b>Browse</b> and select a class and package for the attribute. This field applies for <code>ValueHolderInterface</code> types only.
<b>Map Key Type</b>	Click <b>Browse</b> and select a class and package for the attribute. This field applies for <code>Map</code> types only.
<b>Map Value Type</b>	Click <b>Browse</b> and select a class and package for the attribute. This field applies for <code>Map</code> types only.
<b>Element Type</b>	Click <b>Browse</b> and select a class and package for the attribute. This field applies for <code>List</code> types only.

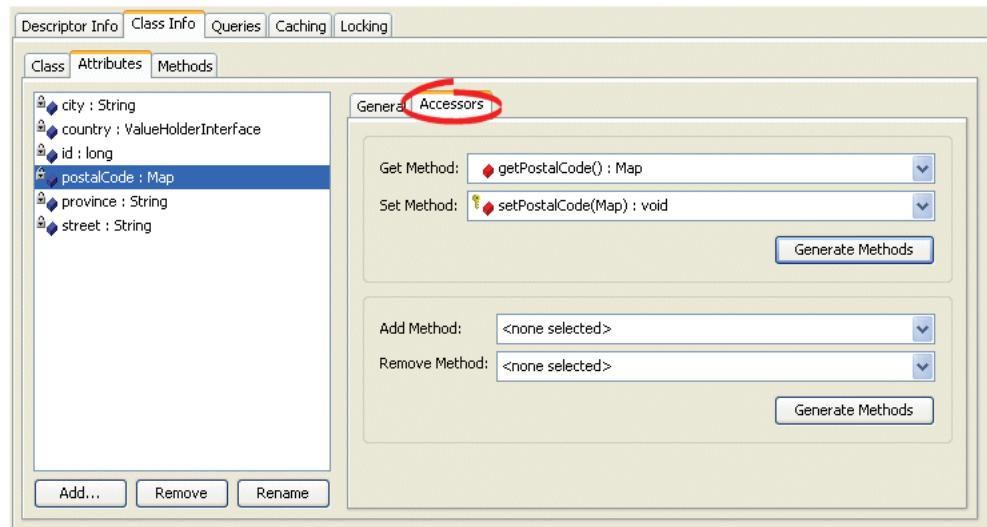
### 5.7.2.7 Configuring Attribute Accessing Methods

This section includes information on [Using TopLink Workbench](#) to configure attribute accessing methods. If you change an attribute and regenerate the accessing methods, TopLink *does not* remove any previously generated methods.

**5.7.2.7.1 Using TopLink Workbench** To specify attribute accessing methods, use this procedure:

1. Select a class in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Class Info** tab in the **Editor**.
3. Click the **Attributes** tab. The Attributes tab contains two sub-tabs.
4. Click the **Accessors** tab.

Figure 5–44 Attributes Tab, Accessors Fields



Use the following information to complete the Accessors fields on the Attributes tab:

Field	Description
<b>Get Method</b>	Choose the <code>get</code> method for the attribute. This field applies for non- <code>Collection</code> types only.
<b>Set Method</b>	Choose the <code>set</code> method for the attribute. This field applies for non- <code>Collection</code> types only.
<b>Add Method</b>	Choose the <code>add</code> method for the attribute. This field applies for <code>List</code> and <code>Map</code> types only.
<b>Remove Method</b>	Choose the <code>remove</code> method for the attribute. This field applies for <code>List</code> and <code>Map</code> types only.
<b>Value Holder Get Method</b>	Choose the method used to return the <code>ValueHolderInterface</code> type. This field applies for <code>ValueHolderInterface</code> types only.
<b>Value Holder Set Method</b>	Choose the method used to set the <code>ValueHolderInterface</code> type. This field applies for <code>ValueHolderInterface</code> types only.
<b>Value Get Method</b>	Choose the method used to return the actual value. This field applies for <code>ValueHolderInterface</code> types only.
<b>Value Set Method</b>	Choose the method used to set the actual value. This field applies for <code>ValueHolderInterface</code> types only.

### 5.7.2.8 Adding Methods

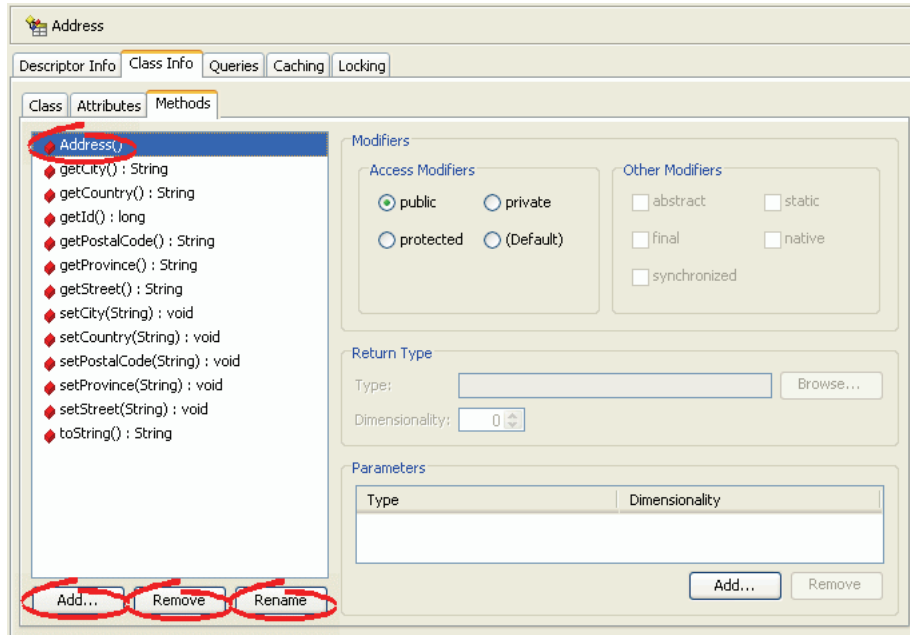
This section includes information on [Using TopLink Workbench](#) to add a method to a class.

**5.7.2.8.1 Using TopLink Workbench** To add or remove methods, use this procedure:

1. Select a class in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Class Info** tab in the **Editor**.

3. Click the **Methods** tab.

**Figure 5–45 Class Info–Methods Tab**



To add a new method to the descriptor, click **Add**.

To delete an existing method, select the method and click **Remove**.

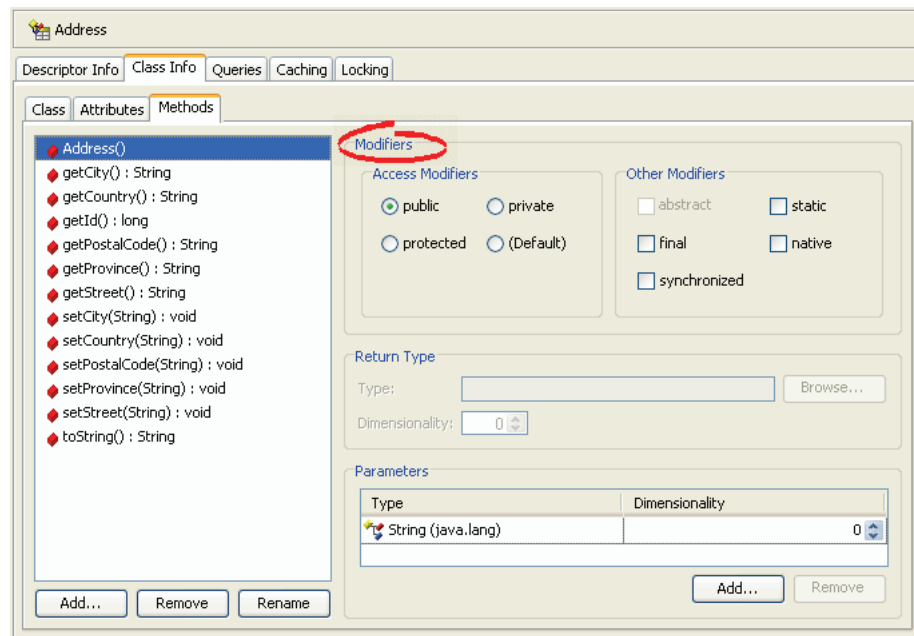
To rename an existing method, select the method and click **Rename**.

### 5.7.2.9 Configuring Method Modifiers








This section includes information on [Using TopLink Workbench](#) to configure method modifiers.

**5.7.2.9.1 Using TopLink Workbench** To specify access modifiers, use this procedure:

1. Select a class in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Class Info** tab in the **Editor**.
3. Click the **Methods** tab.

**Figure 5–46 Methods Tab, Modifiers Fields**

Use the following information to enter data in Modifiers fields on the Methods tab:

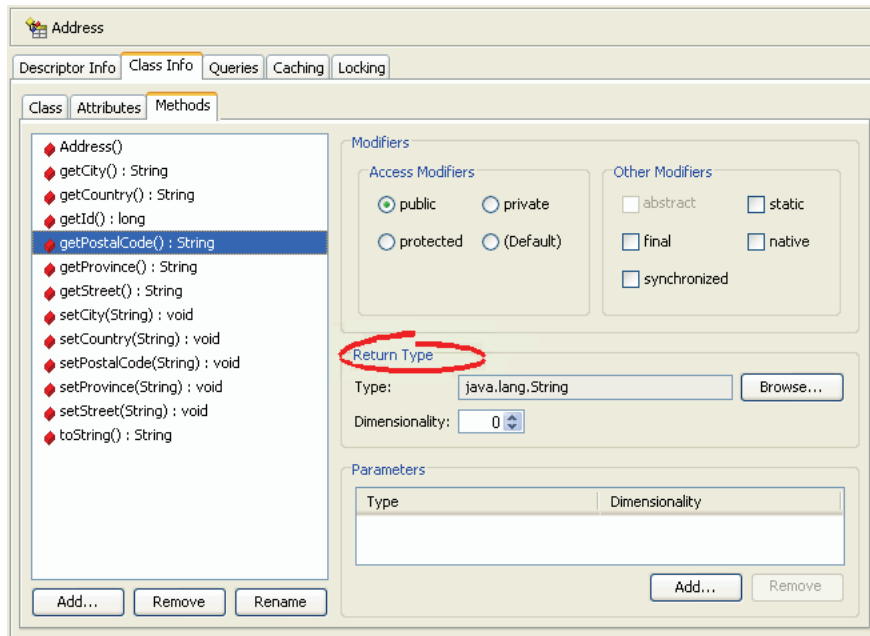
Field	Description
<b>Access Modifiers</b>	Specify how the method can be accessed: <ul style="list-style-type: none"> <li> <b>Public</b></li> <li>  <b>Protected</b>—only visible within its own package and subclasses.</li> <li>  <b>Private</b>—not visible for subclasses.</li> <li>  <b>Default</b>—only visible within its own package.</li> </ul>
<b>Other Modifiers</b>	Specify whether the method is <b>Abstract</b> , <b>Final</b> , <b>Synchronized</b> , <b>Static</b> , or <b>Native</b> . Note: Selecting some modifiers may disable others.

### 5.7.2.10 Configuring Method Return Type

This section includes information on [Using TopLink Workbench](#) to configure method return type.

**5.7.2.10.1 Using TopLink Workbench** To specify method return type, use this procedure:

1. Select a class in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Class Info** tab in the **Editor**.
3. Click the **Methods** tab.

**Figure 5–47** *Methods Tab, Return Type Fields*

Use the following information to enter data in Return Type fields on the Methods tab:

Field	Description
<b>Type</b>	Click <b>Browse</b> and select a class and package for the method.
<b>Dimensionality</b>	Specify the length of an array. This field applies only if <b>Type</b> is an array.

### 5.7.2.11 Configuring Method Parameters

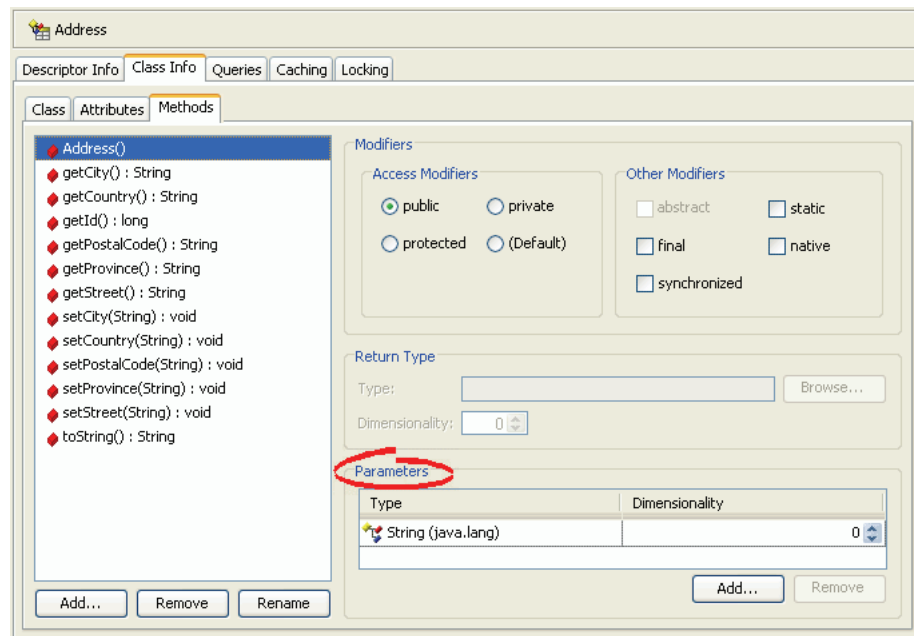
This section includes information on [Using TopLink Workbench](#) to configure method parameters.

**5.7.2.11.1 Using TopLink Workbench** To specify additional method parameters, use this procedure:

1. Select a class in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Class Info** tab in the **Editor**.
3. Click the **Methods** tab.



Figure 5–48 Methods Tab, Method Parameters Fields



Use the following information to enter data in Parameters fields on the Methods tab:

Field	Description
Type	Click <b>Browse</b> and select a class and package for the method.
Dimensionality	Specify the length of an array. This field applies only if <b>Type</b> is an array.

### 5.7.3 How to Import and Update Classes

This section includes information on [Importing and Updating Classes Using TopLink Workbench](#) to import and update Java classes.

You can import Java classes and interfaces created in any IDE.

You can import any class on the system classpath or project classpath.

If a class exists on both the system classpath and the project classpath, TopLink Workbench will update the class from the system classpath. To update or refresh from the project classpath, remove the class from the system classpath and restart TopLink Workbench.

For more information, see [Section 117.3, "Configuring Project Classpath"](#).

#### 5.7.3.1 Importing and Updating Classes Using TopLink Workbench

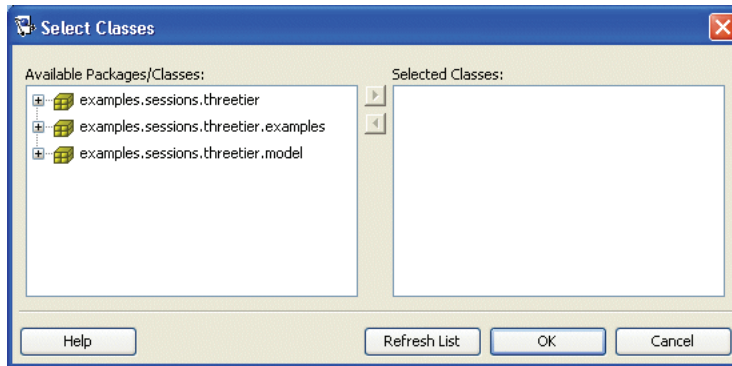
Use this procedure to update or refresh the classes in the TopLink Workbench project.

1. Define the available classes and packages for the project on the **General** tab. See [Section 117.3, "Configuring Project Classpath"](#) for information on classes and packages.



2. Click **Add or Refresh Class**. The Select Classes dialog box appears.

You can also update the classes by choosing **Selected > Add or Refresh Classes** from the menu.

**Figure 5–49 Select Classes Dialog Box**

Select the packages or classes (or both) to import into the project and click **OK**. TopLink Workbench adds the new classes to your project in the **Navigator**.

By default, TopLink Workbench creates the following descriptor types for each package and class (depending on your project type):

- Relational projects—Relational class descriptors (see [Section 22.2.1.1, "Creating Relational Class Descriptors"](#))
- EIS projects—EIS composite descriptors (see [Section 75.2.1.2, "EIS Composite Descriptors"](#))
- XML projects—XML descriptors (see [Section 50.1.1.1, "Composite Descriptors in XML Projects"](#))

See [Chapter 118, "Creating a Descriptor"](#) for more information.

---

**Note:** If the class exists on both the system classpath and the project classpath, TopLink Workbench will update the class from the system classpath. To update or refresh from the project classpath, remove the class from the system classpath and restart TopLink Workbench.

---

**To Remove a Class from a Project, do the following:**



Select the descriptor and click **Remove**, or choose **Selected > Remove** from the menu.

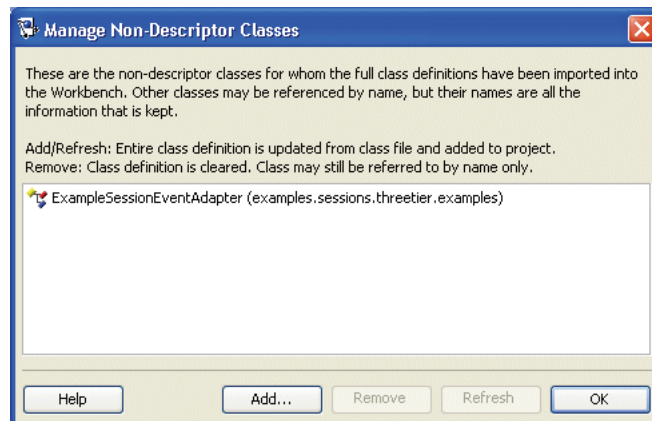
### 5.7.4 How to Manage Nondesoriptor Classes

Some of the mappings in your TopLink project may reference classes that do not have TopLink descriptors or are not included in the project.

To add, remove, or refresh Java classes that do not have TopLink descriptors, use this procedure:

From the menu, select **Workbench > Manage Non-Descriptor Classes**. The Manage Non-Descriptor Classes dialog box appears.

You can access the dialog box by right-clicking the TopLink project icon in the **Navigator** and selecting **Manage Non-Descriptor Classes** from the context menu.

**Figure 5–50 Manage Non-Descriptor Classes Dialog Box**

Select one of the following options:

- To add new classes, click **Add**. The Select Classes dialog box appears.
- To add new classes, click **Add**. The Select Classes dialog box appears (see [Figure 5–49, "Select Classes Dialog Box"](#)).

Only classes that have been added to the project's class path can be added as nondescriptor classes. See [Section 117.3, "Configuring Project Classpath"](#) for more information.

- To delete an existing class, select the class and click **Remove**.
- To refresh the classes (for example, if you edited the classes in an IDE), click **Refresh**.

## 5.7.5 How to Rename Packages

When you add classes to a project, TopLink Workbench shows the classes contained in the package to which they belong (see [Section 5.3.3, "How to Use the Navigator"](#)).

You can use TopLink Workbench to change the package statements in all the Java classes of a selected package (to move the all the classes contained by the selected package to a new package). This is useful if you are refactoring an existing TopLink Workbench project.

---

**Note:** The TopLink Workbench package rename feature is not intended for migrating projects from older versions of TopLink: for this, you must still use the TopLink Package Renamer. The Package Renamer updates import statements for TopLink classes—it does not change the package statements in user application classes.

For information on the TopLink Package Renamer, refer to *Oracle TopLink Release Notes*.

---

For more information on using TopLink Workbench to edit classes, see [Section 5.7.2, "How to Configure Classes"](#).

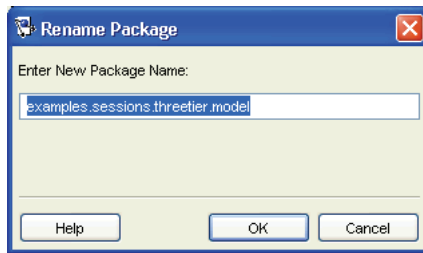
### 5.7.5.1 Renaming Packages Using TopLink Workbench

To change the package of an existing class in TopLink Workbench, use this procedure:

1. Right-click the package in the **Navigator** and select **Rename**.

You can also select the package and choose **Selected > Rename** from the menu.

**Figure 5–51 Rename Package Dialog Box**



Enter the package name and click **OK**. TopLink Workbench changes the name of the package in the Navigator window.

For more information on using TopLink Workbench to edit classes, see [Section 5.7.2, "How to Configure Classes"](#).

## 5.8 Integrating TopLink Workbench with Apache Ant

If you use the Apache Ant Java-based build tool, you can use the Ant task and type definitions that TopLink provides to invoke certain TopLink Workbench functions from an Ant build file. Using these tasks, you can integrate TopLink Workbench into your automated build process.

This section describes the following:

- [How to Configure Ant to Use TopLink Workbench Tasks](#)
- [What You May Need to Know About TopLink Workbench Ant Task API](#)
- [How to Create TopLink Workbench Ant Tasks](#)

For more information about Ant, see <http://ant.apache.org/manual/>.

### 5.8.1 How to Configure Ant to Use TopLink Workbench Tasks

Before you can use TopLink Workbench tasks in your Ant build files, you must consider their library dependencies (see [Section 5.8.1.1, "Creating Library Dependencies"](#)).

To declare TopLink Workbench tasks in your Ant `build.xml` file, declare them directly (see [Section 5.8.1.2, "Declaring TopLink Workbench Tasks"](#)).

#### 5.8.1.1 Creating Library Dependencies

In addition to the Ant library dependencies (see <http://ant.apache.org/manual/install.html#librarydependencies>), [Table 5–4](#) lists the TopLink-specific JAR files that must be in your Ant classpath.

**Table 5–4 TopLink Workbench Ant Task Library Dependencies**

JAR Name	Needed For...	Available At...
toplinkmw.jar	TopLink Workbench Ant task and type definitions.	<TOPLINK_HOME>/utils/workbench/jlib

### 5.8.1.2 Declaring TopLink Workbench Tasks

After you declare the TopLink Workbench task definitions (see [Table 5–6](#)) and data definitions (see [Table 5–4](#)) in the `toplink-ant-lib.xml` file (see [Example 5–5](#)), you can use a TopLink Workbench task in a `build.xml` file, as [Example 5–6](#) shows:

#### **Example 5–5 Declaring TopLink Workbench Ant Task and Data Types in a `toplink-ant-lib.xml` File**

```
<?xml version="1.0"?>
<antlib>
  <taskdef name="mappings.export"
    classname="oracle.toplink.workbench.ant.taskdefs.ExportDeploymentXMLTask" />

  <taskdef name="mappings.validate"
    classname="oracle.toplink.workbench.ant.taskdefs.MappingsValidateTask" />

  <taskdef name="session.validate"
    classname="oracle.toplink.workbench.ant.taskdefs.SessionValidateTask" />

  <typedef name="ignoreerror"
    classname="oracle.toplink.workbench.ant.typedefs.IgnoreError" />

  <typedef name="ignoreerrorset"
    classname="oracle.toplink.workbench.ant.typedefs.IgnoreErrorSet" />

  <typedef name="loginspec"
    classname="oracle.toplink.workbench.ant.typedefs.LoginSpec" />
</antlib>
```

#### **Example 5–6 Specifying the `toplink-ant-lib.xml` File in the `build.xml` File**

```
<project name="MyBuild" default="validate.session" basedir="." xmlns:toplink="toplinklib">
  <typedef file = "toplink-ant-lib.xml" classpathref = "mw.classpath" uri = "toplinklib" />
  ...
</project>
```

## 5.8.2 What You May Need to Know About TopLink Workbench Ant Task API

[Table 5–5](#) lists the TopLink Workbench Ant task definitions that TopLink provides.

**Table 5–5 TopLink Workbench Ant Task Definitions**

Task Name	TopLink Class
<a href="#">How to Create the <code>mappings.validate</code> Task</a>	<code>oracle.toplink.workbench.ant.taskdefs.MappingsValidateTask</code>
<a href="#">How to Create the <code>session.validate</code> Task</a>	<code>oracle.toplink.workbench.ant.taskdefs.SessionValidateTask</code>
<a href="#">How to Create the <code>mappings.export</code> Task</a>	<code>oracle.toplink.workbench.ant.taskdefs.ExportDeploymentXMLTask</code>

[Table 5–6](#) lists the TopLink Workbench Ant type definitions that TopLink provides.

**Table 5–6 TopLink Workbench Ant Type Definitions**

Type Name	TopLink Class
<a href="#">How to Create the <code>ignoreerror</code> Task</a>	<code>oracle.toplink.workbench.ant.typedefs.IgnoreError</code>

**Table 5–6 (Cont.) TopLink Workbench Ant Type Definitions**

Type Name	TopLink Class
<a href="#">How to Create the ignoreerrorset Task</a>	oracle.toplink.workbench.ant.typedefs.IgnoreErrorSet
<a href="#">How to Create the loginspec Task</a>	oracle.toplink.workbench.ant.typedefs.LoginSpec

### 5.8.3 How to Create TopLink Workbench Ant Tasks

[Example 5–7](#) shows a typical Ant build.xml file that declares and uses the TopLink Workbench Ant task and type definitions.

**Example 5–7 Example Ant Build File with TopLink Workbench Ant Tasks**

```
<project name="MyBuild" default="validate.session" basedir="." xmlns:toplink="toplinklib">
  <!-- ===== -->
  <!-- Properties -->
  <!-- ===== -->
  <target name="init">
    <property file="build.properties"/>

    <property name = "toplink.mwp.dir" value = "${basedir}/mw"/>
    <property name = "toplink.sessions.dir" value = "${basedir}/config"/>
    <property name = " myProject.classes" value = "${basedir}/classes" />

    <path id = "database.classpath">
      <pathelement path = "${toplink.home}/mdoules/oracle.toplink_
11.1.1jlib/OracleThinJDBC.jar"/>
    </path>
    <path id = "toplink.classpath">
      <pathelement path = "${toplink.home}/modules/oracle.toplink_11.1.1/toplink.jar"/>
      <pathelement path = "${toplink.home}/modules/oracle.toplink_
11.1.1lib/java/api/ejb.jar"/>
      <pathelement path = "${toplink.home}/modules/xmlparserv2.jar"/>
      <pathelement path = "${toplink.home}/modules/oracle.toplink_
11.1.1/jlib/antlr.jar"/>
    </path>
    <path id = "mw.classpath">
      <pathelement path = "${toplink.home}/modules/oracle.toplink_
11.1.1/jlib/tlmwcore.jar"/>
      <pathelement path = "${toplink.home}/modules/oracle.toplink_
11.1.1/jlib/toplinkmw.jar"/>
    </path>
    <path id = "mwplatforms.classpath">
      <pathelement path = "${toplink.home}/config"/>
    </path>

    <typedef file = "toplink-ant-lib.xml"

      classpathref = "mw.classpath"
      uri = "toplinklib" />
  </target>
  <!-- ===== -->
  <!-- Define task parameter -->
  <!-- ===== -->
  <target name="parameter.definition" depends="init">
    <toplink:ignoreerrorset id = "ignoreErrors">
      <toplink:ignoreerror code = "0233" />
    </toplink:ignoreerrorset>

    <toplink:loginspec id = "loginSpec"
      url = "jdbc:cloudscape:stagedb;create=true"
      driverclass = "COM.cloudscape.core.JDBCdriver"
  </target>
```

```

        user = "scott"
        password="tiger" />
</target>
<!-- ===== -->
<!-- Validate the MW Project -->
<!-- ===== -->
<target name="validate.project" depends="parameter.definition">

    <toplink:mappings.validate
        projectfile = "${toplink.mwp.dir}/myProject.mwp"
        reportfile = "${toplink.mwp.dir}/problem-report.html"
        reportformat = "html"
        property = "mw-valid"
        classpathref = "mwplatforms.classpath" >

        <toplink:classpath refid = "mw.classpath" />
        <toplink:classpath refid = "toplink.classpath" />

        <toplink:ignoreerrorset refid = "ignoreErrors"/>

    </toplink:mappings.validate>
</target>
<!-- ===== -->
<!-- TopLink deployment descriptor XML generation -->
<!-- ===== -->
<target name="export.deployment" depends="validate.project" if="mw-valid">

    <toplink:mappings.export
        projectfile = "${toplink.mwp.dir}/myProject.mwp"
        deploymentfile = "${toplink.sessions.dir}/sessions.xml"
        property = "export-completed"
        failonerror = "true"
        classpathref = "toplink.classpath">

        <toplink:classpath refid = "mw.classpath" />
        <toplink:classpath refid = "mwplatforms.classpath" />

        <toplink:ignoreerrorset refid = "ignoreErrors"/>
        <toplink:loginspec refid = "loginSpec" />
    </toplink:mappings.export>
</target>
<!-- ===== -->
<!-- TopLink Session Validate -->
<!-- ===== -->
<target name="validate.session" depends="export.deployment" if="export-completed">

    <toplink:session.validate
        sessionsfile = "${toplink.sessions.dir}/sessions.xml"
        sessionname = "ThreeTierEmployee"
        property = "session-valid"
        classpathref = "toplink.classpath"
        classpath = "${ myProject.classes}" >

        <toplink:classpath refid = "mw.classpath" />
        <toplink:classpath refid = " database.classpath" />

        <toplink:loginspec refid = "loginSpec" />
    </toplink:session.validate>
</target>
</project>

```

## 5.8.4 How to Create the mappings.validate Task

The `mappings.validate` task is a testing task that you use to list of all the problems in a TopLink Workbench project (`.mwp`) file.

This task lets you do the following:

- log all the problems to a file in text or HTML format;
- set an Ant property to indicate that the TopLink Workbench project is valid (has no errors).

### 5.8.4.1 Using Parameters

**Table 5–7** *mappings.validate* Task Parameters

Attribute	Description	Required
projectfile	Fully qualified TopLink Workbench projects file name (.mwp).	Yes
reportfile	Fully qualified file name to which to write the output.	No
reportformat	The format of the generated output. Must be <code>html</code> or <code>text</code> .	No—default to <code>text</code> .
classpath	Project classpath.	No
classpathref	Reference to a path defined elsewhere.	No
property	The name of the property to set (true if there is no problem).	No

### 5.8.4.2 Specifying Parameters Specified as Nested Elements

You can specify the following parameters as nested elements of this task:

- `classpath`
- `ignoreerror` (see [Section 5.8.8, "How to Create the ignoreerror Task"](#))
- `ignoreerrorset` (see [Section 5.8.9, "How to Create the ignoreerrorset Task"](#))

### 5.8.4.3 Examples

[Example 5–8](#) shows a typical `mappings.validate` task.

#### **Example 5–8** A *mappings.validate* Task

```
<toplink:mappings.validate
  projectfile = "${toplink.mwp.dir}/myProject.mwp"
  reportfile = "${toplink.mwp.dir}/problem-report.html"
  reportformat = "html"
  property = "mw-valid"
  classpath = "${mwplatforms.classpath}" >

  <toplink:classpath refid = "mw.classpath" />
  <toplink:classpath refid = "toplink.classpath" />

  <toplink:ignoreerrorset refid = "ignoreErrors"/>
  <toplink:ignoreerror code = "0555" />
</toplink:mappings.validate>
```

## 5.8.5 How to Create the session.validate Task

The `session.validate` task is a testing task that you use to test your TopLink deployment XML by running TopLink.

This task provides the ability to do the following:

- specify the test type using a nested element;
- set an Ant property to indicate that the TopLink Workbench project is valid (has no errors).



### 5.8.5.1 Using Parameters

**Table 5–8** *session.validate Task Parameters*

Attribute	Description	Required
sessionsfile	Fully qualified <code>sessions.xml</code> file.	No—default to <code>sessions.xml</code> and to <code>classpath</code> .
sessionname	Name of the session to test.	Yes
classpath	Project classpath.	No
classpathref	Reference to a path defined elsewhere.	No
property	The name of the property to set (true if valid).	No

### 5.8.5.2 Specifying Parameters Specified as Nested Elements

You can specify the following parameters as nested elements of this task:

- `classpath`;
- `loginspec` (see [Section 5.8.10, "How to Create the loginspec Task"](#)).

### 5.8.5.3 Examples

[Example 5–9](#) shows a typical `session.validate` task.

**Example 5–9** *A session.validate Task*

```
<toplink:session.validate
  sessionsfile = "${toplink.sessions.dir}/sessions.xml"
  sessionname = "ThreeTierEmployee"
  property = "session-valid"
  classpathref = "toplink.classpath"
  classpath = "${ myProject.classes}" >

  <toplink:classpath refid = "mw.classpath" />
  <toplink:classpath refid = " database.classpath" />

  <toplink:loginspec refid = "loginSpec" />
</toplink:session.validate>
```

## 5.8.6 How to Create the mappings.export Task

The `mappings.export` task is a generation task that you use to generate a TopLink deployment XML file for a given TopLink Workbench project (`.mwp`). The `mappings.export` task executes a `mappings.validate` (see [Section 5.8.4, "How to Create the mappings.validate Task"](#)) before executing. A `BuildException` is thrown if validation fails.

This task provides the ability to override the TopLink Workbench project database login information.

### 5.8.6.1 Using Parameters

**Table 5–9** *mappings.export Task Parameters*

Attribute	Description	Required
projectfile	Fully qualified TopLink Workbench projects file name ( <code>.mwp</code> ).	Yes

**Table 5–9 (Cont.) mappings.export Task Parameters**

Attribute	Description	Required
deploymentfile	Fully qualified TopLink project deployment file name (.xml).	No—default to the name specified in the TopLink Workbench project (.mwp).
ejbjarxmdir	The directory that contains the ejb-jar.xml file (only applicable to Java EE project).	No—default to the directory specified in the TopLink Workbench project (.mwp).
classpath	Project classpath.	No
classpathref	Reference to a path defined elsewhere.	No
failonerror	Indicates whether the build will continue even if there are export errors; defaults to true.	No
property	The name of the property to set (true if export completed successfully).	No

### 5.8.6.2 Specifying Parameters Specified as Nested Elements

You can specify the following parameters as nested elements of this task:

- classpath;
- loginspec (see [Section 5.8.10, "How to Create the loginspec Task"](#));
- ignoreerror (see [Section 5.8.8, "How to Create the ignoreerror Task"](#));
- ignoreerrorset (see [Section 5.8.9, "How to Create the ignoreerrorset Task"](#)).

### 5.8.6.3 Examples

[Example 5–10](#) shows a typical mappings.export task.

#### Example 5–10 A mappings.export Task

```
<toplink:mappings.export
  projectfile = "${toplink.mwp.dir}/myProject.mwp"
  deploymentfile = "${toplink.sessions.dir}/sessions.xml"
  property = "export-completed"
  failonerror = "true"
  classpathref = "toplink.classpath">

  <toplink:classpath refid = "mw.classpath" />
  <toplink:classpath refid = "mwplatforms.classpath" />

  <toplink:ignoreerrorset refid = "ignoreErrors"/>
  <toplink:ignoreerror code = "0545" />
  <toplink:loginspec
    url = "jdbc:cloudscape:stagedb;create=true"
    driverclass = "COM.cloudscape.core.JDBCdriver"
    user = "scott"
    password="tiger" />
</toplink:mappings.export>
```

## 5.8.7 How to Create the classpath Task

Use the classpath element to define the Java classpath necessary to run a task. For more information, see <http://ant.apache.org/manual/using.html#path>.

### 5.8.7.1 Using Parameters

**Table 5–10** *classpath Element Parameters*

Attribute	Description	Required
location	Specifies a single file or directory relative to the project's base directory (or an absolute filename).	No
path	Specifies one or multiple files or directories separated by a colon or semicolon.	No
refid	Reference to a path defined elsewhere.	No

### 5.8.7.2 Specifying Parameters Specified as Nested Elements

You can specify the following parameters as nested elements of this task:

- `pathelement`
- `fileset`
- `dirset`
- `filelist`

### 5.8.7.3 Examples

[Example 5–11](#) shows a typical `classpath` element.

**Example 5–11** *A classpath Element*

```
<classpath>
  <pathelement path="{classpath}"/>
    <fileset dir="lib">
      <include name="**/*.jar"/>
    </fileset>
    <pathelement location="classes"/>
      <dirset dir="{build.dir}">
        <include name="apps/**/classes"/>
        <exclude name="apps/**/*Test*"/>
      </dirset>
      <filelist refid="third-party_jars"/>
    </classpath>
```

## 5.8.8 How to Create the ignoreerror Task

Use the `ignoreerror` element to instruct a TopLink Ant task to ignore a specific TopLink Foundation Library or TopLink Workbench (see [Section A.3, "TopLink Workbench Error Reference"](#)) run-time error code.

To instruct a TopLink Ant task to ignore multiple error codes, consider using an `ignoreerrorset` element (see [Section 5.8.9, "How to Create the ignoreerrorset Task"](#)).

### 5.8.8.1 Using Parameters

**Table 5–11** *ignoreerror Element Parameters*

Attribute	Description	Required
code	Error code of the problem to ignore.	Yes

### 5.8.8.2 Specifying Parameters Specified as Nested Elements

You cannot specify parameters as nested elements of this element.

### 5.8.8.3 Examples

[Example 5–12](#) shows a typical `ignoreerror` element. This element instructs a `mappings.export` task to ignore TopLink Workbench error code 0545.

#### Example 5–12 An ignoreerror Element

```
<toplink:mappings.export
  projectfile = "${toplink.mwp.dir}/myProject.mwp"
  deploymentfile = "${toplink.sessions.dir}/sessions.xml"
  classpathref = "toplink.classpath">

  <toplink:classpath refid = "mw.classpath" />
  <toplink:classpath refid = "mwplatforms.classpath" />
  <toplink:ignoreerror code = "0545" />

</toplink:mappings.export>
```

## 5.8.9 How to Create the ignoreerrorset Task

Use the `ignoreerrorset` element to instruct a TopLink Ant task to ignore any of multiple TopLink Foundation Library or TopLink Workbench (see [Section A.3](#), "TopLink Workbench Error Reference") run-time error codes.

### 5.8.9.1 Using Parameters

**Table 5–12** *ignoreerrorset Element Parameters*

Attribute	Description	Required
id	Unique identifier for this type instance, can be used to reference this type in scripts.	No
refid	Reference to a <code>ignoreerrorset</code> defined elsewhere.	No

### 5.8.9.2 Specifying Parameters Specified as Nested Elements

You can specify the following parameter as nested elements of this element:

- `ignoreerror` (see [Section 5.8.8](#), "How to Create the ignoreerror Task")

### 5.8.9.3 Examples

[Example 5–13](#) shows a typical `ignoreerrorset` element. This element instructs a `mappings.export` task to ignore all of TopLink Workbench error codes 0402 and 0570. Note that the `mappings.export` task also uses an explicitly `ignoreerror` element: this means that the `mappings.export` task will ignore all of error codes 0402, 0570, and 0545.

#### Example 5–13 An ignoreerrorset Element

```
<toplink:ignoreerrorset id = "ignoreErrors">
  <toplink:ignoreerror code = "0402" />
  <toplink:ignoreerror code = "0570" />
</toplink:ignoreerrorset>
...
<toplink:mappings.export
```

```

projectfile = "${toplink.mwp.dir}/myProject.mwp"
deploymentfile = "${toplink.sessions.dir}/sessions.xml"
classpathref = "toplink.classpath">

<toplink:classpath refid = "mw.classpath" />
<toplink:classpath refid = "mwplatforms.classpath" />
<toplink:ignoreerrorset refid = "ignoreErrors"/>
<toplink:ignoreerror code = "0545" />

</toplink:mappings.export>

```

## 5.8.10 How to Create the loginspec Task

Use the `loginspec` element to instruct a TopLink Ant task to override the project database login information in a TopLink Workbench project. For more information, see [Chapter 96, "Introduction to Data Access"](#).

---

**Note:** You can only use this element with a relational project (see [Chapter 18, "Introduction to Relational Projects"](#)).

You cannot use this element with a Java EE project.

---

### 5.8.10.1 Using Parameters

**Table 5–13** *loginspec Element Parameters*

Attribute	Description	Required
id	Unique identifier for this type instance, can be used to reference this type in scripts.	No
refid	Reference to a <code>loginspec</code> defined elsewhere.	No
driverclass	Fully qualified class of the data source driver (see <a href="#">Section 98.3, "Configuring Database Login Connection Options"</a> ).	No—default to the class that the TopLink Workbench project specifies.
url	URL of the driver see <a href="#">Section 98.3, "Configuring Database Login Connection Options"</a> .	Yes
user	Login user name (see <a href="#">Section 97.2, "Configuring User Name and Password"</a> ).	No—default to the value that the TopLink Workbench project specifies
password	Login password (see <a href="#">Section 97.2, "Configuring User Name and Password"</a> ).	No—default to the value that the TopLink Workbench project specifies

### 5.8.10.2 Specifying Parameters Specified as Nested Elements

You cannot specify parameters as nested elements of this element.

### 5.8.10.3 Examples

[Example 5–14](#) shows a typical `loginspec` element.

**Example 5–14** *A loginspec Element*

```

<toplink:mappings.export
  projectfile = "${toplink.mwp.dir}/myProject.mwp"
  deploymentfile = "${toplink.sessions.dir}/sessions.xml"
  classpathref = "toplink.classpath">

```

```
<toplink:classpath refid = "mw.classpath" />
<toplink:classpath refid = "mwplatforms.classpath" />

<toplink:loginspec
  url = "jdbc:cloudscape:stagedb;create=true"
  driverclass = "COM.cloudscape.core.JDBCdriver"
  user = "scott"
  password="tiger" />
</toplink:mappings.export>
```

---

---

## Using the Schema Manager

The `SchemaManager` and its related classes provide API that you can use from a Java application to specify database tables in a generic format, and then create and modify them in a specific relational database. This decouples your TopLink project from a particular database schema while giving you a programmatic means of creating a database schema based on your TopLink project. For example, you can use the schema manager to recreate a production database in a nonproduction environment. This lets you build models of your existing databases, and modify and test them during development.

---

---

**Note:** You can also create database tables manually during development using Oracle JDeveloper TopLink editor (see [Chapter 4, "Using Oracle JDeveloper TopLink Editor"](#)) or TopLink Workbench (see [Section 5.5.1.2, "Creating New Tables"](#) and [Section 5.5.3.4, "Generating Tables on the Database"](#)).

---

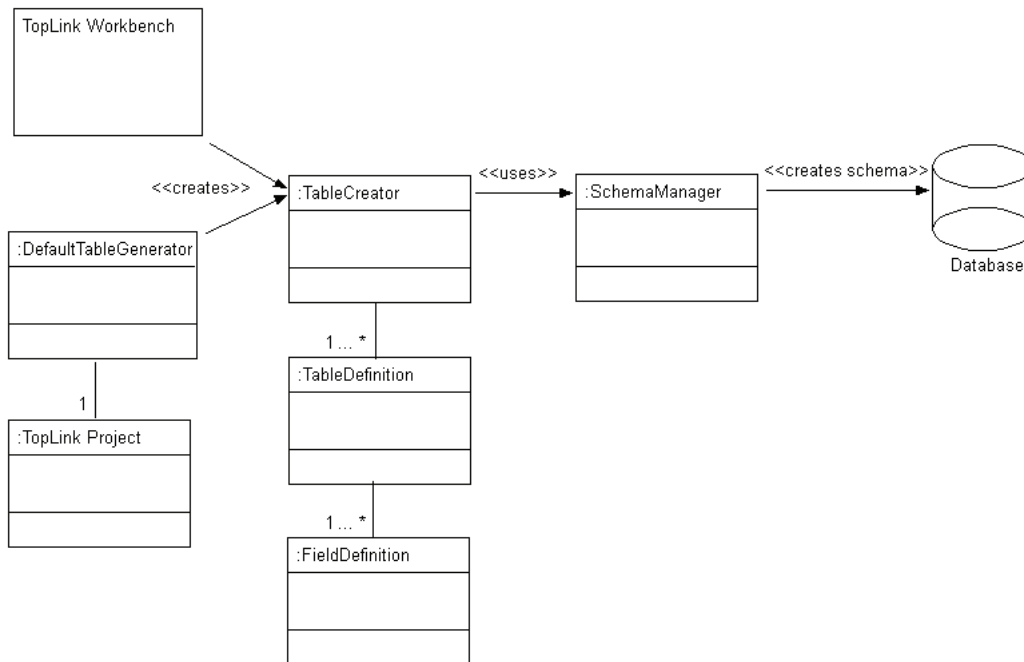
---

This chapter includes the following sections:

- [Introduction to the Schema Manager](#)
- [Creating a Table Creator](#)
- [Creating Tables with a Table Creator](#)
- [Creating Database Tables Automatically](#)

### 6.1 Introduction to the Schema Manager

[Figure 6–1](#) summarizes the important `SchemaManager` classes and the primary means of using them.

**Figure 6–1 SchemaManager Usage**

Although you can use the SchemaManager API directly, Oracle recommends that you create a TableCreator class and use its API (which, in turn, uses the SchemaManager).

You can automatically generate a TableCreator using the following:

- TopLink Workbench during development (see [Section 6.2.1, "How to Use TopLink Workbench During Development"](#))
- DefaultTableGenerator at run time (see [Section 6.2.2, "How to Use the Default Table Generator at Run Time"](#))

The TableCreator class owns one or more TableDefinition classes (one for each database table) and the TableDefinition class owns one or more FieldDefinition classes (one for each field).

The TableDefinition class lets you specify a database table schema in a generic format. At run time, TopLink uses the session associated with your TopLink project to determine the specific database type, and uses the generic schema to create the appropriate tables and fields for that database.

After creating a TableCreator class, you can use its API to create and drop tables (see [Section 6.3, "Creating Tables with a Table Creator"](#)). You can also configure TopLink to do this automatically (see [Section 6.4, "Creating Database Tables Automatically"](#)).

Because the schema manager uses Java types rather than database types, it is database-independent. However, because it does not account for database-specific optimizations, it is best-suited for development purposes rather than production. For more information on how the schema manager maps Java types to database types, see [Section 6.1.1, "How to Use Schema Manager Java and Database Type Conversion"](#).

Although the schema manager can handle the sequencing configuration that you specify in your TopLink project, if you are using sequencing with non-Oracle databases, there are some sequencing restrictions you should be aware of (see [Section 6.1.2, "How to Use Sequencing"](#)).



## 6.1.1 How to Use Schema Manager Java and Database Type Conversion

[Table 6–1](#) lists the Java type to database type conversions that the schema manager supports depending on the database platform your TopLink project uses. This list is specific to the schema manager and does not apply to mappings. TopLink automatically performs conversions between any database types within mappings.

**Table 6–1 Java and Database Field Type Conversion**

Java Type	Oracle	DB2	Sybase	MySQL	MS Access
<code>java.lang.Boolean</code>	NUMBER	SMALLINT	BIT default 0	TINYINT(1)	SHORT
<code>java.lang.Byte</code>	NUMBER	SMALLINT	SMALLINT	TINYINT	SHORT
<code>java.lang.Byte[]</code>	LONG RAW	BLOB	IMAGE	BLOB	LONGBINARY
<code>java.lang.Character</code>	CHAR	CHAR	CHAR	CHAR	TEXT
<code>java.lang.Character[]</code>	LONG	CLOB	TEXT	TEXT	LONGTEXT
<code>java.lang.Double</code>	NUMBER	FLOAT	FLOAT(32)	DOUBLE	DOUBLE
<code>java.lang.Float</code>	NUMBER	FLOAT	FLOAT(16)	FLOAT	DOUBLE
<code>java.lang.Integer</code>	NUMBER	INTEGER	INTEGER	INTEGER	LONG
<code>java.lang.Long</code>	NUMBER	INTEGER	NUMERIC	BIGINT	DOUBLE
<code>java.lang.Short</code>	NUMBER	SMALLINT	SMALLINT	SMALLINT	SHORT
<code>java.lang.String</code>	VARCHAR2	VARCHAR	VARCHAR	VARCHAR	TEXT
<code>java.math.BigDecimal</code>	NUMBER	DECIMAL	NUMERIC	DECIMAL	DOUBLE
<code>java.math.BigInteger</code>	NUMBER	DECIMAL	NUMERIC	BIGINT	DOUBLE
<code>java.sql.Date</code>	DATE	DATE	DATETIME	DATE	DATETIME
<code>java.sql.Time</code>	DATE	TIME	DATETIME	TIME	DATETIME
<code>java.sql.Timestamp</code>	DATE	TIMESTAMP	DATETIME	DATETIME	DATETIME

For more information about database platforms that TopLink supports, see [Section 96.1.3.1, "Database Platforms"](#).

## 6.1.2 How to Use Sequencing

If you generate a `TableCreator` class using TopLink Workbench (see [Section 6.2.1, "How to Use TopLink Workbench During Development"](#)), or `DefaultTableGenerator` (see [Section 6.2.2, "How to Use the Default Table Generator at Run Time"](#)), then sequencing configuration is included in your `TableCreator` according to your TopLink project configuration. In this case, when you use `TableCreator` method `createTables`, it does the following:

- Creates the sequence table as defined in the session `DatabaseLogin`.
- Creates or inserts sequences for each sequence name for all registered descriptors in the session.
- Creates the Oracle sequence object if you use Oracle native sequencing.

You can use advanced API to handle special cases like Sybase or Microsoft SQL Server native sequencing (see [Section 6.2.3, "How to Use Java to Create a Table Creator"](#)).

For more information about sequencing, see [Section 18.2, "Sequencing in Relational Projects"](#).

## 6.2 Creating a Table Creator

You can automatically generate a `TableCreator` using the following:

- Oracle JDeveloper (during development).
- TopLink Workbench (during development) (see [Section 6.2.1, "How to Use TopLink Workbench During Development"](#)).
- `DefaultTableGenerator` (at run time) (see [Section 6.2.2, "How to Use the Default Table Generator at Run Time"](#)).

After creating a `TableCreator` class, you can use its API to create and drop tables (see [Section 6.3, "Creating Tables with a Table Creator"](#)).

### 6.2.1 How to Use TopLink Workbench During Development

To create a `TableCreator` class that you can use in a Java application to recreate a database schema using the `SchemaManager`, use this procedure:

1. Right-click the project in the **Navigator** and choose **Export > Table Creator Java Source** from the context menu. The Table Creator dialog box appears.

You can also select the table and choose **Selected > Export > Table Creator Java Source** from the menu.

2. Enter a name for the table creator class and click **OK**. The Save As dialog box appears.
3. Choose a location for your table creator class and click **OK**. TopLink Workbench exports the table creator Java class to the location you specify.

### 6.2.2 How to Use the Default Table Generator at Run Time

To create a `TableCreator` class in Java using the `DefaultTableGenerator`, use this procedure:

1. Create an instance of `DefaultTableGenerator`, passing in an instance of your TopLink project:

```
DefaultTableGenerator myDefTblGen = new DefaultTableGenerator(toplinkProject);
```

2. Create a `TableCreator` instance:

- If you want a `TableCreator` that can support any session, use:

```
TableCreator myTblCre = myDefTblGen.generateDefaultTableCreator();
```

- If you want a `TableCreator` customized for a specific TopLink session, use:

```
TableCreator myTblCre =  
myDefTblGen.generateFilteredDefaultTableCreator(toplinkSession);
```

You can also configure TopLink to use the `DefaultTableGenerator` to automatically generate and execute a `TableCreator` at run time (see [Section 6.4, "Creating Database Tables Automatically"](#)).

### 6.2.3 How to Use Java to Create a Table Creator

This section describes how to create a `TableCreator` class in Java, including the following:

- [Creating a TableCreator Class](#)
- [Creating a TableDefinition Class](#)
- [Adding Fields to a TableDefinition](#)

- [Defining Sybase and Microsoft SQL Server Native Sequencing](#)

### 6.2.3.1 Creating a TableCreator Class

To create your own `TableCreator` instance, you should extend `TableCreator`, as [Example 6–1](#) shows:

#### **Example 6–1** *Creating a TableCreator Class*

```
public class MyTableCreator extends oracle.toplink.tools.schemaframework.TableCreator {

    public M7TableCreator() {
        setName("MyTableCreator");
        addTableDefinition(buildADDRESSTable());
        ...
    }

    public TableDefinition buildADDRESSTable() {
        TableDefinition table = new TableDefinition();
        ...
        return table;
    }
    ...
}
```

### 6.2.3.2 Creating a TableDefinition Class

The `TableDefinition` class includes all the information required to create a new table, including the names and properties of a table and all its fields.

The `TableDefinition` class has the following methods:

- `setName`
- `addField`
- `addPrimaryKeyField`
- `addIdentityField`
- `addForeignKeyConstraint`

All table definitions must call the `setName` method to set the name of the table that is described by the `TableDefinition`.

### 6.2.3.3 Adding Fields to a TableDefinition

Use the `addField` method to add fields to the `TableDefinition`. To add the primary key field to the table, use the `addPrimaryKeyField` method rather than the `addField` method.

To maintain compatibility among different databases, the type parameter requires a Java class rather than a database field type. TopLink translates the Java class to the appropriate database field type at run time. For example, the `String` class translates to the `CHAR` type for dBase databases. However, if you are connecting to Sybase, the `String` class translates to `VARCHAR`. For more information, see [Section 6.1.1, "How to Use Schema Manager Java and Database Type Conversion"](#).

The `addField` method can also be called with the `fieldSize` or `fieldSubSize` parameters for column types that require size and subsize to be specified.

Some databases require a subsize, but others do not. TopLink automatically provides the required information, as necessary.

### 6.2.3.4 Defining Sybase and Microsoft SQL Server Native Sequencing

Use `FieldDefinition` method `addIdentityField` to add fields representing a generated sequence number from Sybase or Microsoft SQL Server native sequencing. See [Section 18.2.2.6, "Native Sequencing with a Non-Oracle Database Platform"](#) for detailed information on using sequencing.

## 6.3 Creating Tables with a Table Creator

After creating a `TableCreator` class (see [Section 6.2, "Creating a Table Creator"](#)), you can use its API to create and drop tables. The important `TableCreator` methods are the following (each method takes an instance of `DatabaseSession`):

- `createTables`—this method creates tables, adds constraints, and creates sequence tables and sequences (if sequence tables already exist, this method drops them and recreates them).
- `dropTables`—his method drops all constraints and drops all tables (except sequence tables) that the `TableCreator` defines.
- `createConstraints`—this method creates constraints on all pre-existing tables that the `TableCreator` defines.
- `dropConstraints`—this method drops constraints on all pre-existing tables that the `TableCreator` defines.
- `replaceTables`—this method drops and then creates all tables that the `TableCreator` defines.

## 6.4 Creating Database Tables Automatically

You can configure TopLink to create database tables automatically in JPA and EJB CMP projects:

- [Creating Database Tables Automatically in JPA Projects](#)
- [Creating Database Tables Automatically in EJB CMP Projects](#)

### 6.4.1 Creating Database Tables Automatically in JPA Projects

Using EclipseLink JPA persistence unit properties that you can define in a `persistence.xml` file, you can configure schema generation

For more information, see "Using EclipseLink JPA Extensions for Schema Generation" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#Using\\_EclipseLink\\_JPA\\_Extensions\\_for\\_Schema\\_Generation](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#Using_EclipseLink_JPA_Extensions_for_Schema_Generation)

### 6.4.2 Creating Database Tables Automatically in EJB CMP Projects

If you deploy a CMP project to OC4J configured to use TopLink as the persistence manager, then you can configure OC4J to automatically create (and, optionally, delete) database tables for your persistent objects.

You can configure automatic database table creation at one of three levels, as [Table 6-2](#) shows. You can override the system level configuration at the application level, and you can override system and application configuration at the EJB module level.

**Table 6–2 Configuring Automatic Table Generation**

Level	Configuration File	Setting	Values
System (global)	<OC4J_HOME>/config/ application.xml	autocreate-tables	True or False <sup>1</sup>
		autodelete-tables	True or False <sup>1</sup>
Application (EAR)	orion-application.xml	autocreate-tables	True or False <sup>1</sup>
		autodelete-tables	True or False <sup>1</sup>
EJB Module (JAR)	orion-ejb-jar.xml	pm-properties sub-element default-mapping attribute db-table-gen <sup>2</sup>	Create, DropAndCreate, or UseExisting <sup>3</sup>

<sup>1</sup> Default.

<sup>2</sup> For more information, see [Section 9.9.1.3, "Configuring default-mapping Properties"](#).

<sup>3</sup> See [Table 6–3](#).

---

**Note:** In case of default mapping, the default value for autocreate-tables setting is true.

---

If you configure automatic table generation at the EJB module level, the value you assign to the db-table-gen attribute corresponds to the autocreate-tables and autodelete-tables settings, as [Table 6–3](#) shows.

**Table 6–3 Equivalent Settings for db-table-gen**

db-table-gen Setting	autocreate-tables Setting	autodelete-tables Setting
Create	True	False
DropAndCreate	True	True
UseExisting	False	NA

You can use this feature in conjunction with default mapping (see [Section 17.2.3.4, "Default Mapping in EJB 2.n CMP Projects Using OC4J at Run Time"](#)).



---

---

# Using an Integrated Development Environment

This chapter includes information on using TopLink with an integrated development environment (IDE).

This chapter includes the following sections:

- [Configuring TopLink for Oracle JDeveloper](#)
- [Configuring TopLink Workbench with Source Control Management Software](#)

In addition to the development environment described here, TopLink can be used with *any* Java EE development environment and process.

## 7.1 Configuring TopLink for Oracle JDeveloper

This section contains information on how to configure TopLink for use with Oracle JDeveloper. Oracle JDeveloper is a Java EE development environment with end-to-end support to develop, debug, and deploy e-business applications and Web services.

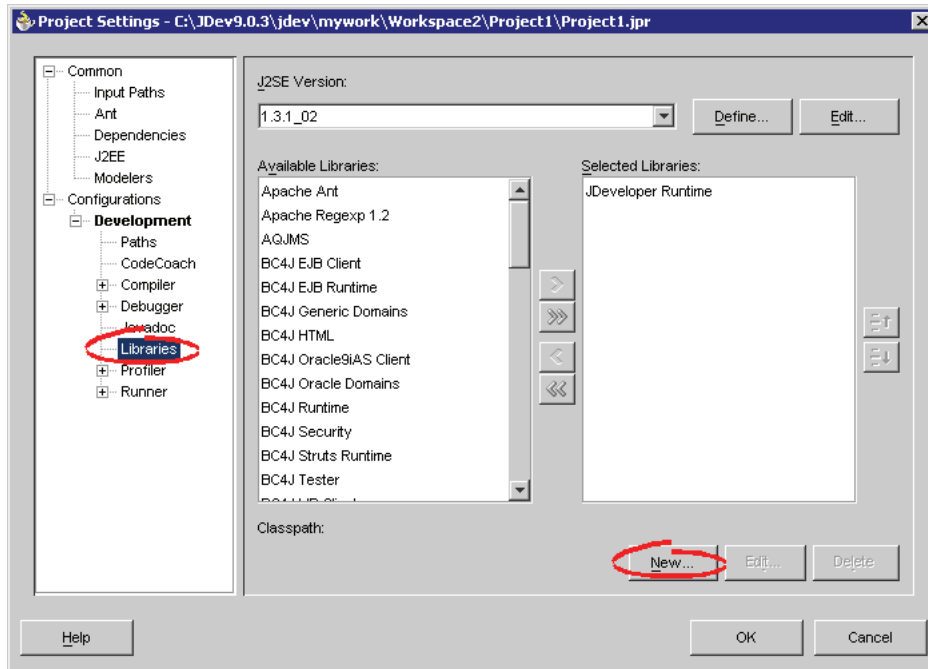
### 7.1.1 How to Use TopLink Mappings

Starting with Oracle JDeveloper 10g, the standard Oracle JDeveloper installation includes an embedded TopLink editor. Refer to the Oracle JDeveloper documentation for complete information.

To use TopLink with Oracle JDeveloper 9.0.4 (and earlier), use the following procedure to add the TopLink JAR files to your Oracle JDeveloper projects:

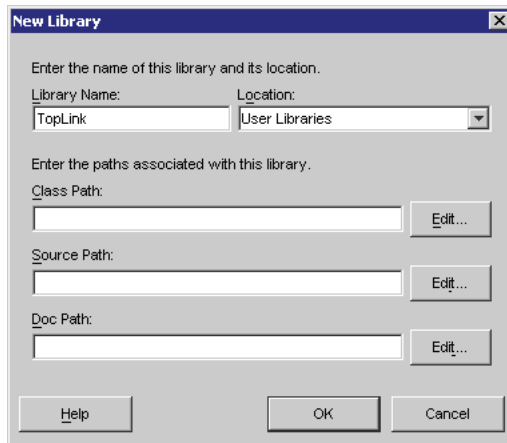
1. Select a Oracle JDeveloper project in the **System Navigator** window.
2. Choose **Project > Project Settings**. The **Project Settings** window appears.
3. Choose **Configurations > Development > Libraries**. A list of predefined and user-defined libraries appears.

**Figure 7–1 List of Available Libraries**



4. Click **New** to create a new library that will contain the TopLink . jar files. The New Library dialog box appears.
5. Enter a name for the new library—for example, **TopLink**. Ensure that the default choice for Location remains as **User Libraries**.

**Figure 7–2 Creating a New Library Dialog Box**



6. To edit the **Class Path** and add the TopLink . jar files, click **Edit**.  
Add the following to the beginning of your **Class Path**:
 

```
<TOPLINK_HOME>\jlib\toplink.jar
<TOPLINK_HOME>\modules\com.bea.core.antlr.runtime_2.7.7.jar
<ORACLE_HOME>\modules\xmlparserv2.jar
<ORACLE_HOME>\modules\xml.jar
```
7. Click **OK**. On the **Project Settings** window click **OK**.



### Using an Existing User-Defined TopLink Library

After a user library is created, it can be referenced again by any other project. Revisit the **Libraries** window of the Project Settings, and add the TopLink Library to any project with which you want to use TopLink.

## 7.2 Configuring TopLink Workbench with Source Control Management Software

You can use TopLink Workbench with a source control management (SCM) system to facilitate enterprise-level team development (see [Section 7.2.1, "How to Use a Source Control Management System"](#)). If you have a small development team, you can manage the changes from within XML files (see [Section 7.2.3, "How to Share Project Objects"](#)).

When using a TopLink Workbench project in a team environment, you must synchronize your changes with other developers. See [Section 7.2.2, "How to Merge Files"](#) for more information.

### 7.2.1 How to Use a Source Control Management System

If you use an enterprise, file-based source control management system to manage your Java source files, you can use the same system with your TopLink Workbench project files. These project files are maintained by TopLink Workbench and written out in XML file format.

The *check in* and *check out* mechanism for the source control system defines how to manage the source (the XML source and TopLink Workbench project file) in a multiuser environment.

#### Checking Out and Checking In TopLink Workbench Project Files

Although your actual development process will vary depending on your SCM tool, a typical process involves the following steps:

1. Determine (based on your SCM system) which files to retrieve from the source management system.
2. Edit the project using TopLink Workbench.
3. Save the edited project. If TopLink Workbench displays the Read-Only Files dialog box, make a note of these files, they must be unlocked and possibly merged. See [Section 7.2.5, "How to Work with Locked Files"](#) for more information.
4. Merge the required project files. See [Section 7.2.2, "How to Merge Files"](#) for details.
5. Check in the modified files, then retrieve from the repository any files that have been added or modified for this TopLink Workbench project.

### 7.2.2 How to Merge Files

The most difficult aspect of application development is merging changes from two (or more) development team members that have simultaneously edited the same file. If you check in your changes, a *merge* condition exists. Use a file comparison tool to determine the merged aspects of the project. The files to edit will vary, depending on the type of merge, as follows:

- [Merging Project Files](#)
- [Merging Table, Descriptor, and Class Files](#)

Example 7-1 and Example 7-2 demonstrate a *merge out* merging technique.

### 7.2.2.1 Merging Project Files

Project files contain references to the objects in the project. Generally, your project `<projectName>.mwp` contains the following elements:

- Database information: `<database>`
  - Database tables: `<tables>`
- Descriptors: `<descriptors>`
- Repository: `<repository>`
  - Classes: `<classpath-entries>`

Changes in these parts of the `.mwp` file are usually caused by adding, deleting, or renaming project elements.

To merge project files, you will generally need to merge a project file if another developer has added or removed a descriptor, table, or class, and checked in the project while you were adding or removing descriptors, tables, or classes from the same project. To merge the project's `.mwp` file, use this procedure:

1. Perform a file comparison between the `<projectName>.mwp` file in the repository and your local copy. The file comparison shows the addition or removal of a project element inside the owner (that is, `<database>`, `<descriptors>`, or `<repository>`).
2. Insert the XML script to, or delete from your local `<projectName>.mwp` file (inside the *corresponding owner element*). This brings your local code up-to-date to the current code in the code repository.
3. Retrieve any updated files, as indicated by your source control system.

Your local source now matches the repository.

#### **Example 7-1 Merging Projects**

Another developer has added and checked in a new **Employee** class descriptor to the `com.demo` package while you were working with the same TopLink Workbench project. To merge your work with the newly changed project, follow these steps:

1. Perform a file comparison on the `<projectName>.mwp` file to determine the differences between your local file and the file in the repository. Your SCM system may show the file in *merge* status.

The file comparison shows the addition of the `<package-descriptor>` tag and a `<name>` element inside that tag:

```
<package-descriptor>
  <name>com.demo.Employee.ClassDescriptor</name>
</package-descriptor>
```

2. Insert this XML into your `<projectName>.mwp` file (inside the `<descriptors>` element) to bring it up-to-date with the current files in the source repository.
3. Retrieve any new or updated files from your source control system. This includes the newly added **Employee** class descriptor.
4. Check in files that you have modified.

### 7.2.2.2 Merging Table, Descriptor, and Class Files

Developers who concurrently modify the same existing table, descriptor, or class file will create a merge condition for the following files:

- Table: `<tableName>.xml` (one for each table)
- Descriptor: `<descriptorName.type>.xml` (one for each descriptor)
- Class: `<className>.xml` (one for each class)

TopLink Workbench changes these files when saving a project if you have changed any of the contents within them (such as adding a mapping to a descriptor, adding an attribute to a class, or a changing a field reference in a table).

If another developer has changed an attribute in a table, descriptor, or class, while you were changing a different mapping on that same descriptor, you will need to merge your project. To merge your project, use this procedure:

1. Perform a file comparison on the specific `.xml` files in merge status (that is, table, descriptor, or class). The file comparison shows the addition or removal of an XML element.
2. Insert the XML script to, or remove from your local `.xml` file to bring it up-to-date with the current files in the source repository.

#### **Example 7-2 Merging Files**

Another developer has added and checked in the `firstName` mapping to the **Employee** class descriptor while you were changing a different mapping on that same descriptor. To merge your work with the newly changed project, use this procedure:

1. Perform a file comparison on the `com.demo.Employee.ClassDescriptor.xml` file located in `<projectRoot>/Descriptor/` directory that is in merge status.

The file comparison shows the addition of the `<mapping>` tag and the elements inside that tag:

```
<mapping>
  <uses-method-accessing>false</uses-method-accessing>
  <inherited>false</inherited>
  <read-only>false</read-only>
  <instance-variable-name>firstName</instance-variable-name>
  <default-field-names>
    <default-field-name>direct field=</default-field-name>
  </default-field-names>
  <field-handle>
    <field-handle>
      <table>EMPLOYEE</table>
      <field-name>F_NAME</field-name>
    </field-handle>
  </field-handle>
</mapping>
<mapping-class>MWDirectToFieldMapping </mapping-class>
</mapping>
```

2. Insert this XML block into your local `com.demo.Employee.ClassDescriptor.xml` file (inside the existing `<mapping>` element) to bring it up to date to the current files in the source repository.
3. Retrieve any new files noted as missing by your source control system. This includes any tables or descriptors that may be referenced by the new mapping.
4. Check in files that you have modified.

## 7.2.3 How to Share Project Objects

You can also share project objects by copying the table or descriptor files into the appropriate directories in the target project.

After copying the files, insert a reference to the table, descriptor, or class in the appropriate place in the `<projectName>.mwp` file. All references contained within the project file must refer to an existing object in the project.

## 7.2.4 How to Manage the `ejb-jar.xml` File

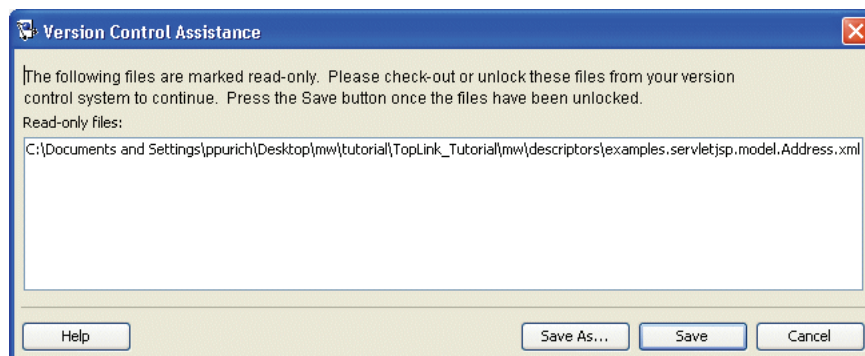
When working in a team environment, manage the `ejb-jar.xml` file similarly to the `.xml` project files. TopLink Workbench edits and updates the `ejb-jar.xml` file, if necessary, when working with an EJB project.

If you use a version control system, perform the same check in and check out procedures. For merge conditions, use a file comparison tool to determine which elements have been added or removed. Modify the file as necessary and check in the file to exercise version control on your work.

## 7.2.5 How to Work with Locked Files

When working in a team environment, your source control system may lock files when you retrieve them from the repository. If TopLink Workbench attempts to save a locked file, the Version Control Assistance dialog box appears, showing the locked files.

**Figure 7–3** Version Control Assistance Dialog Box



Select one of the following methods to save your project:

- Use your source control system to unlock the files, and then click **Save**.
- Click **Save As** to save the project to a new location.

See [Section 116.2.2, "How to Save Projects"](#) for more information.

# Part III

---

## TopLink Application Deployment

This part describes how to package and deploy a TopLink application to an application server. It contains the following chapters:

- [Chapter 8, "Integrating TopLink with an Application Server"](#)

This chapter contains information on software requirements for integrating TopLink with your specific application server.

- [Chapter 9, "Creating TopLink Files for Deployment"](#)

This chapter describes how to create the necessary TopLink files for deployment to your application server.

- [Chapter 10, "Packaging a TopLink Application"](#)

This chapter explains how to package the deployment files.

- [Chapter 11, "Deploying a TopLink Application"](#)

This chapter provides procedures for deploying different types of TopLink applications in a variety of environments.



---



---

# Integrating TopLink with an Application Server

This chapter describes how to configure Oracle TopLink for use with Java EE containers and application servers. Although you can use Oracle TopLink with *any* Java EE container or application server (by using the TopLink API), TopLink provides specific integration and support for the application servers listed in this chapter.

This chapter includes the following sections:

- Introduction to the Application Server Support
- Integrating TopLink with an Application Server
- Integrating TopLink with Oracle WebLogic Server
- Integrating TopLink with Oracle Containers for Java EE (OC4J)
- Integrating TopLink with IBM WebSphere Application Server
- Integrating TopLink with Sun Application Server
- Integrating TopLink with JBoss Application Server
- Defining Security Permissions
- Configuring Miscellaneous EJB CMP Options

For more information, see the following:

- Chapter 9, "Creating TopLink Files for Deployment"
- Chapter 10, "Packaging a TopLink Application"
- Chapter 11, "Deploying a TopLink Application"

## 8.1 Introduction to the Application Server Support

You can use TopLink with any Java EE application server (by using the TopLink API) that meets the requirements shown in "[What Are the Software Requirements](#)" on page 8-2.

[Table 8-1](#) lists the application servers for which TopLink provides specific JPA, CMP 2.1, and session EJB/BMP integration.

**Table 8-1 TopLink Integration Support by Application Server Type**

Application Server Type	Application Server Version	JPA	CMP 2.1	(Session Bean <sup>1</sup> and BMP)
Oracle WebLogic Server	10.3	✓		✓

**Table 8–1 (Cont.) TopLink Integration Support by Application Server Type**

Application Server Type	Application Server Version	JPA	CMP 2.n	(Session Bean <sup>1</sup> and BMP)
Oracle WebLogic Server	9.n			✓
OC4J	10.1.3.n	✓	✓	✓
OC4J	10.1.3		✓	✓
OC4J	9.0.4			✓
OC4J	9.0.3			✓
IBM WebSphere	6.1	✓		✓
SunAS	9	✓		✓
JBoss	4.2.0	✓		✓

<sup>1</sup> This applies to EJB 1.n, 2.n and 3.0 session beans.

For more information, see the following:

- [Section 8.3, "Integrating TopLink with Oracle WebLogic Server"](#)
- [Section 8.4, "Integrating TopLink with Oracle Containers for Java EE \(OC4J\)"](#)
- [Section 8.5, "Integrating TopLink with IBM WebSphere Application Server"](#)
- [Section 8.6, "Integrating TopLink with Sun Application Server"](#)
- [Section 8.7, "Integrating TopLink with JBoss Application Server"](#)

## 8.2 Integrating TopLink with an Application Server

This section describes concepts unique to TopLink application server integration, including the following:

- [What Are the Software Requirements](#)
- [How to Configure the XML Parser Platform](#)
- [How to Set Security Permissions](#)
- [How to Migrate the Persistence Manager](#)
- [How to Integrate Clustering](#)

### 8.2.1 What Are the Software Requirements

To run a TopLink application within a Java EE container, your system must meet the following software requirements:

- An application server or Java EE container (see [Table 8–1](#));
- XML parser (see [Section 8.2.2, "How to Configure the XML Parser Platform"](#));
- A JDBC driver configured to connect with your local database system (for more information, see your database administrator);
- A Java development environment, such as the following:
  - Oracle JDeveloper;
  - IBM WebSphere Studio Application Developer (WSAD);
  - Sun Java Development Kit (JDK) 1.5 or later;



- Any other Java environment that is compatible with the Sun JDK 1.5 or later;
- A command-line JVM executable (such as `java.exe` or `jre.exe`).

## 8.2.2 How to Configure the XML Parser Platform

The TopLink run-time environment uses an XML parser to do the following:

- Read and write XML configuration files (see [Section 9.1.1, "project.xml File"](#) and [Section 9.1.2, "sessions.xml File"](#));
- Read and write TopLink Workbench project files (see [Section 5.1, "Introduction to TopLink Workbench"](#));
- Perform object-to-XML transformations in EIS projects using XML records (see [Chapter 77, "Introduction to EIS Mappings"](#));
- Perform object-to-XML transformations in XML projects (see [Chapter 53, "Introduction to XML Mappings"](#));

Application servers use an XML parser to read deployment files, such as `ejb-jar.xml` and `<Java EE container>-ejb-jar.xml` files (see [Chapter 9, "Creating TopLink Files for Deployment"](#)).

To avoid XML parser conflicts, you must configure your TopLink application to use the same XML parser as that used by the application server on which you deploy your application.

Internally, TopLink accesses its XML parser using an instance of `oracle.toplink.platform.xml.XMLPlatform` class.

You can configure TopLink to use any XML parser for which an `XMLPlatform` class exists (see [Section 8.2.2.1, "Configuring XML Parser Platform"](#)).

You can also create your own `XMLPlatform` to provide access to an XML parser not currently supported by TopLink (see [Section 8.2.2.2, "Creating an XML Parser Platform"](#)).

### 8.2.2.1 Configuring XML Parser Platform

TopLink provides the `XMLPlatform` instances shown in [Table 8–2](#).

**Table 8–2 Supported XML Platforms**

XMLPlatform...	Provides Access to...	Use with...
<code>oracle.toplink.platform.xml.xdk.XDKPlatform</code> <sup>1</sup>	XDKParser: this class provides access to the Oracle XML Developer's Kit (XDK) XML parser (see <a href="http://www.oracle.com/technology/tech/xml/xdkhome.html">http://www.oracle.com/technology/tech/xml/xdkhome.html</a> ).	See <a href="#">Section 8.4, "Integrating TopLink with Oracle Containers for Java EE (OC4J)"</a>
<code>oracle.toplink.platform.xml.jaxp.JAXPPlatform</code>	JAXPParser: this class provides access to the Java SDK XML parser in the <code>javax.xml.parsers</code> package (see <a href="http://java.sun.com/j2ee/1.4/docs/tutorial/doc/JAXPIntro2.html">http://java.sun.com/j2ee/1.4/docs/tutorial/doc/JAXPIntro2.html</a> ).	See the following: <ul style="list-style-type: none"> <li>■ <a href="#">Section 8.3, "Integrating TopLink with Oracle WebLogic Server"</a></li> <li>■ <a href="#">Section 8.5, "Integrating TopLink with IBM WebSphere Application Server"</a></li> </ul>

<sup>1</sup> Default.

---

---

**Note:** To use an XML parser not listed in [Table 8–2](#), create your own `XMLPlatform` (see [Section 8.2.2.2, "Creating an XML Parser Platform"](#)).

---

---

To configure your TopLink application to use a particular instance of the `XMLPlatform` class, set system property `toplink.xml.platform` to the fully qualified name of your `XMLPlatform` class, as [Example 8–1](#) shows.

**Example 8–1 Configuring XML Platform**

```
toplink.xml.platform=oracle.toplink.platform.xml.jaxp.JAXPPlatform
```

### 8.2.2.2 Creating an XML Parser Platform

Using the `oracle.toplink.platform.xml` classes included in the public source files shipped with TopLink (see [Section 13.3, "Using Public Source"](#)), you can create your own instance of the `oracle.toplink.platform.xml.XMLPlatform` class to specify an XML parser not listed in [Table 8–2](#).

After creating your `XMLPlatform`, configure TopLink to use it (see [Section 8.2.2.1, "Configuring XML Parser Platform"](#)).

### 8.2.2.3 XML Parser Limitations

Crimson (<http://xml.apache.org/crimson/>) is the XML parser supplied in the Java Platform, Standard Edition (Java SE) and in some JAXP reference implementations.

If you use Crimson with the JAXP API to parse XML files whose system identifier is not a fully qualified URL, then XML parsing will fail with a *not valid URL* exception.

Other XML parsers defer validation of the system identifier URL until it is specifically referenced.

If you are experiencing this problem, consider one of the following alternatives:

- Ensure that your XML files use a fully qualified system identifier URL.
- Use another XML parser (such as the OracleAS XML Parser for Java v2).

## 8.2.3 How to Set Security Permissions

By default, when you run a TopLink-enabled application in a JVM configured with a nondefault `java.lang.SecurityManager`, the TopLink run-time environment executes certain internal functions by executing a `PrivilegedAction` with `java.security.AccessController` method `doPrivileged`. This ensures that you do not need to grant many permissions to TopLink for it to perform its most common operations. You need only grant certain permissions depending on the types of optional TopLink features you use.

For more information, see [Section 8.8, "Defining Security Permissions"](#).

If you run a TopLink-enabled application in a JVM without a nondefault `SecurityManager`, you do not need to set any permissions.

## 8.2.4 How to Migrate the Persistence Manager

You can configure an application server to use TopLink as the persistence manager.

You can only use one persistence manager for all the entity beans with container-managed persistence in a JAR file.

TopLink provides automated support for migrating an existing Java EE application to use TopLink as the persistence manager. For more information, see [Section 8.4.2, "How to Migrate OC4J Orion CMP Persistence to OC4J TopLink Persistence"](#).

## 8.2.5 How to Integrate Clustering

Most application servers include a clustering service that you can use with your TopLink application.

To use TopLink with an application server cluster, use this general procedure:

1. Install the `toplink.jar` file (and include it in the classpath) on each application server in the cluster to which you deploy TopLink applications.
2. Configure TopLink cache consistency options appropriate for your application.

For more information, see [Chapter 102, "Introduction to Cache"](#).

If you are deploying a CMP application, see also [Section 9.10.1.2, "Configuring cache-synchronization Properties"](#).

3. Configure TopLink coordinated cache support for your application server, if available.
4. Configure clustering on each application server.

For more information, see your application server documentation.

## 8.3 Integrating TopLink with Oracle WebLogic Server

To integrate a TopLink application with Oracle WebLogic Server, you must consider the following:

- [How to Configure Classpath](#)
- [How to Integrate JTA](#)
- [How to Integrate JMX](#)
- [How to Integrate the Security Manager](#)

In addition to configuring these Oracle WebLogic Server-specific options, you must also consider the general application server integration issues in [Section 8.2, "Integrating TopLink with an Application Server"](#).

### 8.3.1 How to Configure Classpath

There is no need for the application server classpath modifications, as TopLink works out of the box in Oracle WebLogic Server.

Note that both the TopLink library in the form of the `com.oracle.toplink_*.jar` file, and the EclipseLink library in the form of the `org.eclipse.persistence_*.jar` file, reside in the following location on the server:

```
<BEA_HOME>/modules/
```

where `<BEA_HOME>` is the directory in which the standalone Oracle WebLogic Server is installed.

---

---

**Note:** Although not suitable for the production environment, during development time you might consider different configurations of the library files. These configuration options depend on the desired location of the libraries in your classpath tree, as well as on whether or not you want to run two different server versions. For more information, see [http://wiki.eclipse.org/EclipseLink/Examples/JPA/WebLogic\\_Web\\_Tutorial#EclipseLink\\_JAR\\_location](http://wiki.eclipse.org/EclipseLink/Examples/JPA/WebLogic_Web_Tutorial#EclipseLink_JAR_location)

---

---

### 8.3.2 How to Integrate JTA

For applications that require JTA integration, specify the external transaction controller when you configure the server platform in your session (see [Section 89.9, "Configuring the Server Platform"](#)).

For more information, see [Section 115.13, "Integrating the Unit of Work with an External Transaction Service"](#).

### 8.3.3 How to Integrate JMX

By default, when you deploy a TopLink application to Oracle WebLogic Server, the TopLink runtime uses the EclipseLink persistence provider to deploy the following Java Management Extensions (JMX) MBean to the Oracle WebLogic Server JMX service for each session:

- `org.eclipse.persistence.services.DevelopmentServices`
- `org.eclipse.persistence.services.RuntimeServices`

For more information, see "How to Integrate JMS" section of the *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Integrating\\_EclipseLink\\_with\\_an\\_Application\\_Server\\_%28ELUG%29#How\\_to\\_Integrate\\_JMX](http://wiki.eclipse.org/Integrating_EclipseLink_with_an_Application_Server_%28ELUG%29#How_to_Integrate_JMX)

For information on Oracle WebLogic Server JMX support, see the following documentation:

- *Oracle Fusion Middleware Developing Manageable Applications With JMX for Oracle WebLogic Server*
- *Oracle Fusion Middleware Developing Manageable Applications With JMX for Oracle WebLogic Server*

For information on JMX in general, see

<http://java.sun.com/docs/books/tutorial/jmx/index.html>.

### 8.3.4 How to Integrate the Security Manager

If you use a security manager, specify a security policy file in the `weblogic.policy` file (usually located in the Oracle WebLogic Server install directory), as follows:

```
-Djava.security.manager  
-Djava.security.policy==<BEA_HOME>\wlserver_10.3\weblogic.policy
```

The Oracle WebLogic Server installation procedure includes a sample security policy file. You need to edit the `weblogic.policy` file to grant permission for TopLink to use reflection.

The following example illustrates only the permissions that TopLink requires, but most `weblogic.policy` files contain more permissions than are shown in this example.

**Example 8–2 A Subset of a "Grant" Section from a weblogic.policy File**

```
grant {
// "enableSubstitution" required to run the WebLogic console
permission java.io.SerializablePermission "enableSubstitution";
// "modifyThreadGroup" required to run the WebLogic server
permission java.lang.RuntimePermission "modifyThreadGroup";
// grant permission for TopLink to use reflection
permission java.lang.reflect.ReflectPermission "suppressAccessChecks";
};
```

## 8.4 Integrating TopLink with Oracle Containers for Java EE (OC4J)

To integrate a TopLink application with OC4J, you must consider the following:

- [How to Integrate CMP](#)
- [How to Migrate OC4J Orion CMP Persistence to OC4J TopLink Persistence](#)
- [How to Integrate JTA](#)
- [How to Integrate with Oracle Application Server Manageability and Diagnosability](#)

In addition to configuring these OC4J-specific options, you must also consider the general application server integration issues in [Section 8.2, "Integrating TopLink with an Application Server"](#).

### 8.4.1 How to Integrate CMP

To enable TopLink CMP integration in OC4J, use the following procedure (this procedure assumes you have already installed TopLink):

1. If necessary, migrate your CMP application using the TopLink migration tool (see [Section 8.4.2, "How to Migrate OC4J Orion CMP Persistence to OC4J TopLink Persistence"](#)).
2. Evaluate your choice of `UnitOfWork` change policy (see [Section 113.2.3, "Unit of Work and Change Policy"](#)).
3. Ensure that all necessary deployment descriptor files are in place (see [Chapter 9, "Creating TopLink Files for Deployment"](#) and [Section 10, "Packaging a TopLink Application"](#)).
4. Optionally, consider the EJB customization options that TopLink provides (see [Section 8.9, "Configuring Miscellaneous EJB CMP Options"](#)).

### 8.4.2 How to Migrate OC4J Orion CMP Persistence to OC4J TopLink Persistence

If you upgrade OC4J 9.0.4 or earlier to 11g Release 1 (11.1.1), you must migrate persistence configuration from your original `orion-ejb-jar.xml` file to the `toplink-ejb-jar.xml` file.

In 11g Release 1 (11.1.1), Oracle provides a TopLink migration tool that you can use to automate this migration for Release 2 (9.0.4) or later OC4J installations.

After using the TopLink migration tool, you may need to make some additional changes as described in [Section 8.4.2.4, "Performing Post-Migration Changes"](#).

If you encounter problems during migration, see [Section 8.4.2.5, "Troubleshooting Your Migration"](#).

This section explains how to use the TopLink migration tool, including the following:

- [What You May Need to Know About Migrating OC4J Orion Persistence to OC4J TopLink Persistence](#)
- [Using the TopLink Migration Tool from TopLink Workbench](#)
- [Using the TopLink Migration Tool from the Command Line](#)
- [Performing Post-Migration Changes](#)
- [Troubleshooting Your Migration](#)

---

---

**Note:** You can also use the TopLink migration tool from Oracle JDeveloper.

---

---

### 8.4.2.1 What You May Need to Know About Migrating OC4J Orion Persistence to OC4J TopLink Persistence

Before using the TopLink migration tool, review this section to understand how the TopLink migration tool works and to determine what OC4J persistence manager metadata is, and is not, migrated.

#### Input and Output

The TopLink migration tool takes the following files as input:

- `ejb-jar.xml`
- `orion-ejb-jar.xml`

It migrates as much OC4J-specific persistence configuration as possible to a new `toplink-ejb-jar.xml` file and creates the following new files in a target directory you specify:

- `orion-ejb-jar.xml`
- `toplink-ejb-jar.xml`
- TopLink Workbench project file `TLCmpProject.mwp`

The `ejb-jar.xml` and `orion-ejb-jar.xml` files may be in an EAR, JAR, or just standalone XML files. If you migrate from standalone XML files (rather than an EAR or JAR file), ensure that the domain classes are accessible and included in your classpath.

The TopLink migration tool creates a new `orion-ejb-jar.xml` and `toplink-ejb-jar.xml` file to the target directory you specify in the same format as it reads the original files. For example, if you specify an EAR file as input, then the TopLink migration tool stages and creates a new EAR file that contains both the new `orion-ejb-jar.xml` and the new `toplink-ejb-jar.xml` file, but is otherwise identical to the original.

The TopLink Workbench project file is always created as a separate file.

---

---

**Note:** Oracle recommends that you make a backup copy of your `orion-ejb-jar.xml` file before using the TopLink migration tool.

---

---

#### Migration

As it operates, the TopLink migration tool logs all errors and diagnostic output to a log file named `oc4j_migration.log` in the output directory. If you use the TopLink

migration tool from TopLink Workbench, see also the TopLink Workbench log file `oracle.toplink.workbench.log` located in your user home directory (for example, `C:\Documents and Settings\<user-name>`).

The TopLink migration tool processes descriptor, mapping, and query information from the input files as follows:

- It builds a TopLink descriptor object for each entity bean and migrates native persistence metadata like mapped tables, primary keys, and mappings for CMP and CMR fields.
- It builds a TopLink mapping object for every CMP and CMR field of an entity bean and migrates native persistence metadata like foreign key references.
- It builds a TopLink query object for each `finder` or `ejbSelect` of an entity bean and migrates persistence metadata like customized query statements.

Table 8–3 lists OC4J `<entity-deployment>` attributes and subelements from the `orion-ejb-jar.xml` file and for each, indicates whether or not the TopLink migration tool:

- Retains it in the new `orion-ejb-jar.xml` file
- Migrates it to the new `toplink-ejb-jar.xml` file

In Table 8–3, elements are identified with angle brackets. Note that in some cases an attribute is migrated when set to one value, but discarded if set to another value (for example, `exclusive-write-access`).

**Table 8–3 OC4J `orion-ejb-jar.xml` Feature Migration**

orion-ejb-jar.xml Feature	Retained in New orion-ejb-jar.xml	Migrated to New toplink-ejb-jar.xml
<code>&lt;entity-deployment&gt;</code>		
<code>clustering-schema</code>	✓	
<code>copy-by-value</code>	✓	
<code>data-source</code>	✓	
<code>location</code>	✓	
<code>max-instances</code>	✓	
<code>min-instances</code>	✓	
<code>max-tx-retries</code>	✓	
<code>disable-wrapper-cache</code>	✓	
<code>name</code>	✓	
<code>pool-cache-timeout</code>	✓	
<code>wrapper</code>	✓	
<code>local-wrapper</code>	✓	
<code>call-timeout</code>		✓
<code>exclusive-write-access</code>		
<code>true</code>		✓
<code>false</code>		
<code>do-select-before-insert</code>		
<code>true</code>		
<code>false</code>		
<code>isolation</code>		✓
<code>locking-mode</code>		

**Table 8–3 (Cont.) OC4J orion-ejb-jar.xml Feature Migration**

orion-ejb-jar.xml Feature	Retained in New orion-ejb-jar.xml	Migrated to New toplink-ejb-jar.xml
pessimistic		✓
optimistic		
read-only		✓
old_pessimistic		
update-changed-fields-only		
true		✓
false		
table		✓
force-update		
true <sup>1</sup>		
false		✓
data-synchronization-option		
ejbCreate		
ejbPostCreate		
batch-size		
Any value greater than 1		
<ior-security-config>	✓	
<env-entry-mapping>	✓	
<resource-ref-mapping>	✓	
<resource-env-ref-mapping>	✓	
<primkey-mapping>		✓
<cmp-field-mapping>		✓
one-to-one-join		
inner		✓
outer <sup>2</sup>		
shared		✓
<finder-method>		✓
<persistence-type> <sup>3</sup>		✓

<sup>1</sup> You can enable `force-update` after migration. For more information, see [Section 119.18, "Configuring a Descriptor with EJB CMP and BMP Information"](#).

<sup>2</sup> TopLink supports both `outer` and `inner` joins at run time. You can manually configure EJB descriptors with these options. For more information, see [Section 108.7.1.5, "Join Reading and Object-Level Read Queries"](#).

<sup>3</sup> The `persistence-type` attribute's `table` column size, if present, is discarded. For more information, see [Section , "Recovering persistence-type Table Column Size"](#).

Table 8–4 lists OC4J features and their TopLink equivalents configured by the TopLink migration tool.

**Table 8–4 OC4J and TopLink Feature Comparison**

Feature	orion-ejb-jar.xml	toplink-ejb-jar.xml
CMP field mapping	Direct	Direct-to-field
	Serialized object	Serialized object

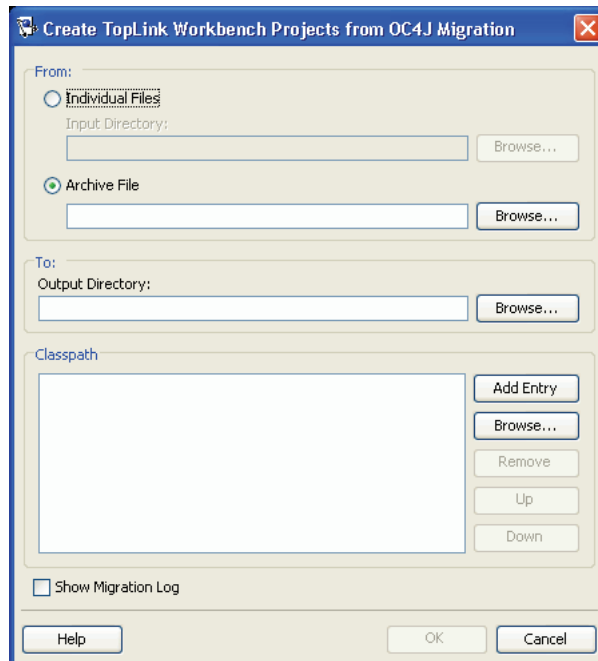


**Table 8–4 (Cont.) OC4J and TopLink Feature Comparison**

Feature	orion-ejb-jar.xml	toplink-ejb-jar.xml
CMR field mapping	One-to-one	One-to-one
	One-to-many	One-to-many
	Many-to-many	Many-to-many
Partial query	Full SQL statement	SQL Call
Finder	Oracle specific syntax	SQL Call or EJB QL
Lazy loading (fetch group)	Lazy loading of primary key and CMP fields	Not supported Alternatively, you can manually configure the TopLink equivalent, if appropriate (see <a href="#">Section 16.2.4, "Fetch Groups"</a> ).
SQL statement caching	Cache static SQL	Not supported at the bean level. TopLink supports parameterized SQL and statement caching at the session and query level (see <a href="#">Chapter 108, "Introduction to TopLink Queries"</a> ).
Locking	Optimistic: database-level	Optimistic: object-level
	Pessimistic: bean instance-level	Pessimistic: query lock at database-level
Read-only	Attempt to change throws <code>Exception</code>	Attempt to change throws <code>Exception</code>
Validity timeout	Read-only bean validity timeout before reloaded.	Cache timeout
Isolation level	Committed	Committed
	Serializable	Serializable
		Not Committed
		Not Repeatable
Delay update until commit	Supported	Supported (see <a href="#">Section 119.18, "Configuring a Descriptor with EJB CMP and BMP Information"</a> ).
Exclusive write access on bean	Default value is <code>false</code>	Assume <code>true</code>
Insert without existence check	Supported	Supported
Update changed fields only	Supported	Supported (see <a href="#">Section 113.2.3.3, "Attribute Change Tracking Policy"</a> ).
Force update	Invoke bean life cycle <code>ejbStore</code> method even though persistent fields have not changed	Supported

### 8.4.2.2 Using the TopLink Migration Tool from TopLink Workbench

1. From TopLink Workbench, select **File > Migrate > From OC4J 9.0.x**.

**Figure 8–1 Create TopLink Workbench Projects from OC4J Migration Dialog Box**

Use the following information to enter data in each field of the Create TopLink Workbench Projects from OC4J Migration dialog box:

Field	Description
<b>From</b>	Use these fields to specify the location of the existing OC4J files. These files may be included as part of a JAR, EAR, or individual files.
<b>Individual Files</b>	Select to convert from individual <code>ejb-jar.xml</code> and <code>orion-ejb-jar.xml</code> files in the <b>Input Directory</b> . Click <b>Browse</b> and select the directory location that contains the XML files to convert from.
<b>Archive File</b>	Select to use a specific archive file. Click <b>Browse</b> and select the archive file to convert from.
<b>To</b>	Use these fields to specify the location to which migrated files are written.
<b>Output Directory</b>	Click <b>Browse</b> and select a directory location in which to create the new XML files and TopLink Workbench project.
<b>Classpath</b>	If you are migrating from individual files, ensure that the domain classes are accessible and included in your classpath.
<b>Show Migration Log</b>	Select to have migration log output displayed in a separate window.

### 8.4.2.3 Using the TopLink Migration Tool from the Command Line

To use the TopLink migration tool from the command line, you must perform the following steps:

1. Ensure that the following is in your classpath:
  - `<TOPLINK_HOME>/modules/oracle.toplink_11.1.1/jlib/antlr.jar`

- `<TOPLINK_HOME>/jlib/toplink.jar`
  - `<TOPLINK_HOME>/utils/workbench/jlib/cmpmigrator.jar`
  - `<TOPLINK_HOME>/utils/workbench/jlib/toplinkmw.jar`
  - `<TOPLINK_HOME>/utils/workbench/jlib/tlmwcore.jar`
  - `<TOPLINK_HOME>/jlib`
  - `<TOPLINK_HOME>/modules/xmlparserv2.jar`
2. If you intend to migrate from plain XML files (rather than an EAR or JAR file), ensure that the domain classes are accessible and included in your classpath.
  3. Make a backup copy of your original XML files.
  4. Execute the TopLink migration tool, as [Example 8-6](#) illustrates, using the appropriate arguments listed in [Table 8-5](#).

The usage information for the TopLink migration tool is as follows:

```
java -Dtoplink.ejbjar.schemavalidation=<true|false>
-Dtoplink.migrationtool.generateWorkbenchProject=<true|false>
-Dhttp.proxyHost=<proxyHost>
-Dhttp.proxyPort=<proxyPort> oracle.toplink.tools.migration.TopLinkCMPMigrator
-s<nativePM> -i<inputDir>
-a<ear>|<jar> -x -o<outputDir> -v
```

To identify the input files, you must specify one of `-a` or `-x`.

For troubleshooting information, see [Section 8.4.2.5, "Troubleshooting Your Migration"](#).

### Example 8-3 Using the TopLink Migration Tool from the Command Line

```
java -Dhttp.proxyHost=www-proxy.us.oracle.com
-Dhttp.proxyPort=80 oracle.toplink.tools.migration.TopLinkCMPMigrator
-sOc4j-native -iC:/mywork/in -aEmployee.ear -oC:/mywork/out -v
```

**Table 8-5 TopLink Migration Tool Arguments**

Argument	Description
<code>toplink.ejbjar.schemavalidation</code>	The system property used to turn on schema validation if <code>ejb-jar.xml</code> uses XML Schema (XSD) instead of DTD. The default value is <code>false</code> .
<code>toplink.migrationtool.generateWorkbenchProject</code>	The system property used enable generation of the TopLink Workbench project. The default value is <code>true</code> .
<code>&lt;proxyHost&gt;</code>	The address of your local HTTP proxy host
<code>&lt;proxyHost&gt;</code>	The port number on which your local HTTP proxy host receives HTTP requests.
<code>-s &lt;source&gt;</code>	The name of the native persistence manager from which you are migrating. For OC4J, use the name <code>Oc4j-native</code> .
<code>-i &lt;input-directory&gt;</code>	Fully qualified path to the input directory that contains both the OC4J <code>ejb-jar.xml</code> and <code>orion-<i>ejb-jar.xml</i></code> files to migrate. Default: current working directory.
<code>-a &lt;EAR-or-JAR&gt;</code>	Fully qualified path to the archive file (either an EAR or JAR) that contains both the OC4J <code>ejb-jar.xml</code> and <code>orion-<i>ejb-jar.xml</i></code> files to migrate.

**Table 8–5 (Cont.) TopLink Migration Tool Arguments**

Argument	Description
-x	Tells the TopLink migration tool that the OC4J files in the input directory to migrate from are plain XML files (not in an archive file).  If you use this option, ensure that the domain classes are accessible and included in your classpath.
-o <output-directory>	<targetDir> is the path to the directory into which the TopLink migration tool writes the new <code>orion-ejb-jar.xml</code> , <code>toplink-ejb-jar.xml</code> , and log files. The path may be absolute or relative to the current working directory. You must specify this argument value.  Ensure that permissions are set on this directory to allow the TopLink migration tool to create files and subdirectories.
-v	Verbose mode. Tells the TopLink migration tool to log errors and diagnostic information to the console.

#### 8.4.2.4 Performing Post-Migration Changes

After you migrate the `orion-ejb-jar.xml` file persistence configuration to your `toplink-ejb-jar.xml` file, consider the following post-migration changes:

- [Recovering persistence-type Table Column Size](#)
- [Updating the Unknown Primary Key Class Mapping Sequence Table](#)
- [Customizing a Project](#)

#### Recovering persistence-type Table Column Size

In the `orion-ejb-jar.xml` file, you can specify this mapping, `cmp-field-mapping`, with a `persistence-type` attribute that provides both the type and column size, as shown in [Example 8–7](#).

#### **Example 8–4 A `cmp-field-mapping` with `persistence-type` Specifying a Column Size**

```
<cmp-field-mapping ejb-reference-home="MyOtherEntity" name="myField"
persistence-name="myField" persistence-type="VARCHAR2(30)">
```

The TopLink migration tool migrates the persistence type but not the column size because a TopLink project does not normally contain this size information.

To recover the `persistence-type` column size, do the following:

1. Perform the migration as described in [Section 8.4.2.3, "Using the TopLink Migration Tool from the Command Line"](#).
2. Launch the generated TopLink Workbench project file `TLCompProject.mwp`.
3. Log in to your database (see [Section 5.5.1.1, "Logging In and Out of a Database"](#)).  
TopLink Workbench retrieves all column sizes.

#### Updating the Unknown Primary Key Class Mapping Sequence Table

TopLink supports the use of an unknown primary key class (see [Section 8.9.2, "How to Configure EJB CMP Unknown Primary Key Class Support"](#)) and so the TopLink migration tool also supports this feature.

OC4J uses a native run-time key generator to generate unique keys for auto-id key fields. In contrast, TopLink uses a sequencing table.

If your OC4J persistence configuration includes the use of an unknown primary key class, then the TopLink migration tool will create a sequencing table for this purpose.

Before deploying your application after migration, you must do the following:

1. Determine the largest key value generated prior to migration.
2. Manually update the counter of the TopLink migration tool-generated sequence table to a number that must be one larger than the largest key value generated prior to migration.

### Customizing a Project

You can customize the following components of your project:

- [EJB 2.1 Persistence Manager Customization](#)
- [Session Event Listener](#)

**8.4.2.4.1 EJB 2.1 Persistence Manager Customization** For an EJB 2.1 CMP application deployed to OC4J, you customize the TopLink persistence manager by configuring properties in the `orion-ejb-jar.xml` file. These properties are used to configure the TopLink session that the TopLink runtime uses internally for CMP projects. For more information, see [Section 9.10.1, "How to Configure persistence-manager Entries"](#).

**8.4.2.4.2 Session Event Listener** After you applied the default settings to your project at deployment time, you may wish to customize the TopLink session by configuring the session event listener. The pre-login event that the session raises is particularly useful. It lets you define the custom (nondefault) specifics for the session just before the session initializes and acquires connections.

For more information, see the following:

- [Section 87.2.3, "Session Customization"](#)
- [Section 87.2.5, "Managing Session Events with the Session Event Manager"](#)
- [Section 89.10, "Configuring Session Event Listeners"](#)

### 8.4.2.5 Troubleshooting Your Migration

This section describes solutions for problems you may encounter during migration, including the following:

- [Log Messages](#)
- [Unexpected Relational Multiplicity](#)

#### Log Messages

As it operates, the TopLink migration tool logs all errors and diagnostic output to a log file named `oc4j_migration.log` in the output directory. If you use the TopLink migration tool from TopLink Workbench, see also the TopLink Workbench log file `oracle.toplink.workbench.log` located in your user home directory (for example, `C:\Documents and Settings\<user-name>`)

In addition to these warnings, the TopLink migration tool logs an error if it encounters a problem that prevents it from completing the migration. [Table 8–6](#) lists these problems and suggests possible solutions.

**Table 8–6 TopLink Migration Tool Error Messages**

Error Message	Description
There is no <code>ejb-jar.xml</code> in the input file. You must provide the <code>ejb-jar.xml</code> in order for the migration process to work.	The <code>ejb-jar.xml</code> file is missing. The TopLink migration tool stops and copies the original input files into the target directory. Verify that the <code>ejb-jar.xml</code> file is present in the specified EAR, JAR, or as a plain XML file. Empty the target directory and execute the TopLink migration tool again.
There is not an <code>orion-ejb-jar.xml</code> with native persistent metadata defined, no migration needed.	The <code>orion-ejb-jar.xml</code> file is missing. The TopLink migration tool stops and copies the original input files into the target directory. Verify that the <code>orion-ejb-jar.xml</code> file is present in the specified EAR, JAR, or as a plain XML file. Empty the target directory and execute the TopLink migration tool again.
<code>toplink-ejb-jar.xml</code> is already defined in the archive, no migration needed.	A <code>toplink-ejb-jar.xml</code> file is already present in the target directory. The TopLink migration tool stops and copies the original input files into the target directory. Remove the <code>toplink-ejb-jar.xml</code> file from the target directory. Empty the target directory and execute the TopLink migration tool again.
The entity ( <code>ENTITY_NAME</code> ) in <code>orion-ejb-jar.xml</code> is not mapped as no table is specified. You need to provide the completely mapped <code>orion-ejb-jar.xml</code> to the migration tool. You can obtain the completely mapped <code>orion-ejb-jar.xml</code> from the <code>application-deployment</code> directory after deploying the application.	An entity is not mapped to database table. Before migration, confirm that you map all CMP entities in your application to one or more database tables in the <code>ejb-jar.xml</code> file, the <code>orion-ejb-jar.xml</code> file, or a combination of both.

### Unexpected Relational Multiplicity

The TopLink migration tool retrieves relationship multiplicity from the `orion-ejb-jar.xml` file and not from the OC4J `ejb-jar.xml` file.

Thus, even though the OC4J `ejb-jar.xml` file defines a relationship to be one-to-many, if the `orion-ejb-jar.xml` file defines the same relationship as many-to-many, then the TopLink migration tool will migrate the relationship as many-to-many.

## 8.4.3 How to Integrate JTA

For applications that require JTA integration, specify the external transaction controller when you configure the server platform in your session (see [Section 89.9, "Configuring the Server Platform"](#)).

For more information, see [Section 115.13, "Integrating the Unit of Work with an External Transaction Service"](#).

## 8.4.4 How to Integrate with Oracle Application Server Manageability and Diagnosability

Oracle recommends that you consider TopLink support for Oracle Application Server Manageability and Diagnosability to simplify application management and problem diagnosis.

For more information, see [Section A.1, "TopLink Support for Oracle Application Server Manageability and Diagnosability"](#).

## 8.5 Integrating TopLink with IBM WebSphere Application Server

To integrate a TopLink application with IBM WebSphere Application Server, you must consider the following:

- [How to Configure Classpath](#)

- [How to Configure Class Loader Order](#)
- [How to Integrate JTA](#)

In addition to configuring these IBM WebSphere application server-specific options, you must also consider the general application server integration issues in [Section 8.2, "Integrating TopLink with an Application Server"](#).

## 8.5.1 How to Configure Classpath

You configure the IBM WebSphere application server classpath differently depending on what version of this server you are using:

- [Configuring Classpath for IBM WebSphere Application Server 6.1 and Later](#)

### 8.5.1.1 Configuring Classpath for IBM WebSphere Application Server 6.1 and Later

TopLink provides JTA and general integration support for IBM WebSphere application server 6.1 and later. To configure the classpath for this version, do the following:

1. Create a shared library that contains the following Toplink JAR files and associate the shared library with the application:

```
<TOPLINK_HOME>\jlib\toplink.jar
```

2. Ensure that your TopLink application defines an XML parser platform (see [Section 8.2.2, "How to Configure the XML Parser Platform"](#)).

## 8.5.2 How to Configure Class Loader Order

If you are deploying a TopLink enabled application that uses TopLink `sessions.xml` or XML project deployment, you must use the WebSphere Application Server Administrative Console to set **Class loader order** to **Class loaded with application class loader first**.

## 8.5.3 How to Integrate JTA

For applications that require JTA integration, specify the external transaction controller when you configure the server platform in your session (see [Section 8.9.9, "Configuring the Server Platform"](#)).

For more information, see [Section 115.13, "Integrating the Unit of Work with an External Transaction Service"](#).

## 8.5.4 How to Configure Clustering on IBM WebSphere Application Server

For information on integrating a TopLink application with an application server cluster, see [Section 8.2.5, "How to Integrate Clustering"](#).

## 8.6 Integrating TopLink with Sun Application Server

To integrate a TopLink application with Sun Application Server (SunAS), you must consider the following:

- [How to Configure Classpath](#)
- [How to Integrate JTA](#)

In addition to configuring these SunAS-specific options, you must also consider the general application server integration issues in [Section 8.2, "Integrating TopLink with an Application Server"](#).

## 8.6.1 How to Configure Classpath

To configure TopLink support for SunAS, do the following:

1. Add the following JAR files to the application server classpath:  
`<TOPLINK_HOME>\jlib\toplink.jar`
2. Ensure that your TopLink application defines an XML parser platform (see [Section 8.2.2, "How to Configure the XML Parser Platform"](#)).

## 8.6.2 How to Integrate JTA

For applications that require JTA integration, specify the external transaction controller when you configure the server platform in your session (see [Section 89.9, "Configuring the Server Platform"](#)).

For more information, see [Section 115.13, "Integrating the Unit of Work with an External Transaction Service"](#).

## 8.7 Integrating TopLink with JBoss Application Server

To integrate a TopLink application with JBoss Application Server, you must consider the following:

- [How to Configure Classpath](#)
- [How to Integrate JTA](#)

In addition to configuring these JBoss-specific options, you must also consider the general application server integration issues in [Section 8.2, "Integrating TopLink with an Application Server"](#).

### 8.7.1 How to Configure Classpath

To configure TopLink support for JBoss, do the following:

1. Add the following JAR files to the application server classpath:  
`<TOPLINK_HOME>\jlib\toplink.jar`
2. Ensure that your TopLink application defines an XML parser platform (see [Section 8.2.2, "How to Configure the XML Parser Platform"](#)).

### 8.7.2 How to Integrate JTA

For applications that require JTA integration, specify the external transaction controller when you configure the server platform in your session (see [Section 89.9, "Configuring the Server Platform"](#)).

For more information, see [Section 115.13, "Integrating the Unit of Work with an External Transaction Service"](#).

### 8.7.3 How to Configure JPA Application Deployment to JBoss 4.2 Application Server

For JPA applications, to enable the container to manage entities, statically weave the entities and reference JBoss as the target server in the `persistence.xml` file.

Perform the following deployment changes:

1. If weaving is required, statically weave the entities before EAR packaging. Use either the command-line weaver or the weaving Ant task (see "How to Configure Static Weaving for JPA Entities" section of *EclipseLink Developer's Guide* at



[http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#How\\_to\\_Configure\\_Static\\_Weaving\\_for\\_JPA\\_Entities](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#How_to_Configure_Static_Weaving_for_JPA_Entities)).

2. Ensure that the `eclipselink.target-server` property (see "Using EclipseLink JPA Extensions for Session, Target Database and Target Application Server" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#Using\\_EclipseLink\\_JPA\\_Extensions\\_for\\_Session.2C\\_Target\\_Database\\_and\\_Target\\_Application\\_Server](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#Using_EclipseLink_JPA_Extensions_for_Session.2C_Target_Database_and_Target_Application_Server)) is set in the `persistence.xml` file of all persistence units deployed to the JBoss container:

```
<property name="eclipselink.target-server" value="JBoss"/>
```

Otherwise, even though the container-managed entities are predeployed, they will not be managed at run time.

For more information, see "How to Deploy an Application to Generic Java EE 5 Application Servers" section of *EclipseLink Developer's Guide* at

[http://wiki.eclipse.org/Packaging\\_and\\_Deploying\\_EclipseLink\\_JPA\\_Applications\\_%28ELUG%29#How\\_to\\_Deploy\\_an\\_Application\\_to\\_Generic\\_Java\\_EE\\_5\\_Application\\_Servers](http://wiki.eclipse.org/Packaging_and_Deploying_EclipseLink_JPA_Applications_%28ELUG%29#How_to_Deploy_an_Application_to_Generic_Java_EE_5_Application_Servers).

## 8.8 Defining Security Permissions

By default, when you run a TopLink-enabled application in a JVM configured with a nondefault `java.lang.SecurityManager`, the TopLink run time executes certain internal functions by executing a `PrivilegedAction` with `java.security.AccessController` method `doPrivileged`. This ensures that you do not need to grant many permissions to TopLink for it to perform its most common operations. You need only grant certain permissions depending on the types of optional TopLink features you use (see [Section 8.8.1, "How to Define Permissions Required by TopLink Features"](#)).

While using `doPrivileged` method provides enhanced security, it will severely impact overall performance. Alternatively, you can configure TopLink to disable the use of `doPrivileged` method even when a nondefault `SecurityManager` is present (see [Section 8.8.3, "How to Disable doPrivileged Operation"](#)). In this case, you must grant TopLink all required permissions (see [Section 8.8.1, "How to Define Permissions Required by TopLink Features"](#) and [Section 8.8.2, "How to Define Permissions Required when doPrivileged Is Disabled"](#)).

---



---

**Note:** While enabling the use of `doPrivileged` method enhances TopLink application security, it does not guarantee that secure code cannot be called by application code in ways that the system did not intend. You must consider the use of `doPrivileged` method within the context of your overall application security strategy. For more information, see <http://java.sun.com/security/index.jsp>.

---



---

If you run a TopLink-enabled application in a JVM without a nondefault `SecurityManager`, you do not need to grant any permissions.

### 8.8.1 How to Define Permissions Required by TopLink Features

When you run a TopLink-enabled application in a JVM configured with a nondefault `java.lang.SecurityManager` and `doPrivileged` operation is enabled, you may need to grant additional permissions if your application requires any of the following:

- [Defining System Properties](#)
- [Loading project.xml or sessions.xml Files](#)
- [Defining Cache Coordination](#)
- [Accessing a Data Source by Port](#)
- [Logging with java.util.logging](#)
- [Granting Permissions for Java EE Application Deployment](#)

### 8.8.1.1 Defining System Properties

By default, a TopLink-enabled application requires access to the system properties granted in the default `<JAVA_HOME>/lib/security/java.policy` file. If your application requires access to other platform-specific, environment, or custom properties, then grant further `PropertyPermission` permissions, as [Example 8-8](#) shows.

#### **Example 8-5 Permissions for System Properties**

```
permission java.util.PropertyPermission "my.property", "read";
```

### 8.8.1.2 Loading project.xml or sessions.xml Files

Most TopLink-enabled applications read in `project.xml` and `sessions.xml` files directly. Grant permissions to the specific files or file locations, as [Example 8-9](#) shows. This example assumes that both `project.xml` and `sessions.xml` files are located in the same directory (given by application-specific system property `deployment.xml.home`). Alternatively, you can specify a separate `FilePermission` for each file.

#### **Example 8-6 Permissions for Loading Deployment XML Files**

```
permission java.io.FilePermission "${deployment.xml.home}/*.xml", "read";
```

For information on `FilePermission` settings for Java EE applications, see [Section 8.8.1.6, "Granting Permissions for Java EE Application Deployment"](#).

### 8.8.1.3 Defining Cache Coordination

If your application uses cache coordination (see [Section 102.3, "Cache Coordination"](#)), then grant `accept`, `connect`, `listen`, and `resolve` permissions to the specific sockets used by your coordinated cache, as [Example 8-10](#) shows. This example assumes that the coordinated cache multicast port (see [Section 103.5, "Configuring a Multicast Port"](#)) is 1024.

#### **Example 8-7 Permissions for Cache Coordination**

```
permission java.net.SocketPermission "localhost:1024-", "accept, connect, listen, resolve";
```

### 8.8.1.4 Accessing a Data Source by Port

If your TopLink-enabled application accesses a data source using a socket, then grant `connect` and `resolve` permissions for that socket, as [Example 8-11](#) shows. This example assumes that the host name (or IP address) of the remote host that provides the data source (such as a relational database server host) is given by application-specific system property `remote.data.source.host` and that this host accepts data source connections on port 1025.

**Example 8–8 Permissions for non-Java EE Data Source Connections**

```
permission java.net.SocketPermission "${remote.data.source.host}:1025-", "connect, resolve";
```

For Java EE applications, data source socket permissions are usually handled by the application server.

**8.8.1.5 Logging with java.util.logging**

If you configure your TopLink-enabled application to use `java.util.logging` package (see [Section 89.4, "Configuring Logging"](#)), then grant your application control permissions, as [Example 8–12](#) shows.

**Example 8–9 Permissions for java.util.logging**

```
permission java.util.logging.LoggingPermission "control"
```

**8.8.1.6 Granting Permissions for Java EE Application Deployment**

If you are deploying a TopLink-enabled Java EE application, you must grant permissions for the following:

- The `toplink.jar` file. For example:

```
grant codeBase "file:<TOPLINK_HOME>/jlib/toplink.jar" {
    permission java.security.AllPermission;
};
```

If you are using an XML platform, you must also grant the following permissions:

- The `toplink.xml.platform` system property. For Example:

```
permission java.util.PropertyPermission "toplink.xml.platform", "read"
```

**8.8.2 How to Define Permissions Required when doPrivileged Is Disabled**

If you disable `doPrivileged` operation when you run a TopLink-enabled application in a JVM configured with a nondefault `java.lang.SecurityManager`, you must grant the following permissions:

- `java.lang.reflect.ReflectPermission "suppressAccessChecks"`
- `java.lang.RuntimePermission "accessDeclaredMembers"`
- `java.lang.RuntimePermission "getClassLoader"`

You may also have to grant additional permissions depending on the TopLink features your application uses. For more information, see [Section 8.8.1, "How to Define Permissions Required by TopLink Features"](#).

**8.8.3 How to Disable doPrivileged Operation**

To disable `doPrivileged` operation when you run a TopLink-enabled application in a JVM configured with a nondefault `java.lang.SecurityManager`, set system property `oracle.j2ee.toplink.security.usedoprivileged` to `false`. If you are using OC4J, set system property `oracle.j2ee.security.usedoprivileged` to `false`.

To enable `doPrivileged` operation, set these system properties to `true`.

**8.9 Configuring Miscellaneous EJB CMP Options**

TopLink provides system properties that you can use to customize the following EJB CMP options:

- Setter Parameter Type Checking (see [Section 8.9.1, "How to Configure EJB CMP Setter Parameter Type Checking"](#))
- Unknown Primary Key Class Support (see [Section 8.9.2, "How to Configure EJB CMP Unknown Primary Key Class Support"](#))
- Single-Object Finder Return Type Checking (see [Section 8.9.3, "How to Configure EJB CMP Single-Object Finder Return Type Checking"](#))

### 8.9.1 How to Configure EJB CMP Setter Parameter Type Checking

To make TopLink verify that the parameters to one-to-one and one-to-many relationship setters are of the same type as the corresponding CMR field, set system property `toplink.cts.collection.checkParameters` to a value of `true` (not case sensitive). If the setters are not the same type, then TopLink throws a `java.lang.IllegalArgumentException`.

---

---

**Note:** Setting this property to `true` will affect performance. Use this setting only if necessary.

---

---

If you set the property to `false` (the default value), TopLink does not make this verification. In this case, it is up to your application to make sure the parameters are of the correct type.

For more information, see Section 10.3.6 of the EJB 2.1 specification.

### 8.9.2 How to Configure EJB CMP Unknown Primary Key Class Support

In special situations, you may choose not to specify the primary key class or the primary key fields for an entity bean with container-managed persistence. For example, if the entity bean does not have a natural primary key or you want the deployer to select the primary key fields at deployment time, you may choose to defer primary key type specification.

If this is the case, you must declare the type of the argument of the `findByPrimaryKey` method as `java.lang.Object` and you must also specify the primary key class (`prim-key-class`) in the deployment descriptor (`ejb-jar.xml`) as `java.lang.Object`.

TopLink provides run-time support for such deferred primary key type specification.

For more information, see Section 10.8.3 of the EJB 2.1 specification.

### 8.9.3 How to Configure EJB CMP Single-Object Finder Return Type Checking

By setting system property `toplink.cts.checkMultipleRows` to `true`, you can configure TopLink to throw a `javax.ejb.FinderException` if multiple beans are returned from a single-object finder method.

For more information, see Section 10.5.6.1 of the EJB 2.1 specification.

---

---

## Creating TopLink Files for Deployment

This chapter includes TopLink information that you need when creating deployment files for various types of applications.

This chapter includes the following sections:

- Introduction to the TopLink Deployment File Creation
- Creating Deployment Files for Java Applications
- Creating Deployment Files for JavaServer Pages and Servlet Applications
- Creating Deployment Files for Session Bean Applications
- Creating Deployment Files for JPA Applications
- Creating Deployment Files for CMP Applications
- Creating Deployment Files for BMP Applications
- Configuring the weblogic-ejb-jar.xml File for Oracle WebLogic Server
- Configuring the orion-ejb-jar.xml File for OC4J

For more information on packaging and deployment, see the following:

- Section 9.1, "Introduction to the TopLink Deployment File Creation"
- Chapter 8, "Integrating TopLink with an Application Server"
- Chapter 10, "Packaging a TopLink Application"
- Chapter 11, "Deploying a TopLink Application"
- "Packaging and Deploying EclipseLink JPA Applications" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Packaging\\_and\\_Deploying\\_EclipseLink\\_JPA\\_Applications\\_%28ELUG%29](http://wiki.eclipse.org/Packaging_and_Deploying_EclipseLink_JPA_Applications_%28ELUG%29).

### 9.1 Introduction to the TopLink Deployment File Creation

Depending on the type of application you are deploying, you may need to create any of the following deployment files:

- project.xml File
- sessions.xml File
- ejb-jar.xml File
- JAVA-EE-CONTAINER-ejb-jar.xml File
- toplink-ejb-jar.xml File

TopLink Workbench provides the ability to create deployment files from a TopLink Workbench project (see [Section 116.3, "Exporting Project Information"](#)). After you build a project, you have two options to create the deployment files:

- Create XML deployment files that require no compiling.
- Create Java source files, which you compile and deploy outside of TopLink Workbench.

Oracle recommends XML deployment because XML files are easier to deploy and troubleshoot than compiled Java files. This approach gives you a very flexible configuration that enables you to make changes safely and easily. XML deployment files do not require third-party applications or compilers to deploy successfully.

---

---

**Note:** If you are using EJB 3.0, you can use annotations to specify most of what you formerly specified in deployment descriptors. Use deployment descriptors to override annotations or specify options not supported by annotations.

---

---

### 9.1.1 project.xml File

The `project.xml` file is the core of your application. It contains the descriptors and mappings you define and also includes any named queries or finders associated with your project.

This section describes the following:

- [XSD File Format](#)
- [POJO Applications and Project Metadata](#)
- [JPA Applications and Project Metadata](#)
- [CMP Applications and Project Metadata](#)
- [Creating the project.xml File with Oracle JDeveloper](#)
- [Creating the project.xml File with TopLink Workbench](#)
- [Creating project.xml Programmatically](#)

#### 9.1.1.1 XSD File Format

The `project.xml` file XSD is `toplink-object-persistence_11_1_1.xsd` and it is located in the `<TOPLINK_HOME>\xsds` directory.

#### 9.1.1.2 POJO Applications and Project Metadata

For a POJO application, you define your project metadata in a `project.xml` file.

The `project.xml` file provides a simple and flexible way to configure, modify, and troubleshoot the project metadata. Because of these attributes, the `project.xml` file is the preferred way to configure a TopLink project.

TopLink Workbench provides a graphical tool to build and edit the `project.xml` file. For information on creating projects with TopLink Workbench, see [Section 9.1.1.6, "Creating the project.xml File with TopLink Workbench"](#).

#### 9.1.1.3 JPA Applications and Project Metadata

For a JPA application, you can express project metadata using JPA annotations, `persistence.xml`, `orm.xml`, and EclipseLink JPA annotation and

`persistence.xml` property extensions. The EclipseLink JPA persistence provider interprets all these sources of metadata to create an in-memory session and project at run time.

Using EclipseLink JPA, you also have the option of specifying your metadata using `sessions.xml` and `project.xml` while accessing your persistent classes using JPA and an `EntityManager`. For more information, see "What You May Need to Know About EclipseLink JPA Overriding Mechanisms" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#What\\_You\\_May\\_Need\\_to\\_Know\\_About\\_EclipseLink\\_JPA\\_Overriding\\_Mechanisms](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#What_You_May_Need_to_Know_About_EclipseLink_JPA_Overriding_Mechanisms).

#### 9.1.1.4 CMP Applications and Project Metadata

For a CMP application, how you specify project metadata is dependent upon the Java EE application server you are deploying your application (see [Section 9.1.5](#), "toplink-ejb-jar.xml File").

#### 9.1.1.5 Creating the project.xml File with Oracle JDeveloper

In Oracle JDeveloper, TopLink mapping project information is maintained in the TopLink Map. For more information, see the Oracle JDeveloper online help.

#### 9.1.1.6 Creating the project.xml File with TopLink Workbench

Because you must synchronize the `project.xml` file with the classes and data source associated with your application, Oracle recommends that you not modify this file manually. TopLink Workbench ensures proper synchronization, and is the best way to make changes to the project. Simply modify the project in TopLink Workbench and redeploy the `project.xml` file. Using this option reduces development time by eliminating the need to regenerate and recompile Java code each time the project changes.

See [Section 116.3](#), "Exporting Project Information" for detailed information on exporting the deployment XML information.

---



---

**Note:** You can name this file with a name other than `project.xml`; however, for clarity, this discussion assumes that the file has not been renamed.

---



---

#### 9.1.1.7 Creating project.xml Programmatically

Optionally, you can use the `DeploymentXMLGenerator` API to programmatically generate the `project.xml` file in either of the following ways:

- From an application, instantiate the `DeploymentXMLGenerator` and your java source. Call the following method:

```
generate (<MW_Project.mwp>, <output file.xml>)
```

- From the command line, use the following:

```
java -classpath
toplink.jar;toplinkmw.jar;xmlparserv2.jar;ejb.jar;oracle.toplink.workbench.mappings.DeploymentXMLGenerator <MW_Project.mwp> <output file.xml>
```

Before you use either method, ensure that your classpath includes the `<TOPLINK_HOME>\xsds` directory.

---

---

**Note:** If you are using EJB 3.0, you can use annotations to specify most of what you formerly specified in the `project.xml` file. To override annotations or specify options not supported by annotations, you can still provide a `project.xml` file in your EJB 3.0 application. For more information on what annotations are currently supported, see *Oracle Fusion Middleware Enterprise JavaBeans Developer's Guide for Oracle Containers for Java EE*

---

---

## 9.1.2 sessions.xml File

Each TopLink project belongs to a TopLink *session*. A session is the facade through which an application accesses TopLink functionality (for more information on sessions, see [Part XXI, "TopLink Sessions"](#)). Where you define a session differs depending on whether or not your application uses CMP.

This section describes the following:

- [XSD File Format](#)
- [POJO Applications and Session Metadata](#)
- [JPA Applications and Session Metadata](#)
- [CMP Applications and Session Metadata](#)

### 9.1.2.1 XSD File Format

The `sessions.xml` file XSD is `sessions_11_1_1.xsd` and it is located in the `<TOPLINK_HOME>\xsds` directory.

When you use the XSD formatted `sessions.xml` file, the TopLink run time separates `sessions.xml` file validation from session instantiation. Separating XML file formatting problems from Session Manager session instantiation problems simplifies troubleshooting. Exceptions thrown during validation clearly indicate that the failure is due to an invalid `sessions.xml` file, as [Example 9-1](#) illustrates.

#### **Example 9-1 Enhanced Validation Exceptions**

```
Exception [TOPLINK-9010] (Oracle TopLink - 10g (10.0.3) (Build 040127Dev)):  
oracle.toplink.exceptions.SessionLoaderException  
Exception Description: A End tag does not match start tag 'session'. was thrown while parsing  
the XML file against the XML schema.  
Internal Exception: oracle.xml.parser.v2.XMLParseException: End tag does not match start tag  
'session'.
```

### 9.1.2.2 POJO Applications and Session Metadata

For a POJO application, you define your sessions in a `sessions.xml` file.

The `sessions.xml` file provides a simple and flexible way to configure, modify, and troubleshoot the application sessions. Because of these attributes, the `sessions.xml` file is the preferred way to configure a TopLink session.

TopLink provides graphical tools to build and edit the `sessions.xml` file. For information see [Chapter 88, "Creating a Session"](#).

### 9.1.2.3 JPA Applications and Session Metadata

For a JPA application, you can express session metadata using JPA annotations, `persistence.xml`, `orm.xml`, and EclipseLink JPA annotation and `persistence.xml` property extensions. The EclipseLink JPA persistence provider



interprets all these sources of metadata to create an in-memory TopLink session and project at run time.

Using EclipseLink JPA, you also have the option of specifying your metadata using `sessions.xml` and `project.xml` while accessing your persistent classes using JPA and an `EntityManager`. For more information, see "What You May Need to Know About EclipseLink JPA Overriding Mechanisms" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#What\\_You\\_May\\_Need\\_to\\_Know\\_About\\_EclipseLink\\_JPA\\_Overriding\\_Mechanisms](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#What_You_May_Need_to_Know_About_EclipseLink_JPA_Overriding_Mechanisms).

#### 9.1.2.4 CMP Applications and Session Metadata

For a CMP project, how you specify session metadata is dependent upon the Java EE application server to which you are deploying your application.

For OC4J, the session configuration is done in the `orion-ejb-jar.xml` file. You can specify the `data-source`, some common session options, and a session customizer class (see Table 9-3, "orion-ejb-jar.xml File persistence-manager Subentries for pm-properties"). In this case, you name the TopLink project XML file as `toplink-ejb-jar.xml` (see Section 9.1.1, "project.xml File")

---



---

**Note:** If you are using EJB 3.0, you cannot use annotations to specify session configuration. You must provide a `sessions.xml` file, if one is applicable to your application type.

---



---

### 9.1.3 ejb-jar.xml File

Each EJB module contains one `ejb-jar.xml` file that describes all the EJB in the module.

Most IDEs provide facilities to create the `ejb-jar.xml` file. For more information about generating this file from your IDE, see your IDE documentation.

If you build an EJB application, Oracle recommends that you use TopLink Workbench to build the `ejb-jar.xml` file. Because TopLink Workbench can both read and write the `ejb-jar.xml` file, you can use TopLink Workbench to maintain your `ejb-jar.xml` file in the following ways:

- When you change the file manually outside of TopLink Workbench, reimport the `ejb-jar.xml` file into TopLink Workbench project to refresh the project.
- When you change the TopLink Workbench project, TopLink Workbench updates the `ejb-jar.xml` file automatically when you save the project.

For more information about managing the `ejb-jar.xml` file in TopLink Workbench, see Section 19.7, "Working with the ejb-xml.File".

---



---

**Note:** If you are using EJB 3.0, you can use annotations to specify most of what you formerly specified in the `ejb-jar.xml` file. To override annotations or specify options not supported by annotations, you can still provide an `ejb-jar.xml` file in your EJB 3.0 application.

---



---

## 9.1.4 JAVA-EE-CONTAINER-ejb-jar.xml File

The contents of the `JAVA-EE-CONTAINER-ejb-jar.xml` file depend on the container to which you deploy your EJB. To create this file, use the tools that accompany your container.

In most cases, the `JAVA-EE-CONTAINER-ejb-jar.xml` file integrates with TopLink without revision. However, in some cases, you must make some TopLink-specific modifications.

---

**Note:** If you are using EJB 3.0, you can use annotations to specify most of what you formerly specified in the `JAVA-EE-CONTAINER-ejb-jar.xml` file. To override annotations or specify options not supported by annotations, you can still provide a `JAVA-EE-CONTAINER-ejb-jar.xml` file in your EJB 3.0 application. For more information on what annotations are currently supported, see *Oracle Fusion Middleware Enterprise JavaBeans Developer's Guide for Oracle Containers for Java EE*.

---

For more information, see the following:

- [Section 9.1.4.1, "Oracle WebLogic Server and the weblogic-ejb-jar.xml File"](#)
- [Section 9.1.4.2, "OC4J and the orion-ejb-jar.xml File"](#)

### 9.1.4.1 Oracle WebLogic Server and the weblogic-ejb-jar.xml File

For more information on configuring the `weblogic-ejb-jar.xml`, see [Section 9.8, "Configuring the weblogic-ejb-jar.xml File for Oracle WebLogic Server"](#).

### 9.1.4.2 OC4J and the orion-ejb-jar.xml File

[Table 9–1](#) summarizes the scenarios in which you may choose to modify the `orion-ejb-jar.xml` file.

**Table 9–1** When to Modify the `orion-ejb-jar.xml` File

CMP Type	Mapping Type	Action
Orion	Specified in <code>orion-ejb-jar.xml</code>	1. Deploy.
Orion	Default mappings	1. Edit the <code>orion-ejb-jar.xml</code> file to set persistence-manager attribute name to <code>orion</code> . 2. Deploy.
Toplink	Specified in <code>toplink-ejb-jar.xml</code>	1. Deploy.
Toplink	Specified in <code>toplink-ejb-jar.xml</code> (custom persistence manager properties)	1. Edit the <code>orion-ejb-jar.xml</code> file to set persistence-manager attribute name to <code>toplink</code> . 2. Edit additional persistence-manager subentries (see <a href="#">Section 9.9, "Configuring the orion-ejb-jar.xml File for OC4J"</a> ). 3. Deploy.
Toplink	Default mappings (no <code>toplink-ejb-jar.xml</code> )	1. Deploy.

For more information on configuring the `orion-ejb-jar.xml` file, see [Section 9.9, "Configuring the orion-ejb-jar.xml File for OC4J"](#).

### 9.1.5 toplink-ejb-jar.xml File

The `toplink-ejb-jar.xml` file is used only in CMP projects. The TopLink runtime uses properties set in the `JAVA-EE-CONTAINER-ejb-jar.xml` file (see [Section 9.1.4, "JAVA-EE-CONTAINER-ejb-jar.xml File"](#)) to locate the `toplink-ejb-jar.xml` file and read it in.

The purpose of `toplink-ejb-jar.xml` file depends on the type of application server you are using.

#### 9.1.5.1 OC4J and the toplink-ejb-jar.xml File

When deploying a CMP application to OC4J, the `toplink-ejb-jar.xml` file is the name used for the `project.xml` file.

To create the `toplink-ejb-jar.xml` file in this case, simply rename your `project.xml` file. For more information, see [Section 9.1.1, "project.xml File"](#).

## 9.2 Creating Deployment Files for Java Applications

In a Java application, TopLink does not use a Java EE container for deployment. Instead, it relies on TopLink mechanisms to provide functionality and persistence. The key elements of this type of application are the lack of a Java EE container and the fact that you deploy the application by placing the application JAR file on the classpath.

Java applications require the following deployment files:

- [project.xml File](#)
- [sessions.xml File](#)

## 9.3 Creating Deployment Files for JavaServer Pages and Servlet Applications

Many designers build TopLink applications that use JavaServer Pages (JSP) and Java servlets. This type of design generally supports Web-based applications.

JSP and servlet applications require the following deployment files:

- [project.xml File](#)
- [sessions.xml File](#)

## 9.4 Creating Deployment Files for Session Bean Applications

Session beans generally model a process, operation, or service and as such, are not persistent. You can build TopLink applications that wrap interaction with TopLink in session beans. Session beans execute all TopLink-related operations on behalf of the client.

This type of design uses JTS and externally managed transactions, but does not incur the overhead associated with persistence applications. Session bean applications also scale and deploy easily.

This section describes the following:

- [How to Create Deployment Files for EJB 1.n and 2.n Session Bean Applications](#)
- [How to Create Deployment Files for EJB 3.0 Session Bean Applications](#)

## 9.4.1 How to Create Deployment Files for EJB 1.*n* and 2.*n* Session Bean Applications

EJB 1.*n* and 2.*n* session bean applications require the following deployment files:

- [project.xml File](#)
- [sessions.xml File](#)

## 9.4.2 How to Create Deployment Files for EJB 3.0 Session Bean Applications

Oracle recommends using JPA annotations and persistence unit properties, or a special-case `eclipselink.session-xml` persistence unit property (see "EclipseLink JPA Persistence Unit Properties for Database, Session, and Application Server" table of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#Using\\_EclipseLink\\_JPA\\_Extensions\\_for\\_Session.2C\\_Target\\_Database\\_and\\_Target\\_Application\\_Server](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#Using_EclipseLink_JPA_Extensions_for_Session.2C_Target_Database_and_Target_Application_Server)) in your EJB 3.0 session bean application.

You may also choose to use the [project.xml File](#) and [sessions.xml File](#).

For more information, see the following:

- "Java Persistence API Overview" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Introduction\\_to\\_Java\\_Persistence\\_API\\_%28ELUG%29](http://wiki.eclipse.org/Introduction_to_Java_Persistence_API_%28ELUG%29)
- "Introduction to EclipseLink JPA" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Introduction\\_to\\_EclipseLink\\_JPA\\_%28ELUG%29](http://wiki.eclipse.org/Introduction_to_EclipseLink_JPA_%28ELUG%29)
- [http://wiki.eclipse.org/EclipseLink/UserGuide/Developing\\_JPA\\_Projects\\_%28ELUG%29](http://wiki.eclipse.org/EclipseLink/UserGuide/Developing_JPA_Projects_%28ELUG%29)

## 9.5 Creating Deployment Files for JPA Applications

See "Packaging and Deploying EclipseLink JPA Applications" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Packaging\\_and\\_Deploying\\_EclipseLink\\_JPA\\_Applications\\_%28ELUG%29](http://wiki.eclipse.org/Packaging_and_Deploying_EclipseLink_JPA_Applications_%28ELUG%29) for information on how to create deployment files for your JPA application.

## 9.6 Creating Deployment Files for CMP Applications

Many applications use the persistence mechanisms a Java EE container offers. TopLink provides full support for this type of application.

You can only use one persistence manager for all the entity beans with container-managed persistence in a JAR file.

CMP applications require the following deployment files:

- `ejb-jar.xml` (see [Section 9.1.3, "ejb-jar.xml File"](#))
- `JAVA-EE-CONTAINER-ejb-jar.xml` (see [Section 9.1.4, "JAVA-EE-CONTAINER-\*ejb-jar.xml\* File"](#))
- `toplink-ejb-jar.xml` (see [Section 9.1.5, "toplink-\*ejb-jar.xml\* File"](#))

## 9.7 Creating Deployment Files for BMP Applications

If you choose to write your own persistence code with BMP, you can take advantage of the classes in `oracle.toplink.ejb.bmp` package. Whether or not you use these classes, BMP applications require the following deployment files:

- `project.xml` (see [Section 9.1.1, "project.xml File"](#))
- `sessions.xml` (see [Section 9.1.2, "sessions.xml File"](#))
- `ejb-jar.xml` (see [Section 9.1.3, "ejb-jar.xml File"](#))

## 9.8 Configuring the weblogic-ejb-jar.xml File for Oracle WebLogic Server

Before you deploy a TopLink application to a Oracle WebLogic Server, you must modify the `weblogic-ejb-jar.xml` file.

Avoid the `weblogic-ejb-jar.xml` tags that TopLink either does not support or does not require (see [Section 9.8.1, "What You May Need to Know About Unsupported weblogic-ejb-jar.xml File Tags"](#)).

### 9.8.1 What You May Need to Know About Unsupported weblogic-ejb-jar.xml File Tags

The `weblogic-ejb-jar.xml` file includes the following tags that TopLink either does not support or does not require:

- `concurrency-strategy`: This tag specifies how Oracle WebLogic Server manages concurrent users for a given bean. Because TopLink manages concurrent access internally, it does not require this tag.

For more information about the TopLink concurrency strategy, see [Section 119.26, "Configuring Locking Policy"](#).

- `db-is-shared`: Because TopLink does not make any assumptions about the exclusivity of database access, TopLink does not require this tag. TopLink addresses multiuser access issues through various locking and refreshing policies.
- `delay-updates-until-end-of-tx`: TopLink always delays updates until the end of a transaction, and does not require this tag.
- `finders-load-bean`: TopLink always loads the bean upon execution of the finder, and does not require this tag.
- `pool`: TopLink does not use a pooling strategy for entity beans. This avoids object-identity problems that can occur due to pooling.
- `lifecycle`: This element manages beans that follow a pooling strategy. Because TopLink does not use a pooling strategy, TopLink ignores this tag.
- `is-modified-method-name`: TopLink does not require a bean developer-defined method to detect changes in the object state.
- `isolation-level`: Because isolation level settings for the cache or database transactions are specified in the TopLink project, TopLink ignores this tag.
- `cache`: Because you define TopLink cache properties in TopLink Workbench, this tag is unnecessary.

## 9.9 Configuring the orion-ejb-jar.xml File for OC4J

To deploy a TopLink application to OC4J, modify the `orion-ejb-jar.xml` file. For more information, see [Section 9.9.1, "How to Configure persistence-manager Entries"](#).

If you are migrating an application from a previous release of OC4J, you can use the TopLink migration tool to automatically migrate persistence information from your `orion-ejb-jar.xml` file to a new `toplink-ejb-jar.xml`. For more information, see [Section 8.4.2, "How to Migrate OC4J Orion CMP Persistence to OC4J TopLink Persistence"](#).

## 9.9.1 How to Configure persistence-manager Entries

If you are using TopLink as your OC4J persistence manager, you can configure the `persistence-manager` subentry (see [Table 9-2](#)) in the `orion-ejb-jar.xml` file. For more information on the scenarios in which you would want to modify `orion-ejb-jar.xml`, see [Section 9.1.4.2, "OC4J and the orion-ejb-jar.xml File"](#).

If you are not using TopLink as your OC4J persistence manager, do not modify the `persistence-manager` subentries.

OC4J does not support `entity-deployment` attribute `pm-name`. Use `persistence-manager` attribute name instead (see [Table 9-2](#)). When OC4J parses the `orion-ejb-jar.xml` file, if it finds a `pm-name` attribute, OC4J ignores its value and logs the following warning message:

---



---

**WARNING:** Use of `pm-name` is unsupported and will be removed in a future release. Specify `pm` usage using `<persistence-manager> 'name'` instead.

---



---

**Table 9-2** *orion-ejb-jar.xml File persistence-manager Entries*

Entry	Description
<code>name</code>	The name of the persistence manager to use. Set this value to <code>toplink</code> . If you set the name property to <code>toplink</code> , you may also configure <code>pm-properties</code> (see <a href="#">Section 9.9.1.1, "Configuring pm-properties"</a> ).
<code>class-name</code>	Do not configure this attribute. If <code>name</code> is set to <code>toplink</code> , then <code>class-name</code> is set correctly by default.
<code>descriptor</code>	This property applies only when <code>name</code> is set to <code>toplink</code> . If you export your TopLink mapping metadata to a deployment XML file, set this property to the name of the deployment XML file (default: <code>toplink-ejb-jar.xml</code> ). Do not set this property if you are using a TopLink project class instead of a mapping metadata file (see <code>project-class</code> in <a href="#">Table 9-3</a> ).

### 9.9.1.1 Configuring pm-properties

When you select TopLink as the persistence manager (see `name` in [Table 9-2](#)), use the `persistence-manager` subentries for `pm-properties` (see [Table 9-3](#)) to configure the TopLink session that the TopLink runtime creates and uses internally for CMP projects. The `persistence-manager` subentries take the place of a `sessions.xml` file in a CMP project.

---



---

**Note:** You can only configure a subset of session features using these properties and in most cases, default configuration applies. To configure all session features and to override defaults, you must use a customization class (see `customization-class` in [Table 9-3](#)).

---



---

**Table 9–3** *orion-ejb-jar.xml File persistence-manager Subentries for pm-properties*

Entry	Description
session-name	<p>Unique name for this TopLink-persisted EJB deployment JAR file. Must be unique among all TopLink-persisted deployed JAR files in this application server instance.</p> <p>When the TopLink run time internally creates a TopLink session for this TopLink-persisted deployed JAR file, the TopLink session manager stores the session instance under this <code>session-name</code>. For more information about the session manager, see <a href="#">Chapter 90, "Acquiring and Using Sessions at Run Time"</a>.</p> <p>If you do not specify a name, the TopLink runtime will generate a unique name.</p>
project-class	<p>If you export your TopLink mapping metadata to a Java class (that extends <code>oracle.toplink.sessions.Project</code>), set this property to the name of the class, fully qualified by its package name. Be sure to include the class file in the deployable JAR file.</p> <p>Do not set this property if you are using a mapping metadata file (see descriptor in <a href="#">Table 9–2</a>).</p>
customization-class	<p>Optional Java class (that implements <code>oracle.toplink.ejb.cmp.DeploymentCustomization</code>) used to allow deployment customization of TopLink mapping and run-time configuration. At deployment time, the TopLink run time creates a new instance of this class and invokes its methods <code>beforeLoginCustomization</code> (before the TopLink run time logs into the session) and <code>afterLoginCustomization</code> (after the TopLink runtime logs into the session), passing in the TopLink session as a parameter.</p> <p>Use your implementation of the <code>beforeLoginCustomization</code> method to configure session attributes not supported by the <code>pm-properties</code> including: cache coordination (see also <a href="#">Section 9.9.1.2, "Configuring cache-synchronization Properties"</a>), parameterized SQL, native SQL, batch writing/batch size, byte-array/string binding, EIS login, event listeners, table qualifier, and sequencing. For more information about session configuration, see <a href="#">Chapter 89, "Configuring a Session"</a>.</p> <p>The class must be fully qualified by its package name and included in the deployment JAR file.</p>
db-platform-class	<p>Optional TopLink database platform class (instance of <code>oracle.toplink.platform.database</code> or <code>oracle.toplink.platform.database.oracle</code>) containing TopLink support specific to a particular database.</p> <p>Set this value to the database platform class that corresponds to the database that your application uses. The class must be fully qualified by its package name.</p>
remote-relationships	<p>Optional flag to allow relationships between remote objects. Valid values are:</p> <ul style="list-style-type: none"> <li>■ <code>true</code>: All relationships will be maintained through the remote interfaces of the entity beans</li> <li>■ <code>false</code>: Disables this feature.</li> </ul>
cache-synchronization	See <a href="#">Section 9.9.1.2, "Configuring cache-synchronization Properties"</a> .
default-mapping	See <a href="#">Section 9.9.1.3, "Configuring default-mapping Properties"</a> .

### 9.9.1.2 Configuring cache-synchronization Properties

When you select TopLink as the persistence manager (see name in [Table 9–2](#)), use the `pm-properties` subentry for `cache-synchronization` (see [Table 9–4](#)) to configure TopLink cache coordination features of the session that the TopLink runtime uses internally for CMP projects. For more information about TopLink cache coordination, see [Section 102.3, "Cache Coordination"](#).

When this subentry is present, you must use a customization class (see `customization-class` in [Table 9–3](#)) to complete cache coordination configuration.

For more information about TopLink cache coordination configuration, see [Chapter 103, "Configuring a Coordinated Cache"](#).

**Table 9–4** *orion-ejb-jar.xml* File *pm-properties* Subentries for cache-synchronization

Entry	Description
mode	An indicator of whether or not cache coordination updates should be propagated to other servers synchronously or asynchronously. Valid values are as follows: <ul style="list-style-type: none"> <li>▪ asynchronous (default)</li> <li>▪ synchronous</li> </ul>
server-url	<p><b>For a JMS coordinated cache:</b> assuming that you are using the OC4J JNDI naming service and that all the hosts in your coordinated cache can communicate using OC4J proprietary RMI protocol ORMI, use a URL similar to the following:</p> <pre>ormi://&lt;JMS-host-IP&gt;:&lt;JMS-host-port&gt;</pre> <p>where <code>JMS-host-IP</code> is the IP address of the host on which the JMS service provider is running, and <code>JMS-host-port</code> is the port on which the JMS service provider is listening for JMS requests.</p> <p><b>For an RMI or CORBA coordinated cache:</b> assuming that you are using the OC4J JNDI naming service and that all the hosts in your coordinated cache can communicate using OC4J proprietary RMI protocol ORMI on OC4J default port 23791, use a URL similar to the following:</p> <pre>ormi://&lt;session-host-IP&gt;:23791</pre> <p>where <code>session-host-IP</code> is the IP address of the host on which this session is deployed.</p>
server-user	Optional user name required to log in to the JNDI naming service.

### 9.9.1.3 Configuring default-mapping Properties

When you select TopLink as the persistence manager (see name in [Table 9–2](#)), use the `pm-properties` subentry for `default-mapping` (see [Table 9–5](#)) to configure the TopLink default mapping and automatic table generation feature.

For more information about TopLink default mappings, see [Section 17.2.3.4, "Default Mapping in EJB 2.n CMP Projects Using OC4J at Run Time"](#).

For more information about TopLink automatic table generation, see [Section 6.4, "Creating Database Tables Automatically"](#).



**Table 9–5** *orion-ejb-jar.xml File pm-properties Subentries for default-mapping*

Entry	Description
db-table-gen	<p>Optional element that determines what TopLink will do to prepare the database tables that are being mapped to. Valid values are:</p> <ul style="list-style-type: none"> <li>▪ <b>Create</b> (default): This value tells TopLink to create the mapped tables during the deployment. If the tables already exist, TopLink will log an appropriate warning messages (such as "Table already existed...") and keeps processing the deployment.</li> <li>▪ <b>DropAndCreate</b>: This value tells TopLink to drop tables before creating them during deployment. If a table does not initially exist, the drop operation will cause anSQLException to be thrown through the driver. However, TopLink handles the exception (logs and ignores it) and moves on to process the table creation operation. The deployment fails only if both drop and create operations fail.</li> <li>▪ <b>UseExisting</b>: This value tells TopLink to perform no table manipulation. If the tables do not exist, deployment still goes through without error.</li> </ul> <p>If no <code>orion-ejb-jar.xml</code> file is defined in your EAR file, the OC4J container generates one during deployment. In this case, to specify a value for <code>db-table-gen</code>, use the TopLink system property <code>toplink.defaultmapping.dbTableGenSetting</code>. For example:  <code>-Dtoplink.defaultmapping.dbTableGenSetting="DropAndCreate"</code>.</p> <p>The <code>orion-ejb-jar.xml</code> property overrides the system property. If both the <code>orion-ejb-jar.xml</code> property and the system property are present, TopLink retrieves the setting from the <code>orion-ejb-jar.xml</code> file.</p> <p>This setting overrides <code>autocreate-tables</code> and <code>autodelete-tables</code> configuration at the application (EAR) or system level. For more information, see <a href="#">Section 6.4, "Creating Database Tables Automatically"</a>.</p>
extended-table-names	<p>An element used if the generated table names are not long enough to be unique. Values are restricted to <code>true</code> or <code>false</code> (default). When set to <code>true</code>, the TopLink run time will ensure that generated tables names are unique.</p> <p>In default mapping, each entity is mapped to one table. The only exception is in many-to-many mappings where there is one extra relation table involved in the source and target entities.</p> <p>When <code>extended-table-names</code> is set to <code>false</code> (the default), a simple table naming algorithm is used as follows: table names are defined as <code>TL_&lt;bean_name&gt;</code>. For example, if the bean name is <code>Employee</code>, the associated table name would be <code>TL_EMPLOYEE</code>.</p> <p>However, if the same entity is defined in multiple JAR files in an application, or across multiple applications, table-naming collision is inevitable.</p> <p>To address this problem, set <code>extended-table-names</code> to <code>true</code>. When set to <code>true</code>, TopLink uses an alternative table-naming algorithm as follows: table names are defined as <code>&lt;bean_name&gt;_&lt;jar_name&gt;_&lt;app_name&gt;</code>. This algorithm uses the combination of bean, JAR, and EAR names to form a table name unique across the application. For example, given a bean named <code>Employee</code>, which is in <code>Test.jar</code>, which is in <code>Demo.ear</code> (and the application name is "Demo"), then the corresponding table name will be <code>EMPLOYEE_TEST_DEMO</code>.</p> <p>If there is no <code>orion-ejb-jar.xml</code> file defined in the EAR file, the OC4J container generates one during deployment. In this case, to specify a value for <code>extended-table-names</code>, use the TopLink system property <code>toplink.defaultmapping.useExtendedTableNames</code>. For example:  <code>-Dtoplink.defaultmapping.useExtendedTableNames="true"</code>.</p> <p>The <code>orion-ejb-jar.xml</code> property overrides the system property. If both the <code>orion-ejb-jar.xml</code> property and the system property are present, TopLink retrieves the setting from the <code>orion-ejb-jar.xml</code> file.</p>



---

---

## Packaging a TopLink Application

How you package the components of your application depends on the type of application and how you plan to deploy it.

This chapter describes TopLink-specific details applicable to the common packaging strategies used for various types of applications.

This chapter includes the following sections:

- [Packaging Java Applications](#)
- [Packaging JavaServer Pages and Servlet Applications](#)
- [Packaging Session Bean Applications](#)
- [Packaging JPA Applications](#)
- [Packaging a POJO Application for Weaving](#)
- [Packaging CMP Applications](#)
- [Packaging BMP Applications](#)
- [Packaging with TopLink Metadata File Resource Paths](#)

---

---

**Note:** If you are using EJB 3.0, you may be using annotations instead of some deployment files. Include deployment descriptors to override annotations or specify options not supported by annotations.

---

---

For more information, see the following:

- [Chapter 8, "Integrating TopLink with an Application Server"](#)
- [Chapter 9, "Creating TopLink Files for Deployment"](#)
- [Chapter 11, "Deploying a TopLink Application"](#)

### 10.1 Packaging Java Applications

For non-Java EE Java applications, it is common to package the application in a single JAR file, as [Example 10–1](#) shows.

**Example 10–1** *Packaging a non-Java EE Java Application*

```
domain_module.jar
  Java classes that represent the objects mapped
  project.xml
  session.xml
```

```
META-INF
  Manifest.mf
```

This JAR contains the TopLink files and domain objects required by the application, including the following:

- `sessions.xml` (see [Section 9.1.2, "sessions.xml File"](#));
- `project.xml` (see [Section 9.1.1, "project.xml File"](#)) (or the compiled `Project` class file if you are not using XML files for deployment);
- The mapped classes required by the application, in a fully-resolved directory structure.

When you create the JAR file, the JAR building utility automatically creates a directory structure within the JAR. Ensure that the `sessions.xml` file and the `project.xml` file (or `project` class file) appear at the root of the JAR file. Ensure that the class directory structure starts at the root of the JAR.

If you do not store the `project.xml` or `sessions.xml` files at the root of the JAR file, see [Section 10.8, "Packaging with TopLink Metadata File Resource Paths"](#).

## 10.2 Packaging JavaServer Pages and Servlet Applications

For simple Java EE applications without EJB, it is common to package the application in an Enterprise Archive (EAR) file made up of various Java EE application component archives, as [Example 10–2](#) shows.

### **Example 10–2 Packaging a Java EE JSP or Servlet Application Without EJB**

```
appname.ear
  META-INF
    application.xml
    orion-application.xml
  domain_module.jar
    Java classes that represent the object mapped
    project.xml
    session.xml
  META-INF
    Manifest.mf
  web_module.war
    html pages, JSP's, etc.
  META-INF
    web.xml
    orion-web.xml
  classes
    servlet classes
  lib
  client_module.jar
    Client classes
  META-INF
    application-client.xml
    orion-application-client.xml
```

The component archives with TopLink dependencies include TopLink domain JAR (see [Section 10.2.1, "How to Create the TopLink Domain JAR"](#)).

## 10.2.1 How to Create the TopLink Domain JAR

The domain JAR contains the TopLink files and domain objects required by the application, including the following:

- `sessions.xml` (see [Section 9.1.2, "sessions.xml File"](#));
- `project.xml` (see [Section 9.1.1, "project.xml File"](#)) (or the compiled `Project` class file, if you are not using XML files for deployment);
- The mapped classes required by the application, in a fully resolved directory structure.

When you create the JAR file, the JAR building utility automatically creates a directory structure within the JAR. Ensure that the `sessions.xml` file and the `project.xml` file (or `project.class` file) appear at the root of the JAR file. Also ensure that the class directory structure starts at the root of the JAR.

If you do not store the `project.xml` or `sessions.xml` files at the root of the JAR file, see [Section 10.8, "Packaging with TopLink Metadata File Resource Paths"](#).

## 10.3 Packaging Session Bean Applications

The packaging strategy you choose depends on whether you are packaging an EJB 1.*n* or 2.*n* session bean application (see [Section 10.3.1, "How to Package an EJB 1.n and 2.n Session Bean Application"](#)), or an EJB 3.0 session bean application (see [Section 10.3.2, "How to Package an EJB 3.0 Session Bean Application"](#)).

### 10.3.1 How to Package an EJB 1.*n* and 2.*n* Session Bean Application

For Java EE applications with session beans, it is common to package the application in an Enterprise Archive (EAR) file made up of various Java EE application component archives, as [Example 10-3](#) shows.

#### **Example 10-3 Packaging a Java EE Application with Session Beans**

```

appname.ear
  META-INF
    application.xml
    orion-application.xml
  ejb_module_X.jar
    EJB classes - session and non-entity beans
    META-INF
      ejb-jar.xml
      orion-ejb-jar.xml - no persistence-manager subentries
  domain_module.jar
    Java classes that represent the object mapped
    project.xml
    session.xml
    META-INF
      Manifest.mf
  web_module.war
    html pages, JSP's, etc.
    META-INF
      web.xml
      orion-web.xml
    classes
      servlet classes
    lib
  client_module.jar

```

```
Client classes
META-INF
  application-client.xml
  orion-application-client.xml
```

The component archives with TopLink dependencies include the following:

- TopLink domain JAR (see [Section 10.2.1, "How to Create the TopLink Domain JAR"](#));
- EJB JAR (see [Section 10.3.4, "How to Create the EJB JAR"](#)).

### 10.3.2 How to Package an EJB 3.0 Session Bean Application

For information on how to package an EJB 3.0 session bean application, see "Packaging an EclipseLink JPA Application" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Packaging\\_and\\_Deploying\\_EclipseLink\\_JPA\\_Applications\\_%28ELUG%29#Packaging\\_an\\_EclipseLink\\_JPA\\_Application](http://wiki.eclipse.org/Packaging_and_Deploying_EclipseLink_JPA_Applications_%28ELUG%29#Packaging_an_EclipseLink_JPA_Application).

### 10.3.3 How to Create the TopLink Domain JAR

The domain JAR contains the TopLink files and domain objects required by the application, including the following:

- `sessions.xml` (see [Section 9.1.2, "sessions.xml File"](#));
- `project.xml` (see [Section 9.1.1, "project.xml File"](#)) (or the compiled `Project.class` file if you are not using XML files for deployment);
- The mapped classes required by the application, in a fully-resolved directory structure.

When you create the JAR file, the JAR building utility automatically creates a directory structure within the JAR. Ensure that the `sessions.xml` file and the `project.xml` file (or `project.class` file) appear at the root of the JAR file. Also ensure that the class directory structure starts at the root of the JAR.

If you do not store the `project.xml` or `sessions.xml` files at the root of the JAR file, see [Section 10.8, "Packaging with TopLink Metadata File Resource Paths"](#).

### 10.3.4 How to Create the EJB JAR

In this type of application, the EJB JAR contains session beans. Consequently, its `orion-ejb-jar.xml` file does not contain `persistence-manager` or `pm-properties` entries. These entries apply only to CMP applications.

## 10.4 Packaging JPA Applications

See "Packaging an EclipseLink JPA Application" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Packaging\\_and\\_Deploying\\_EclipseLink\\_JPA\\_Applications\\_%28ELUG%29#Packaging\\_an\\_EclipseLink\\_JPA\\_Application](http://wiki.eclipse.org/Packaging_and_Deploying_EclipseLink_JPA_Applications_%28ELUG%29#Packaging_an_EclipseLink_JPA_Application) for information on how to package your JPA application.

## 10.5 Packaging a POJO Application for Weaving

To package a POJO application for weaving, you create a JAR that contains a `sessions.xml` file and a `persistence.xml` file.

For more information on weaving, see [Section 2.10.4.1, "To Package a POJO Application for Weaving"](#).

## 10.6 Packaging CMP Applications

For Java EE applications that use CMP to persist entity beans, it is common to package the application in an Enterprise Archive (EAR) file made up of various Java EE application component archives, as [Example 10-4](#) shows.

### **Example 10-4 Packaging a Java EE Application with Entity Beans with Container-Managed Persistence**

```

appname.ear
  META-INF
    application.xml
    orion-application.xml
  cmp_ejb_module_1.jar
    EJB classes - cmp entity beans
    META-INF
      ejb-jar.xml
      orion-ejb-jar.xml - includes persistence-manager properties
      toplink-ejb-jar.xml
  ejb_module_X.jar
    EJB classes - non-entity beans
    META-INF
      ejb-jar.xml
      orion-ejb-jar.xml - no persistence-manager subentries
  web_module.war
    html pages, JSP's, etc.
    META-INF
      web.xml
      orion-web.xml
    classes
      servlet classes
    lib
  client_module.jar
    Client classes
    META-INF
      application-client.xml
      orion-application-client.xml

```

The component archives with TopLink dependencies include EJB JAR (see [Section 10.3.4, "How to Create the EJB JAR"](#)).

### 10.6.1 How to Create the EJB JAR

In this type of application, the EJB JAR file specifically service both non-entity and entity EJB. It includes the following:

- The home and remote, and all implementation code for all mapped beans in the application.
- All mapped non-EJB classes from the Oracle JDeveloper or TopLink Workbench project
- The home and remote, and all implementation code for any session beans included in the application.
- Helper classes that contain TopLink amendment methods, and any other classes the application requires.

For example, an instance of `oracle.toplink.ejb.cmp.DeploymentCustomization` (for more information, see [customization-class](#) in [Table 9-3](#) in [Section 9.9.1](#), "How to Configure persistence-manager Entries").

Store the following XML files in the EJB JAR `\meta-inf` directory:

- `ejb-jar.xml` (see [ejb-jar.xml File](#))
- `JAVA-EE-COMTAINER-ejb-jar.xml` (see [JAVA-EE-COMTAINER-\*ejb-jar.xml\* File](#))
- `toplink-ejb-jar.xml` (see [toplink-\*ejb-jar.xml\* File](#))

---

**Note:** If you do not use XML files for deployment, include your compiled `oracle.toplink.sessions.Project` file at the root of the EJB JAR (not in the `\meta-inf` directory).

---

You must persist all of the entity beans to the same data source. For a CMP application, TopLink does not support the session broker functionality (see [Section 87.7](#), "Session Broker and Client Sessions").

## 10.7 Packaging BMP Applications

For Java EE applications that use BMP to persist entity beans, it is common to package the application in an Enterprise Archive (EAR) file made up of various Java EE application component archives, as [Example 10-5](#) shows.

### **Example 10-5** *Packaging a Java EE Application with Entity Beans with Bean-Managed Persistence*

```

appname.ear
  META-INF
    application.xml
    orion-application.xml
  bmp_ejb_module_1.jar
    EJB classes - bmp entity beans
    META-INF
      ejb-jar.xml
      orion-ejb-jar.xml - includes persistence-manager properties
      toplink-ejb-jar.xml
  ejb_module_X.jar
    EJB classes - non-entity beans
    META-INF
      ejb-jar.xml
      orion-ejb-jar.xml - no persistence-manager subentries
  domain_module.jar
    Java classes that represent the object mapped
    project.xml
    session.xml
    META-INF
      Manifest.mf
  web_module.war
    html pages, JSP's, etc.
    META-INF
      web.xml
      orion-web.xml
    classes

```



```

    servlet classes
  lib
client_module.jar
  Client classes
  META-INF
    application-client.xml
    orion-application-client.xml

```

The component archives with TopLink dependencies include the following:

- TopLink domain JAR (see [Section 10.2.1, "How to Create the TopLink Domain JAR"](#))
- EJB JAR (see [Section 10.3.4, "How to Create the EJB JAR"](#))

### 10.7.1 How to Create the TopLink Domain JAR

The domain JAR contains the TopLink files and domain objects required by the application, including the following:

- `sessions.xml` (see [Section 9.1.2, "sessions.xml File"](#));
- `project.xml` (see [Section 9.1.1, "project.xml File"](#)) (or the compiled `Project.class` file if you are not using XML files for deployment);
- The mapped classes required by the application, in a fully resolved directory structure.

When you create the JAR file, the JAR building utility automatically creates a directory structure within the JAR. Ensure that the `sessions.xml` file and the `project.xml` file (or `project.class` file) appear at the root of the JAR file. Also ensure that the class directory structure starts at the root of the JAR.

If you do not store the `project.xml` or `sessions.xml` files at the root of the JAR file, see [Section 10.8, "Packaging with TopLink Metadata File Resource Paths"](#).

### 10.7.2 How to Create EJB JAR File

In this type of application, the EJB JAR file specifically services both session and entity beans. It includes the following:

- The home and remote, and all implementation code for all mapped beans in the application.
- All mapped non-EJB classes from the Oracle JDeveloper or TopLink Workbench project.
- The home and remote, and all implementation code for any session beans included in the application.
- Helper classes that contain TopLink amendment methods, and any other classes the application requires.

Store the following XML files in the EJB JAR `\meta-inf` directory:

- `ejb-jar.xml` (see [ejb-jar.xml File](#))
- `JAVA-EE-COMTAINER-ejb-jar.xml` (see [JAVA-EE-COMTAINER-`ejb-jar.xml` File](#))

Because the EJB JAR does not contain entity beans with container-managed persistence, its `orion-ejb-jar.xml` file must not contain `persistence-manager` or `pm-properties` entries.

For more information, see [Section 9.9.1, "How to Configure persistence-manager Entries"](#).

## 10.8 Packaging with TopLink Metadata File Resource Paths

If you do not store the `project.xml` or `sessions.xml` files at the root of the JAR file, then you must provide the full resource path to the files when accessing them. Ensure that you use "/" in resources paths, not "\". Using "\" will not work in Java.

For example, in the `jar` element, reference the `project.xml` and `sessions.xml` files as follows:

```
<jar>/myapp/ordersys/persist/sessions.xml  
<jar>/myapp/ordersys/persist/project.xml
```

In the `sessions.xml` file, reference the `project.xml` as follows:

```
myapp/ordersys/persist/project.xml
```

To acquire the session, use the following:

```
SessionManager.getManager().getSession(  
    new XMLSessionConfigLoader("myapp/ordersys/persist/sessions.xml"),  
    "OrdersysSession",  
    getClass().getClassLoader()  
);
```

For more information about acquiring sessions at run time, see [Section 90.3, "Acquiring a Session from the Session Manager"](#).

---

---

## Deploying a TopLink Application

This chapter includes deployment information on various types of TopLink applications.

This chapter includes the following sections:

- [Deploying Java Applications](#)
- [Deploying JavaServer Pages and Servlets](#)
- [Deploying Session Bean Applications](#)
- [Deploying JPA Applications](#)
- [Deploying CMP Applications](#)
- [Deploying BMP Applications](#)
- [Performing Hot Deployment of EJB](#)

For more information, see the following:

- [Chapter 8, "Integrating TopLink with an Application Server"](#)
- [Chapter 9, "Creating TopLink Files for Deployment"](#)
- [Chapter 10, "Packaging a TopLink Application"](#)

### 11.1 Deploying Java Applications

Build the JAR file (see [Section 10.1, "Packaging Java Applications"](#)) and place it on the classpath.

For more information on accessing TopLink from your client application, see [Chapter 90, "Acquiring and Using Sessions at Run Time"](#).

### 11.2 Deploying JavaServer Pages and Servlets

After you build the WAR and JAR files (see [Section 10.2, "Packaging JavaServer Pages and Servlet Applications"](#)), build them into an EAR file for deployment. To deploy the EAR to your JSP servlet server, copy the EAR to a commonly used directory. You may also need to use server-specific deployment tools. For more information, see the server documentation.

For more information on accessing TopLink from your client application, see [Section 90.3.2, "How to Load a Session from sessions.xml with an Alternative Class Loader"](#).

## 11.3 Deploying Session Bean Applications

After you build the WAR and JAR files (see [Section 10.3, "Packaging Session Bean Applications"](#)), build them into an EAR file for deployment. To deploy the EAR file to your Java EE server, copy the EAR to a commonly used directory. You may also need to use server-specific deployment tools. For more information, see the server documentation.

For more information on accessing TopLink from your client application, see [Section 90.3.2, "How to Load a Session from sessions.xml with an Alternative Class Loader"](#).

Optionally, you may also consider [Section 11.7, "Performing Hot Deployment of EJB"](#).

## 11.4 Deploying JPA Applications

After you packaged your JPA application, deploy it to an application server of your choice.

For more information, see "Deploying an EclipseLink JPA Application" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Packaging\\_and\\_Deploying\\_EclipseLink\\_JPA\\_Applications\\_%28ELUG%29#Deploying\\_an\\_EclipseLink\\_JPA\\_Application](http://wiki.eclipse.org/Packaging_and_Deploying_EclipseLink_JPA_Applications_%28ELUG%29#Deploying_an_EclipseLink_JPA_Application).

## 11.5 Deploying CMP Applications

After you build the WAR and JAR files (see [Section 10.6, "Packaging CMP Applications"](#)), build them into an EAR file for deployment. To deploy the EAR file to your Java EE server, copy the EAR to a commonly used directory. You may also need to use server-specific deployment tools. For more information, see the server documentation.

This section describes [How to Deploy a CMP Application to OC4J](#).

For additional information on server-specific configuration, see [Chapter 8, "Integrating TopLink with an Application Server"](#).

Optionally, you may also consider [Section 11.7, "Performing Hot Deployment of EJB"](#).

### 11.5.1 How to Deploy a CMP Application to OC4J

For more information, see *Oracle Fusion Middleware Administrator's Guide*.

When you deploy a CMP application to OC4J, the following happens:

- OC4J performs a partial EJB conformance check on the beans and their associated interfaces.
- OC4J builds the internal OC4J classes that manage security and transactions, as well as the RMI stubs and skeletons that enable client access to the beans.
- TopLink builds concrete bean subclasses and EJB finder method implementations.

## 11.6 Deploying BMP Applications

After you build the WAR and JAR files, build them into an EAR file for deployment. To deploy the EAR file to your Java EE server, copy the EAR to a commonly used directory. You may also need to use server-specific deployment tools. For more information, see the server documentation.

For additional information on server-specific configuration, see [Chapter 8, "Integrating TopLink with an Application Server"](#).

Optionally, you may also consider [Section 11.7, "Performing Hot Deployment of EJB"](#).

## 11.7 Performing Hot Deployment of EJB

Many Java EE containers support *hot deployment*, a feature that enables you to deploy EJB on a running server. Hot deployment allows you to do the following:

- Deploy newly developed EJB to a running production system.
- Remove (undeploy) deployed EJB from a running server.
- Modify (redeploy) the behavior of deployed EJB by updating the bean class definition.

The client receives deployment exceptions when attempting to access undeployed or re-deployed bean instances. The client application must catch and handle the exceptions.

How you configure hot deployment of EJB depends on the type of Java EE application you are deploying:

- CMP application (see [Section 11.7.1, "How to Perform Hot Deployment in a CMP Application"](#));
- POJO application (see [Section 11.7.2, "How to Perform Hot Deployment in a POJO Application"](#)).

For more information about hot deployment, see the Java EE container documentation.

### 11.7.1 How to Perform Hot Deployment in a CMP Application

When you take advantage of hot deployment in a CMP application, consider the following:

- You must deploy all related beans (all beans that share a common TopLink project) within the same EJB JAR file. Because TopLink views deployment on a project level, deploy all the project beans (rather than just a portion of them) to maintain consistency across the project.
- When you redeploy a bean, you automatically reset its TopLink project. This flushes all object caches and rolls back any active object transactions associated with the project.

### 11.7.2 How to Perform Hot Deployment in a POJO Application

When you take advantage of hot deployment in a POJO application, you must refresh the TopLink session using the `SessionManager` method `getSession` with the appropriate arguments (see [Section 90.3.6, "How to Refresh a Session when the Class Loader Changes"](#)).

If you do not use this `SessionManager` method, then your application is responsible for destroying or refreshing the session when a hot deployment (or hot redeployment) occurs.



# Part IV

---

## Optimization and Customization of a TopLink Application

This part describes how to optimize and customize a TopLink application. It contains the following chapters:

- [Chapter 12, "Optimizing the TopLink Application"](#)

This chapter contains information on the diverse set of features TopLink provides to optimize performance.

- [Chapter 13, "Customizing the TopLink Application"](#)

This chapter describes how to customize various aspects of TopLink, based on your application's specific needs.





---

---

## Optimizing the TopLink Application

TopLink provides a diverse set of features to measure and optimize application performance. You can enable or disable most features in the descriptors or session, making any resulting performance gains global.

This chapter includes the following sections:

- [Introduction to Optimization](#)
- [Identifying Sources of Application Performance Problems](#)
- [Measuring TopLink Performance with the TopLink Profiler](#)
- [Measuring TopLink Performance with the Oracle Dynamic Monitoring System \(DMS\)](#)
- [Identifying General Performance Optimization](#)
- [Optimizing for a Production Environment](#)
- [Optimizing Schema](#)
- [Optimizing Mappings and Descriptors](#)
- [Optimizing Sessions](#)
- [Optimizing Cache](#)
- [Optimizing Data Access](#)
- [Optimizing Queries](#)
- [Optimizing the Unit of Work](#)
- [Optimizing Using Weaving](#)
- [Optimizing the Application Server and Database Optimization](#)
- [Optimizing Storage and Retrieval of Binary Data in XML](#)

### 12.1 Introduction to Optimization

Performance considerations are present at every step of the development cycle. Although this implies an awareness of performance issues in your design and implementation, it does not mean that you should expect to achieve the best possible performance in your first pass.

For example, if optimization complicates the design, leave it until the final development phase. You should still plan for these optimizations from your first iteration, to make them easier to integrate later.

The most important concept associated with tuning your TopLink application is the idea of an iterative approach. The most effective way to tune your application is to do the following:

1. Measure application performance using the TopLink profiler (see [Section 12.3, "Measuring TopLink Performance with the TopLink Profiler"](#)) or the Oracle Dynamic Monitoring System (DMS) profiler (see [Section 12.4, "Measuring TopLink Performance with the Oracle Dynamic Monitoring System \(DMS\)"](#));
2. Modify application components (see [Section 12.2, "Identifying Sources of Application Performance Problems"](#));
3. Measure performance again.

To identify the changes that improve your application performance, modify only one or two components at a time. You should also tune your application in a nonproduction environment before you deploy the application.

## 12.2 Identifying Sources of Application Performance Problems

For various parts of a TopLink enabled application, this section describes the performance problems most commonly encountered and provides suggestions for improving performance. Areas of the application where performance problems could occur include the following:

- General (see [Section 12.5, "Identifying General Performance Optimization"](#))
- Schema (see [Section 12.7, "Optimizing Schema"](#))
- Mappings and descriptors (see [Section 12.8, "Optimizing Mappings and Descriptors"](#))
- Sessions (see [Section 12.9, "Optimizing Sessions"](#))
- Cache (see [Section 12.10, "Optimizing Cache"](#))
- Data access (see [Section 12.11, "Optimizing Data Access"](#))
- Queries (see [Section 12.12, "Optimizing Queries"](#))
- Unit of work (see [Section 12.13, "Optimizing the Unit of Work"](#))
- Application server and database (see [Section 12.15, "Optimizing the Application Server and Database Optimization"](#))

## 12.3 Measuring TopLink Performance with the TopLink Profiler

The most important challenge to performance tuning is knowing what to optimize. To improve the performance of your application, identify the areas of your application that do not operate at peak efficiency. The TopLink performance profiler helps you identify performance problems by logging performance statistics for every executed query in a given session.

---

---

**Note:** You should also consider using general performance profilers, such as Oracle JDeveloper or JProbe, to analyze performance problems. These tools can provide more detail that may be required to properly diagnose a problem.

---

---

The TopLink performance profiler logs the following information to the TopLink log file (for general information about TopLink logging, see [Section 87.2.6, "Logging"](#)):

- query class;
- domain class;
- total time, total execution time of the query (in milliseconds);
- local time, the amount of time spent on the user's workstation (in milliseconds);
- number of objects, the total number of objects affected;
- number of objects handled per second;
- logging, the amount of time spent printing logging messages (in milliseconds);
- SQL prepare, the amount of time spent preparing the SQL script (in milliseconds);
- SQL execute, the amount of time spent executing the SQL script (in milliseconds);
- row fetch, the amount of time spent fetching rows from the database (in milliseconds);
- cache, the amount of time spent searching or updating the object cache (in milliseconds);
- object build, the amount of time spent building the domain object (in milliseconds);
- query prepare, the amount of time spent to prepare the query prior to execution (in milliseconds);
- SQL generation, the amount of time spent to generate the SQL script before it is sent to the database (in milliseconds).

---

**Note:** Use the TopLink performance profiler to profile single-threaded finite use cases to determine the bottle neck.

Do not use the TopLink performance profiler to enable monitoring of a long-running multi-threaded server—use the DMS profiler instead (see [Section 12.4, "Measuring TopLink Performance with the Oracle Dynamic Monitoring System \(DMS\)"](#)).

---

This section includes information on the following topics:

- [How to Configure the TopLink Performance Profiler](#)
- [How to Access the TopLink Profiler Results](#)

### 12.3.1 How to Configure the TopLink Performance Profiler

To enable the TopLink performance profiler, select the **TopLink** profiler option when configuring your session (see [Section 89.6, "Configuring a Performance Profiler"](#)).

The TopLink performance profiler is an instance of `oracle.toplink.tools.profiler.PerformanceProfiler` class. It provides the following public API:

- `logProfile`—enables the profiler;
- `dontLogProfile`—disables the profiler;
- `logProfileSummary`—organizes the profiler log into a summary of all the individual operation profiles including operation statistics like the shortest time of all the operations that were profiled, the total time of all the operations, the

number of objects returned by profiled queries, and the total time that was spent in each kind of operation that was profiled;

- `logProfileSummaryByQuery`—organizes the profiler log into a summary of all the individual operation profiles by query;
- `logProfileSummaryByClass`—organizes the profiler log into a summary of all the individual operation profiles by class.

### 12.3.2 How to Access the TopLink Profiler Results

The simplest way to view TopLink profiler results is to read the TopLink log files with a text editor. For general information about TopLink logging, such as logging file location, see [Section 87.2.6, "Logging"](#).

Alternatively, you can use the graphical performance profiler that the TopLink Web client provides. For more information, refer to the Web client online Help and README files.

[Example 12-1](#) shows an example of the TopLink profiler output.

#### **Example 12-1 Performance Profiler Output**

```
Begin Profile of{
ReadAllQuery(oracle.toplink.demos.employee.domain.Employee)
Profile(ReadAllQuery,# of obj=12, time=1399,sql execute=217,
prepare=495, row fetch=390, time/obj=116,obj/sec=8)
} End Profile
```

The second line of the profile contains the following information about a query:

- `ReadAllQuery(oracle.toplink.demos.employee.domain.Employee)`: specific query profiled, and its arguments.
- `Profile(ReadAllQuery)`: start of the profile and the type of query.
- `# of obj=12`: number of objects involved in the query.
- `time=1399`: total execution time of the query (in milliseconds).
- `sql execute=217`: total time spent executing the SQL statement.
- `prepare=495`: total time spent preparing the SQL statement.
- `row fetch=390`: total time spent fetching rows from the database.
- `time/obj=116`: number of milliseconds spent on each object.
- `obj/sec=8`: number of objects handled per second.

## 12.4 Measuring TopLink Performance with the Oracle Dynamic Monitoring System (DMS)

Oracle DMS is a library that enables application and system developers to use a variety of DMS sensors to measure and export customized performance metrics for specific software components (called nouns).

TopLink includes DMS instrumentation in essential objects to provide efficient monitoring of run-time data in TopLink-enabled applications, including both Java EE and non-Java EE applications.

DMS instrumentation plays an important role in Oracle Application Server Manageability and Diagnosability. TopLink-specific DMS instrumentation ensures that

you can use Manageability and Diagnosability to simplify management and problem diagnosis for TopLink-enabled applications deployed to Oracle Application Server.

By enabling DMS profiling in a TopLink application (see [Section 12.4.1, "How to Configure the Oracle DMS Profiler"](#)), you can collect and easily access run-time data that can help you with application administration tasks and performance tuning.

---

**Note:** You should also consider using general performance profilers, such as Oracle JDeveloper or JProbe, to analyze performance problems. These tools can provide more detail that may be required to properly diagnose a problem.

---

[Table 12-1](#) lists the many performance and status metrics TopLink provides through DMS for TopLink server session name *SERVER-SESSION-NAME*. TopLink also provides query metrics organized by domain class, query type, query name (if defined), and operation (if defined).

[Table 12-2](#) lists the miscellaneous TopLink operations for which DMS metrics are collected. These operations apply to any TopLink metric in [Table 12-1](#) with an *OPERATION-NAME* in its sensor name.

[Table 12-3](#) lists the various profiling levels you can use to adjust the level of profiling to the amount of monitoring information you require. Levels are listed in order of increasing system overhead.

You can easily access DMS data at run time using a management application that supports the Java Management Extensions (JMX) API such as the JConsole, or using any Web browser and the DMS Spy servlet.

For more information, see the following:

- [Section 12.4.2, "How to Access Oracle DMS Profiler Data Using JMX"](#)
- [Section 12.4.3, "How to Access Oracle DMS Profiler Data Using the DMS Spy Servlet"](#)
- *Oracle Fusion Middleware Administrator's Guide*

**Table 12-1 TopLink DMS Metrics**

DMS Noun Name	Sensor Name	Level <sup>1</sup>	Description
Cache_ <i>SERVER-SESSION-NAME</i>	CacheHits	HEAVY	The number of times that the object was found in the cache.
	CacheMisses	HEAVY	The number of times that the object was not found in the cache.
	Caching	ALL	Includes time spent adding, looking up, and removing objects in the cache.
Connection_ <i>SERVER-SESSION-NAME</i>	ConnectCalls	HEAVY	Total number of connect calls made.
	ConnectionManagement	ALL	Time spent managing connections including connecting, reconnecting, and disconnecting from a data source.
	ConnectionsInUse (default)	HEAVY	Number of connections in use per pool for any exclusive ConnectionPool(Write, ExclusiveRead).

**Table 12–1 (Cont.) TopLink DMS Metrics**

DMS Noun Name	Sensor Name	Level <sup>1</sup>	Description
Miscellaneous_ <i>SERVER-SESSION-NAME</i>	ConnectionsInUse ( <i>POOL-NAME</i> )	HEAVY	Number of connections in use per pool for any exclusive ConnectionPool(Write, ExclusiveRead).
	DisconnectCalls	HEAVY	Total number of disconnect calls made.
	DescriptorEvents	ALL	Time spent on execute DescriptorEvent.
	logging	ALL	Time spent logging TopLink activities.
	<i>OPERATION-NAME</i>	HEAVY	Total time spent on operation: <i>OPERATION-NAME</i> . This is for special operations not included in any query nouns, such as batch writing.  For valid <i>OPERATION-NAME</i> values, see <a href="#">Section 12–2, "TopLink Operations"</a> .
	SessionEvents	ALL	Time spent on execute SessionEvent.
	wrapping	ALL	Time spent wrapping both CMP and BMP beans.
RCM_ <i>SERVER-SESSION-NAME</i>	ChangesNotProcessed	ALL	The number of ObjectChangeSet thrown away because the object was not found in the cache.
	ChangesProcessed	ALL	The number of ObjectChangeSet was found in the cache.
	MessagesReceived	HEAVY	Number of messages that been received through the RCM.
	MessagesSent	HEAVY	Number of messages that been sent through the RCM.
	RCMStatus	HEAVY	One of [not configured, started, stopped].
Session_ <i>SERVER-SESSION-NAME</i>	RemoteChangeSets	HEAVY	Number of change sets received from remote machines and processed.
	ClientSession	HEAVY	The number of ClientSession was logged in.
	loginTime	NORMAL	Time the session was logged in.
	UnitOfWork	HEAVY	Count of total number of UnitOfWork objects created.

**Table 12–1 (Cont.) TopLink DMS Metrics**

DMS Noun Name	Sensor Name	Level <sup>1</sup>	Description
TopLink_SERVER-SESSION-NAME_ DOMAIN-CLASS_QUERY-TYPE_ QUERY-NAME	TopLink_SERVER-SESSION-NAME_ DOMAIN-CLASS_QUERY-TYPE_ QUERY-NAME_OPERATION-NAME	HEAVY	<p>Total time spent on operation: TopLink_SERVER-SESSION-NAME_DOMAIN-CLASS_QUERY-TYPE_QUERY-NAME_OPERATION-NAME.</p> <p>Where QUERY-NAME and OPERATION-NAME are optional.</p> <p>For example, given:</p> <pre>import com.acme.model.Employee; UpdateAllQuery updateAllQuery = new UpdateAllQuery(Employee.class); updateAllQuery.setName("UpdateAllEmployee");</pre> <p>The DMS sensor name would be:</p> <pre>TopLink_ SERVER-SESSION-NAME_ com.acme.model.Employee_ _UpdateAllQuery_ UpdateAllEmployee_ OPERATION-NAME</pre> <p>For valid OPERATION-NAME values, see <a href="#">Section 12–2</a>, "TopLink Operations".</p>
Transaction_ SERVER-SESSION-NAME	deleted_object	ALL	Object need to be removed from identity map from ObjectChangeSet.
	DistributedMerge	ALL	Time spent merging remote transaction changes into local shared cache. Appears when cache sync is used.
	MergeTime	ALL	Time spent merging changes into the shared cache.
	OptimisticLocks	HEAVY	Number of optimistic lock exceptions which were thrown.
	Sequencing	ALL	Includes time spent maintaining the sequence number mechanism and actually setting the sequence number on objects.
	TXAfterCompletion	ALL	Time spent on JTS afterCompletion(Object status) method.
	TXBeforeCompletion	ALL	Time spent on JTS beforeCompletion() method.

**Table 12–1 (Cont.) TopLink DMS Metrics**

DMS Noun Name	Sensor Name	Level <sup>1</sup>	Description
	UnitOfWorkCommits	HEAVY	Measures the commit process of the UnitOfWork.
	UnitOfWorkRegister	ALL	Includes time spent in registerExistingObject, registerNewContainerBean, registerNewContainerBeanForCMP, registerNewObject, registerObject, and readIntoWorkingCopy.
	UnitOfWorkRollbacks	HEAVY	Number of UnitOfWork commits that were rollback.

<sup>1</sup> See Table 12–3 for a description of each level setting.

**Table 12–2 TopLink Operations**

Operation	Description
DatabaseExecution	Time spent in calls to the JDBC statement. Includes time spent in calls to: close, executeUpdate, and executeQuery.
EISExecuteTime	Time spent building an InteractionSpec and building DatabaseRow objects from the execution on the data source. Specific to EIS.
ObjectBuilding	Time spent building persistent objects from database rows.
QueryPreparation	Time to prepare the query. Does not include SQL prepare.
RowFetch	Time taken to build DatabaseRow objects from the JDBC ResultSet. Includes regular SQL calls and stored procedure calls.
SqlGeneration	Time spent generating SQL. In the case of TopLink expressions, time spent converting expression to SQL.
SqlPrepare	Object relational: Time spent in JDBC preparing the statement. EIS: Time spent in EIS creating an interaction associated with a connection, and creating input and output record objects.

**Table 12–3 DMS Metric Collection Levels**

Level	Description
NONE	Disable collection of all DMS metrics.
NORMAL	Enable collection of TopLink DMS metrics. Adds very low overhead. This is the default setting.
HEAVY	Enable collection of basic TopLink DMS metrics. Adds about 1 percent overhead.
ALL	Enable all possible TopLink DMS metrics. Adds about 3 percent overhead.

### 12.4.1 How to Configure the Oracle DMS Profiler

To enable DMS metric collection for TopLink Java EE applications deployed to an application sever other than OC4J, or for TopLink Java SE applications, do the following:

1. Ensure that the `dms.jar` file is in your application classpath.  
By default, the `dms.jar` file is located in `<ORACLE_HOME>\jlib` directory.
2. Set system property `oracle.dms.sensors=<level>`, where `<level>` is one of the values listed in Table 12–3.
3. To enable the DMS profiler, select the **DMS profiler** option when configuring your TopLink session (see Section 89.6, "Configuring a Performance Profiler").



To enable DMS metric collection for EclipseLink JPA applications deployed to Oracle WebLogic Server, set the `eclipselink.profiler` persistence unit property to `DMSPerformanceProfiler` in the `persistence.xml` file, as follows:

```
<property name="eclipselink.profiler" value="DMSPerformanceProfiler"/>
```

For more information, see "How to Use the Persistence Unit Properties for Optimization" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#How\\_to\\_Use\\_the\\_Persistence\\_Unit\\_Properties\\_for\\_Optimization](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#How_to_Use_the_Persistence_Unit_Properties_for_Optimization).

You configure DMS support in your TopLink application differently depending on the type of application:

- [Configuring the Oracle DMS Profiler in a TopLink CMP Application on OC4J](#)
- [Configuring the Oracle DMS Profiler in a EclipseLink JPA Application on OC4J](#)

#### 12.4.1.1 Configuring the Oracle DMS Profiler in a TopLink CMP Application on OC4J

By default, DMS metric collection is enabled for TopLink CMP applications deployed to OC4J. For BMP or POJO applications deployed to OC4J, you must configure DMS metric collection (see [Section 89.6, "Configuring a Performance Profiler"](#)).

TopLink EJB deployed in OC4J are subject to the DMS configuration specified by the OC4J command line-property `-Doracle.dms.sensors=<level>` where `<level>` is one of the values listed in [Table 12-3](#).

#### 12.4.1.2 Configuring the Oracle DMS Profiler in a EclipseLink JPA Application on OC4J

To enable DMS metric collection for EclipseLink JPA applications deployed to OC4J, set the `eclipselink.profiler` persistence unit property to `DMSPerformanceProfiler` in the `persistence.xml` file, as follows:

```
<property name="eclipselink.profiler" value="DMSPerformanceProfiler"/>
```

For more information, see "How to Use the Persistence Unit Properties for Optimization" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#How\\_to\\_Use\\_the\\_Persistence\\_Unit\\_Properties\\_for\\_Optimization](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#How_to_Use_the_Persistence_Unit_Properties_for_Optimization).

## 12.4.2 How to Access Oracle DMS Profiler Data Using JMX

Using the Java Management Extensions (JMX) API, you can publish DMS profiler run-time data from a managed application (TopLink) to a JMX-compliant management application, by way of EJB-like MBean components.

When you configure your TopLink application to enable run-time services (see [Section 89.6, "Configuring a Performance Profiler"](#)) and you deploy your application to Oracle WebLogic Server, the TopLink runtime will deploy a JMX MBean so that a JMX management application can access the DMS profiler run-time data your application publishes.

For more information on using JMX on Oracle WebLogic Server, see [Section 8.3.3, "How to Integrate JMX"](#).

### 12.4.3 How to Access Oracle DMS Profiler Data Using the DMS Spy Servlet

Once your DMS-enabled TopLink application is running, you can access the DMS data it is collecting.

The DMS Spy servlet is available in all Java processes that use DMS. It lets you monitor metrics for a single Java process from a Web browser.

To access DMS data directly using the DMS Spy servlet, do the following:

1. Ensure that the `dms.jar` file is in your application classpath.  
By default, the `dms.jar` file is located in `<ORACLE_HOME>\jlib`.
2. Set the following system properties for the DMS enabled Java process you want to monitor:

```
oracle.dms.publisher.classes=oracle.dms.http.Httpd
oracle.dms.httpd.port.start=<port>
where <port> is the HTTP port on which DMS accepts requests (the default value
is 46080).
```

3. Apply the system property changes by restarting the Java process you want to monitor.
4. Using a Web browser, connect to the Java process and access the Spy servlet by entering the following URL:

```
http://<host>:<port>/dms0/Spy
where <host> is the host name of your Java process and <port> is the value
specified by the oracle.dms.httpd.port.start system property.
```

The Spy servlet displays all TopLink DMS-enabled objects appropriate for the current DMS level setting. Figure 12–1 shows an example of the DMS Spy servlet display.

Figure 12–1 DMS Spy Servlet Display

The screenshot shows the DMS Spy Servlet interface. On the left, there is a sidebar with a 'Metric Tables' section containing a list of links: JVM, TopLink Cache, TopLink Connections, TopLink Miscellaneous, TopLink Session, TopLink Transaction, Toplink Queries, and Toplink RCM. Below this list is the timestamp 'Thu Feb 19 17:02:02 EST 2004'. The main content area is titled 'TopLink\_Connections' and includes the URL 'xchen-pc:46080/dms0/Spy: TopLink:46080'. Below the title, there are links for 'TopLink\_Connections' and 'Metric Definitions'. The central part of the display is a table with the following data:

Name	Parent Host	Process	ConnectCalls, ops	ConnectionManagement	DisconnectCalls, ops	
Connection / (Test)	xchen-TopLink pc	TopLink: 46080		1 active, threads avg, msec completed, ops maxActive, threads maxTime, msec minTime, msec time, msec	0 1,453 1 1 1,453 1,453 1,453	0

At the bottom of the table, there are links for 'Top | Text | Metric Definitions' and a timestamp 'Thu Feb 19 17:02:07 EST 2004'.

## 12.5 Identifying General Performance Optimization

In general, avoid overriding TopLink default behavior unless your application requires it. Some TopLink defaults are suitable for a development environment; you should change these defaults to suit your production environment (see Section 12.6, "Optimizing for a Production Environment").

Use Oracle JDeveloper TopLink Editor or TopLink Workbench rather than manual coding. These tools are not only easy to use: the default configuration they export to deployment XML (and the code it generates, if required) represents best practices optimized for most applications.

## 12.6 Optimizing for a Production Environment

Some TopLink defaults are suitable for a development environment but Oracle recommends that you change these to suit your production environment for optimal performance. These defaults include:

- Batch writing: enable.  
For more information, see [Section 12.11.3, "How to Use Batch Writing for Optimization"](#).
- Statement caching: enable either in TopLink when using an internal connection pool or in the data source when using an external connection pool and choose a statement cache size appropriate for your application.  
For more information, see [Section 12.11.5, "How to Use Parameterized SQL \(Parameter Binding\) and Prepared Statement Caching for Optimization"](#).
- Read and write connection pool size: increase to the desired number of concurrent threads (for example, 50).  
For more information, see [Section 96.1.6, "Connection Pools"](#).
- Session cache size: increase to the desired number of objects to be cached in memory (for example, 1000). Note that you can configure session cache size for each class individually.  
For more information, see [Section 102.2.1.6, "Guidelines for Configuring the Cache and Identity Maps"](#).

## 12.7 Optimizing Schema

Optimization is an important consideration when you design your database schema and object model. Most performance issues occur when the object model or database schema is too complex, which can make the database slow and difficult to query. This is most likely to happen if you derive your database schema directly from a complex object model.

To optimize performance, design the object model and database schema together. However, allow each model to be designed optimally: do not require a direct one-to-one correlation between the two.

This section includes the following schema optimization examples:

- [Schema Case 1: Aggregation of Two Tables Into One](#)
- [Schema Case 2: Splitting One Table Into Many](#)
- [Schema Case 3: Collapsed Hierarchy](#)
- [Schema Case 4: Choosing One Out of Many](#)

### 12.7.1 Schema Case 1: Aggregation of Two Tables Into One

A common schema optimization technique is to aggregate two tables into a single table. This improves read and write performance by requiring only one database operation instead of two.

Table 12–4 and Table 12–5 illustrate the table aggregation technique.

**Table 12–4 Original Schema (Aggregation of Two Tables Case)**

Elements	Details
Title	ACME Member Location Tracking System
Classes	Member, Address
Tables	MEMBER, ADDRESS
Relationships	Source, Instance Variable, Mapping, Target, Member, address, one-to-one, Address

The nature of this application dictates that you always look up employees and addresses together. As a result, querying a member based on address information requires a database join, and reading a member and its address requires two read statements. Writing a member requires two write statements. This adds unnecessary complexity to the system, and results in poor performance.

A better solution is to combine the MEMBER and ADDRESS tables into a single table, and change the one-to-one relationship to an aggregate relationship. This lets you read all information with a single operation, and doubles the update and insert speed, because only a single row in one table requires modifications.

**Table 12–5 Optimized Schema (Aggregation of Two Tables Case)**

Elements	Details
Classes	Member, Address
Tables	MEMBER
Relationships	Source, Instance Variable, Mapping, Target, Member, address, aggregate, Address

### 12.7.2 Schema Case 2: Splitting One Table Into Many

To improve overall performance of the system, split large tables into two or more smaller tables. This significantly reduces the amount of data traffic required to query the database.

For example, the system illustrated in Table 12–6 assigns employees to projects within an organization. The most common operation reads a set of employees and projects, assigns employees to projects, and update the employees. The employee’s address or job classification is also occasionally used to determine the project on which the employee is placed.

**Table 12–6 Original Schema (Splitting One Table into Many Case)**

Elements	Details	Instance Variable	Mapping	Target
Title	ACME Employee Workflow System			

**Table 12–6 (Cont.) Original Schema (Splitting One Table into Many Case)**

Elements	Details	Instance Variable	Mapping	Target
Classes	Employee, Address, PhoneNumber, EmailAddress, JobClassification, Project			
Tables	EMPLOYEE, PROJECT, PROJ_EMP			
Relationships	Employee	address	aggregate	Address
	Employee	phoneNumber	aggregate	EmailAddress
	Employee	emailAddress	aggregate	EmailAddress
	Employee	job	aggregate	JobClassification
	Employee	projects	many-to-many	Project

When you read a large volume of employee records from the database, you must also read their aggregate parts. Because of this, the system suffers from general read performance issues. To resolve this, break the EMPLOYEE table into the EMPLOYEE, ADDRESS, PHONE, EMAIL, and JOB tables, as illustrated in [Table 12–7](#).

Because you usually read only the employee information, splitting the table reduces the amount of data transferred from the database to the client. This improves your read performance by reducing the amount of data traffic by 25 percent.

**Table 12–7 Optimized Schema (Splitting One Table into Many Case)**

Elements	Details	Instance Variable	Mapping	Target
Title	ACME Employee Workflow System			
Classes	Employee, Address, PhoneNumber, EmailAddress, JobClassification, Project			
Tables	EMPLOYEE, ADDRESS, PHONE, EMAIL, JOB, PROJECT, PROJ_EMP			
Relationships	Employee	address	one-to-one	Address
	Employee	phoneNumber	one-to-one	EmailAddress
	Employee	emailAddress	one-to-one	EmailAddress

**Table 12–7 (Cont.) Optimized Schema (Splitting One Table into Many Case)**

Elements	Details	Instance Variable	Mapping	Target
	Employee	job	one-to-one	JobClassification
	Employee	projects	many-to-many	Project

### 12.7.3 Schema Case 3: Collapsed Hierarchy

A common mistake when you transform an object-oriented design into a relational model, is to build a large hierarchy of tables on the database. This makes querying difficult, because queries against this type of design can require a large number of joins. It is usually a good idea to collapse some of the levels in your inheritance hierarchy into a single table.

Table 12–8 represents a system that assigns clients to a company’s sales representatives. The managers also track the sales representatives that report to them.

**Table 12–8 Original Schema (Collapsed Hierarchy Case)**

Elements	Details
Title	ACME Sales Force System
Classes	Tables
Person	PERSON
Employee	PERSON, EMPLOYEE
SalesRep	PERSON, EMPLOYEE, REP
Staff	PERSON, EMPLOYEE, STAFF
Client	PERSON, CLIENT
Contact	PERSON, CONTACT

The system suffers from complexity issues that hinder system development and performance. Nearly all queries against the database require large, resource-intensive joins. If you collapse the three-level table hierarchy into a single table, as illustrated in Table 12–9, you substantially reduce system complexity. You eliminate joins from the system, and simplify queries.

**Table 12–9 Optimized Schema (Collapsed Hierarchy Case)**

Elements	Details
Classes	Tables
Person	none
Employee	EMPLOYEE
SalesRep	EMPLOYEE
Staff	EMPLOYEE
Client	CLIENT
Contact	CLIENT

## 12.7.4 Schema Case 4: Choosing One Out of Many

In a one-to-many relationship, a single source object has a collection of other objects. In some cases, the source object frequently requires one particular object in the collection, but requires the other objects only infrequently. You can reduce the size of the returned result set in this type of case by adding an instance variable for the frequently required object. This lets you access the object without instantiating the other objects in the collection.

Table 12–10 represents a system by which an international shipping company tracks the location of packages in transit. When a package moves from one location to another, the system creates a new a location entry for the package in the database. The most common query against any given package is for its current location.

**Table 12–10 Original Schema (Choosing One out of Many Case)**

Elements	Details	Instance Variable	Mapping	Target
Title	ACME Shipping Package Location Tracking system			
Classes	Package, Location			
Tables	PACKAGE, LOCATION			
Relationships	Package	locations	one-to-many	Location

A package in this system can accumulate several location values in its LOCATION collection as it travels to its destination. Reading all locations from the database is resource intensive, especially when the only location of interest is the current location.

To resolve this type of problem, add a specific instance variable that represents the current location. You then add a one-to-one mapping for the instance variable, and use the instance variable to query for the current location. As illustrated in Table 12–10, because you can now query for the current location without reading all locations associated with the package, this dramatically improves the performance of the system.

**Table 12–11 Optimized Schema (Choosing One out of Many Case)**

Elements	Details	Instance Variable	Mapping	Target
Classes	Package, Location			
Tables	PACKAGE, LOCATION			
Relationships	Package	locations	one-to-many	Location
	Package	currentLocation	one-to-many	Location

## 12.8 Optimizing Mappings and Descriptors

Always use indirection (lazy loading). It is not only critical in optimizing database access, but also allows TopLink to make several other optimizations including optimizing its cache access and unit of work processing. See [Section 121.3, "Configuring Indirection \(Lazy Loading\)"](#).

Avoid using the existence checking option `checkCacheThenDatabase` on descriptors (see [Section 119.17, "Configuring Cache Existence Checking at the Descriptor Level"](#)), unless required by the application. The default existence checking behavior offers better performance.

Avoid expensive initialization in the default constructor that TopLink uses to instantiate objects. Instead, use lazy initialization or use a TopLink instantiation policy (see [Section 119.28, "Configuring Instantiation Policy"](#)) to configure the descriptor to use a different constructor.

Avoid using method access in your TopLink mappings (see [Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"](#)), especially if you have expensive or potentially dangerous side-effect code in your get or set methods; use the default direct attribute access instead.

## 12.9 Optimizing Sessions

Use a Server session in a server environment, not a `DatabaseSession`.

Use the TopLink client session instead of remote session. A client session is appropriate for most multiuser Java EE application server environments.

Do not pool client sessions. Pooling sessions offers no performance gains.

Oracle recommends you increase the size of your session read and write connection pools to the desired number of concurrent threads (for example, 50). You configure this in TopLink when using an internal connection pool or in the data source when using an external connection pool.

For more information, see the following:

- [Section 12.6, "Optimizing for a Production Environment"](#)
- [Section 87.3, "Server and Client Sessions"](#)
- [Section 96.1.6, "Connection Pools"](#)

## 12.10 Optimizing Cache

Cache coordination (see [Section 102.3, "Cache Coordination"](#)) is one way to allow multiple, possibly distributed, instances of a session to broadcast object changes among each other so that each session's cache can be kept up-to-date.

However, cache coordination is best suited to applications with specific characteristics (see [Section 102.3.1, "When to Use Cache Coordination"](#)). Before implementing cache coordination, tune the TopLink cache for each class using alternatives such as object identity type (see [Section 119.12, "Configuring Cache Type and Size at the Descriptor Level"](#)), cache invalidation (see [Section 102.2.5, "Cache Invalidation"](#)), or cache isolation (see [Section 102.2.7, "Cache Isolation"](#)). Doing so lets you configure the optimal cache configuration for each type of class (see [Table 12-12](#)) and may eliminate the need for distributed cache coordination altogether.



**Table 12–12 Identity Map and Cache Configuration by Class Type**

Class Type	Identity Map Options	Cache Options
read-only	soft, hard, or full <sup>1</sup>	
read-mostly	soft or hard	cache invalidation or cache coordination
write-mostly	weak	cache invalidation

<sup>1</sup> If the number of instances is finite.

If you do use cache coordination, use JMS for cache coordination rather than RMI. JMS is more robust, easier to configure, and runs asynchronously. If you require synchronous cache coordination, use RMI.

You can configure a descriptor to control when the TopLink runtime will refresh the session cache when an instance of this object type is queried (see [Section 119.9, "Configuring Cache Refreshing"](#)). Oracle does not recommend the use of **Always Refresh** or **Disable Cache Hits**.

Using **Always Refresh** may result in refreshing the cache on queries when not required or desired. As an alternative, consider configuring cache refresh on a query by query basis (see [Section 108.16.5, "How to Refresh the Cache"](#)).

Using **Disable Cache Hits** instructs TopLink to bypass the cache for object read queries based on primary key. This results in a database round trip every time an object read query based on primary key is executed on this object type, negating the performance advantage of the cache. When used in conjunction with **Always Refresh**, this option ensures that all queries go to the database. This can have a significant impact on performance. These options should only be used in specialized circumstances.

## 12.11 Optimizing Data Access

Depending on the type of data source your application accesses, TopLink offers a variety of `LogIn` options that you can use to tune the performance of low level data reads and writes.

You can use several techniques to improve data access performance for your application. This section discusses some of the more common approaches, including the following:

- [How to Optimize JDBC Driver Properties](#)
- [How to Optimize Data Format](#)
- [How to Use Batch Writing for Optimization](#)
- [How to Use Outer-Join Reading with Inherited Subclasses](#)
- [How to Use Parameterized SQL \(Parameter Binding\) and Prepared Statement Caching for Optimization](#)

### 12.11.1 How to Optimize JDBC Driver Properties

Consider the default behavior of the JDBC driver you choose for your application. Some JDBC driver options can affect data access performance.

Some important JDBC driver properties can be configured directly using Oracle JDeveloper TopLink Editor, TopLink Workbench, or TopLink API (for example, see [Section 12.12.6, "How to Use JDBC Fetch Size for Optimization"](#)).

JDBC driver properties that are not supported directly by Oracle JDeveloper TopLink Editor, TopLink Workbench, or TopLink API can still be configured as generic JDBC properties that TopLink passes to the JDBC driver.

For example, some JDBC drivers, such as Sybase JConnect, perform a database round trip to test whether or not a connection is closed: that is, calling the JDBC driver method `isClosed` results in a stored procedure call or SQL select. This database round-trip can cause a significant performance reduction. To avoid this, you can disable this behavior: for Sybase JConnect, you can set property name `CLOSED_TEST` to value `INTERNAL`.

For more information about configuring general JDBC driver properties from within your TopLink application, see [Section 97.5, "Configuring Properties"](#).

## 12.11.2 How to Optimize Data Format

By default, TopLink optimizes data access by accessing the data from JDBC in the format the application requires. For example, TopLink retrieves `long` data types from JDBC instead of having the driver return a `BigDecimal` that TopLink would then have to convert into a `long`.

Some older JDBC drivers do not perform data conversion correctly and conflict with this optimization. In this case, you can disable this optimization (see [Section 98.7, "Configuring Advanced Options"](#)).

## 12.11.3 How to Use Batch Writing for Optimization

Batch writing can improve database performance by sending groups of `INSERT`, `UPDATE`, and `DELETE` statements to the database in a single transaction, rather than individually.

When used without parameterized SQL, this is known as dynamic batch writing.

When used with parameterized SQL (see [Section 12.11.5, "How to Use Parameterized SQL \(Parameter Binding\) and Prepared Statement Caching for Optimization"](#)), this is known as parameterized batch writing. This allows a repeatedly executed statement, such as a group of inserts of the same type, to be executed as a single statement and a set of bind parameters. This can provide a large performance benefit as the database does not have to parse the batch.

When using batch writing, you can tune the maximum batch writing size.

In JPA applications, you can use persistence unit property `eclipselink.jdbc.batch-writing` (see "EclipseLink JPA Persistence Unit Properties for JDBC Connection Communication" table of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#How\\_to\\_Use\\_EclipseLink\\_JPA\\_Extensions\\_for\\_JDBC\\_Connection\\_Communication](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#How_to_Use_EclipseLink_JPA_Extensions_for_JDBC_Connection_Communication)).

In CMP or POJO applications, you can use `setMaxBatchWritingSize` method of the `Login` interface. The meaning of this value depends on whether or not you are using parameterized SQL:

- If you are using parameterized SQL (you configure your `Login` by calling its `bindAllParameters` method), the maximum batch writing size is the number of statements to batch with 100 being the default.
- If you are using dynamic SQL, the maximum batch writing size is the size of the SQL string buffer in characters with 32000 being the default.

By default, TopLink does not enable batch writing because not all databases and JDBC drivers support it. Oracle recommends that you enable batch writing for selected databases and JDBC drivers that support this option. If your JDBC driver does not support batch writing, use the batch writing capabilities of TopLink, known as TopLink batch writing (see [Section 98.6, "Configuring JDBC Options"](#)).

For a more detailed example of using batch writing to optimize write queries, see [Section 12.12.10.1.2, "Batch Writing and Parameterized SQL"](#).

#### 12.11.4 How to Use Outer-Join Reading with Inherited Subclasses

You can configure an object-level read query to allow inherited subclasses to be outer-joined to avoid the cost of a single query per class, as [Example 12–2](#) shows.

##### **Example 12–2 Configuring an ObjectLevelReadQuery to Outer-Join Inherited Subclasses**

```
objectLevelReadQuery.setShouldOuterJoinSubclasses(true);
```

You can configure a descriptor's `InheritancePolicy` to allow the same thing as [Example 12–3](#) shows. By configuring the `InheritancePolicy`, this option applies to all queries on the descriptor's class.

##### **Example 12–3 Configuring a Descriptor to Allow Inherited Subclasses to be Outer-Joined**

```
myDescriptor.getInheritancePolicy().setShouldOuterJoinSubclasses(true);
```

For more information, see the following:

- [Section 16.3, "Descriptors and Inheritance"](#)
- [Section 119.19, "Configuring Reading Subclasses on Queries"](#)
- [Section 108.7.1.5, "Join Reading and Object-Level Read Queries"](#)

#### 12.11.5 How to Use Parameterized SQL (Parameter Binding) and Prepared Statement Caching for Optimization

Using parameterized SQL, you can keep the overall length of an SQL query from exceeding the statement length limit that your JDBC driver or database server imposes.

Using parameterized SQL and prepared statement caching, you can improve performance by reducing the number of times the database SQL engine parses and prepares SQL for a frequently called query.

By default, TopLink enables parameterized SQL but not prepared statement caching. Oracle recommends that you enable statement caching either in TopLink when using an internal connection pool or in the data source when using an external connection pool and choose a statement cache size appropriate for your application.

---

**Note:** When parameter binding is enabled, querying a database field with a fixed CHAR length may result in no results returned. This is because the white space may not be trimmed. Instead, you can:

1. Use a Variable length column type (for example, VARCHAR).
  2. Force the proper padding manually (either in your application or in a converter).
  3. Not use parameter binding.
-

Not all JDBC drivers support all JDBC binding options (see [Section 98.6, "Configuring JDBC Options"](#)). Selecting a combination of options may result in different behavior from one driver to another. Before selecting JDBC options, consult your JDBC driver documentation. When choosing binding options, consider the following approach:

1. Try binding all parameters with all other binding options disabled.
2. If this fails to bind some large parameters, consider enabling one of the following options, depending on the parameter's data type and the binding options that your JDBC driver supports:
  - a. To bind large `String` parameters, try enabling string binding.  
If large `String` parameters still fail to bind, consider adjusting the maximum `String` size. `TopLink` sets the maximum `String` size to 32000 characters by default.
  - b. To bind large `Byte` array parameters, try enabling byte array binding.
3. If this fails to bind some large parameters, try enabling streams for binding.  
Typically, configuring string or byte array binding will invoke streams for binding. If not, explicitly configuring streams for binding may help.

For Java EE applications that use `TopLink` external connection pools, you must configure parameterized SQL in `TopLink`, but you cannot configure prepared statement caching in `TopLink`. In this case, you must configure prepared statement caching in the application server connection pool. For example, in `OC4J`, if you configure your `data-source.xml` file with a managed `data-source` (where `connection-driver` is `oracle.jdbc.OracleDriver`, and `class` is `oracle.j2ee.sql.DriverManagerDataSource`), you can configure a non-zero `num-cached-statements` that enables JDBC statement caching and defines the maximum number of statements cached.

For applications that use `TopLink` internal connection pools, you can configure parameterized SQL and prepared statement caching.

You can configure parameterized SQL and prepared statement caching at the following levels:

- session database login level—applies to all queries and provides additional parameter binding API to alleviate the limit imposed by some drivers on SQL statement size.

Oracle recommends that you use this approach.

For more information, see the following:

- JPA applications: see persistence unit properties `eclipselink.jdbc.bind-parameters` and `eclipselink.jdbc.cache-statements.size` in "EclipseLink JPA Persistence Unit Properties for JDBC Connection Communication" table of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#How\\_to\\_Use\\_EclipseLink\\_JPA\\_Extensions\\_for\\_JDBC\\_Connection\\_Communication](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#How_to_Use_EclipseLink_JPA_Extensions_for_JDBC_Connection_Communication).
- CMP and POJO applications: see [Section 98.6, "Configuring JDBC Options"](#)
- project level—applies to all named queries (see [Section 20.7, "Configuring Named Query Parameterized SQL and Statement Caching at the Project Level"](#));

- descriptor level—applies on a per-named-query basis (see [Section 119.7.1.9, "Configuring Named Query Options"](#));
- query level—applies on a per-query basis (see [Section 109.2.9, "How to Use Parameterized SQL and Statement Caching in a DatabaseQuery"](#)).

## 12.12 Optimizing Queries

TopLink provides an extensive query API for reading, writing, and updating data. This section describes ways of optimizing query performance in various circumstances.

Before optimizing queries, consider the optimization suggestions in [Section 12.11, "Optimizing Data Access"](#).

This section includes information on the following:

- [How to Use Parameterized SQL and Prepared Statement Caching for Optimization](#)
- [How to Use Named Queries for Optimization](#)
- [How to Use Batch and Join Reading for Optimization](#)
- [How to Use Partial Object Queries and Fetch Groups for Optimization](#)
- [How to Use Read-Only Queries for Optimization](#)
- [How to Use JDBC Fetch Size for Optimization](#)
- [How to Use Cursored Streams and Scrollable Cursors for Optimization](#)
- [How to Use Result Set Pagination for Optimization](#)
- [Read Optimization Examples](#)
- [Write Optimization Examples](#)

### 12.12.1 How to Use Parameterized SQL and Prepared Statement Caching for Optimization

These features let you cache and reuse a query's prepared database statement when the query is reexecuted.

For more information, see [Section 12.11.5, "How to Use Parameterized SQL \(Parameter Binding\) and Prepared Statement Caching for Optimization"](#).

### 12.12.2 How to Use Named Queries for Optimization

Whenever possible, use named queries in your application. Named queries help you avoid duplication, are easy to maintain and reuse, and easily add complex query behavior to the application. Using named queries also allows for the query to be prepared once, and for the SQL generation to be cached.

For more information, see [Section 108.8, "Named Queries"](#).

### 12.12.3 How to Use Batch and Join Reading for Optimization

To optimize database read operations, TopLink supports both batch and join reading. When you use these techniques, you dramatically decrease the number of times you access the database during a read operation, especially when your result set contains a large number of objects.

For more information, see the following:

- [Section 12.11.3, "How to Use Batch Writing for Optimization"](#)
- [Section 12.11.4, "How to Use Outer-Join Reading with Inherited Subclasses"](#)
- For JPA applications, see the following:
  - "How to Use the @JoinFetch Annotation" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#How\\_to\\_Use\\_the\\_.40JoinFetch\\_Annotation](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#How_to_Use_the_.40JoinFetch_Annotation)
  - Persistence unit property `eclipselink.jdbc.batch-writing` in "EclipseLink JPA Persistence Unit Properties for JDBC Connection Communication" table of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#How\\_to\\_Use\\_EclipseLink\\_JPA\\_Extensions\\_for\\_JDBC\\_Connection\\_Communication](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#How_to_Use_EclipseLink_JPA_Extensions_for_JDBC_Connection_Communication)
- For CMP and POJO applications, see the following:
  - [Section 108.7.1.5, "Join Reading and Object-Level Read Queries"](#)
  - [Section 109.2.1.9, "Using Batch Reading"](#)

## 12.12.4 How to Use Partial Object Queries and Fetch Groups for Optimization

Partial object queries let you retrieve partially populated objects from the database rather than complete objects.

When using weaving with JPA or POJO or when using CMP applications, you can use fetch groups to accomplish the same performance optimization.

For more information about partial object reading, see [Section 108.7.1.3, "Partial Object Queries"](#).

For more information about fetch groups, see [Section 16.2.4, "Fetch Groups"](#).

## 12.12.5 How to Use Read-Only Queries for Optimization

You can configure an object-level read query as read-only, as [Example 12–4](#) shows. When you execute such a query in the context of a `UnitOfWork` (or EclipseLink JPA persistence provider), TopLink returns a read-only, non-registered object. You can improve performance by querying read-only data in this way because the read-only objects need not be registered or checked for changes.

### **Example 12–4** *Configuring an ObjectLevelReadQuery as Read-Only*

```
objectLevelReadQuery.setIsReadOnly(true);
```

For more information, see the following:

- [Section 119.3, "Configuring Read-Only Descriptors"](#)
- [Section 108.7.1.4, "Read-Only Query"](#)

## 12.12.6 How to Use JDBC Fetch Size for Optimization

The JDBC fetch size gives the JDBC driver a hint as to the number of rows that should be fetched from the database when more rows are needed.

For large queries that return a large number of objects you can configure the row fetch size used in the query to improve performance by reducing the number database hits required to satisfy the selection criteria.

Most JDBC drivers default to a fetch size of 10, so if you are reading 1000 objects, increasing the fetch size to 256 can significantly reduce the time required to fetch the query's results. The optimal fetch size is not always obvious. Usually, a fetch size of one half or one quarter of the total expected result size is optimal. Note that if you are unsure of the result set size, incorrectly setting a fetch size too large or too small can decrease performance.

Set the query fetch size with `ReadQuery` method `setFetchSize`, as [Example 12-5](#) shows. Alternatively, you can use `ReadQuery` method `setMaxRows` to set the limit for the maximum number of rows that any `ResultSet` can contain.

#### **Example 12-5 JDBC Driver Fetch Size**

```
// Create query and set Employee as its reference class
ReadAllQuery query = new ReadAllQuery(Employee.class);
ExpressionBuilder builder = query.getExpressionBuilder();
query.setSelectionCriteria(builder.get("id").greaterThan(100));

// Set the JDBC fetch size
query.setFetchSize(50);

// Configure the query to return results as a ScrollableCursor
query.useScrollableCursor();

// Execute the query
ScrollableCursor cursor = (ScrollableCursor) session.executeQuery(query);

// Iterate over the results
while (cursor.hasNext()) {
    System.out.println(cursor.next().toString());
}
cursor.close();
```

In this example, when you execute the query, the JDBC driver retrieves the first 50 rows from the database (or all rows if less than 50 rows satisfy the selection criteria). As you iterate over the first 50 rows, each time you call `cursor.next()`, the JDBC driver returns a row from local memory—it does not need to retrieve the row from the database. When you try to access the fifty first row (assuming there are more than 50 rows that satisfy the selection criteria), the JDBC driver again goes to the database and retrieves another 50 rows. In this way, 100 rows are returned with only two database hits.

If you specify a value of zero (default; means the fetch size is not set), then the hint is ignored and the JDBC driver's default is used.

For more information about configuring JDBC driver properties from within your TopLink application, see [Section 97.5, "Configuring Properties"](#).

## **12.12.7 How to Use Cursored Streams and Scrollable Cursors for Optimization**

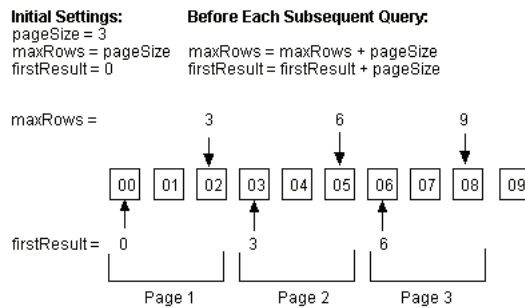
You can configure a query to retrieve data from the database using a cursored Java stream or scrollable cursor. This lets you view a result set in manageable increments rather than as a complete collection. This is useful when you have a large result set. You can further tune performance by configuring the JDBC driver fetch size used (see [Section 12.12.6, "How to Use JDBC Fetch Size for Optimization"](#)).

For more information about scrollable cursors, see [Section 111.11, "Handling Cursor and Stream Query Results"](#).

## 12.12.8 How to Use Result Set Pagination for Optimization

As [Figure 12–2](#) shows, using `ReadQuery` methods `setMaxRows(maxRows)` and `setFirstResult(firstResult)`, you can configure a query to retrieve a result set in pages, that is, a partial result as a `List` of `pageSize` (or less) results.

**Figure 12–2 Using Result Set Pagination**



In this example, for the first query invocation, `pageSize=3`, `maxRows=pageSize`, and `firstResult=0`. This returns a `List` of results 00 through 02.

For each subsequent query invocation, you increment `maxRows=maxRows+pageSize` and `firstResult=firstResult+pageSize`. This returns a new `List` for each page of results 03 through 05, 06 through 08, and so on.

Typically, you use this approach when you do not necessarily need to process the entire result set. For example, when a user wishes to scan the result set a page at a time looking for a particular result and may abandon the query after the desired record is found.

The advantage of this approach over cursors is that it does not require any state or live connection on the server; you only need to store the `firstResult` index on the client. This makes it useful for paging through a Web result.

For more information, see the following:

- [Section 111.12, "Handling Query Results Using Pagination"](#)
- [Section 12.12.7, "How to Use Cursored Streams and Scrollable Cursors for Optimization"](#)

## 12.12.9 Read Optimization Examples

TopLink provides the read optimization features listed in [Table 12–13](#).

This section includes the following read optimization examples:

- [Reading Case 1: Displaying Names in a List](#)
- [Reading Case 2: Batch Reading Objects](#)
- [Reading Case 3: Using Complex Custom SQL Queries](#)
- [Reading Case 4: Using View Objects](#)
- [Reading Case 5: Inheritance Subclass Outer-Joining](#)



**Table 12–13 Read Optimization Features**

Feature	Function	Performance Technique
Unit of work	Tracks object changes within the unit of work.	To minimize the amount of tracking required, registers only those objects that will change.  For more information, see <a href="#">Chapter 113, "Introduction to TopLink Transactions"</a> .
Indirection (lazy loading)	Uses indirection objects to defer the loading and processing of relationships.	Provides a major performance benefit. It allows database access to be optimized and allows TopLink to internally make several optimizations in caching and unit of work.
Soft cache, weak identity map	Offers client-side caching for objects read from database, and drops objects from the cache when memory becomes low.	Reduces database calls and improves memory performance.  For more information, see <a href="#">Section 102.2.1, "Cache Type and Object Identity"</a> .
Weak identity map	Offers client-side caching for objects.	Reduces database access and maintains a cache of all referenced objects.  For more information, see <a href="#">Section 102.2.1, "Cache Type and Object Identity"</a> .
Batch reading and joining	Reduces database access by batching many queries into a single query that reads more data.	Dramatically reduces the number of database accesses required to perform a read query.  For more information, see <a href="#">Section 109.2.1.9, "Using Batch Reading"</a> and <a href="#">Section 109.2.1.10, "Using Join Reading with ObjectLevelReadQuery"</a> .
Partial object reading and fetch groups.	Allows reading of a subset of a result set of the object's attributes.	Reduces the amount of data read from the database.  For more information, see <a href="#">Section 108.7.1.3, "Partial Object Queries"</a> .  For more information about fetch groups, see <a href="#">Section 16.2.4, "Fetch Groups"</a> .
Report query	Similar to partial object reading, but returns only the data instead of the objects.	Supports complex reporting functions such as aggregation and group-by functions. Also lets you compute complex results on the database, instead of reading the objects into the application and computing the results locally.  For more information, see <a href="#">Section 108.7.5, "Report Query"</a> .
Read-only query	TopLink returns a read-only, non-registered object.	The read-only objects need not be registered or checked for changes.  For more information, see <a href="#">Section 12.12.5, "How to Use Read-Only Queries for Optimization"</a>

**Table 12–13 (Cont.) Read Optimization Features**

Feature	Function	Performance Technique
JDBC fetch size and ReadQuery first result maximum rows	Reduces the number of database hits required to return all the rows that satisfy selection criteria.	For more information, see <a href="#">Section 12.12.6, "How to Use JDBC Fetch Size for Optimization"</a> .
Cursors	Lets you view a large result set in manageable increments rather than as a complete collection	For more information, see <a href="#">Section 12.12.7, "How to Use Cursored Streams and Scrollable Cursors for Optimization"</a>
Inheritance subclass outer joins	Allows queries against an inheritance superclass that can read all of its subclasses in a single query, instead of multiple queries, with or without a view.	For more information, see <a href="#">Section 12.12.9.5, "Reading Case 5: Inheritance Subclass Outer-Joining"</a> .
Soft identity map	Similar to the weak identity map, except that the map uses soft references instead of weak references. This method allows full garbage collection and provides full caching and guaranteed identity	Allows for optimal caching of the objects without the overhead of a sub-cache, while still allowing the JVM to garbage collect the objects if memory is low.  For more information, see <a href="#">Section 102.2.1.3, "Soft Identity Map"</a> .

**12.12.9.1 Reading Case 1: Displaying Names in a List**

An application may ask the user to choose an element from a list. Because the list displays only a subset of the information contained in the objects, it is not necessary to query for all information for objects from the database.

TopLink features that optimize these types of operations include the following:

- [Partial Object Reading](#)
- [Report Query](#)
- [Fetch Groups](#)

These features let you query only the information required to display the list. The user can then select an object from the list.

**12.12.9.1.1 Partial Object Reading** Partial object reading is a query designed to extract only the required information from a selected record in a database, rather than all the information the record contains. Because partial object reading does not fully populate objects, you can neither cache nor edit partially read objects.

For more information about partial object queries, see [Section 108.7.1.3, "Partial Object Queries"](#).

In [Example 12–6](#), the query builds complete employee objects, even though the list displays only employee last names. With no optimization, the query reads all the employee data.

**Example 12–6 No Optimization**

```
/* Read all the employees from the database, ask the user to choose one and return
it. This must read in all the information for all the employees */
List list;

// Fetch data from database and add to list box
```

```
Vector employees = (Vector) session.readAllObjects(Employee.class);
list.addAll(employees);
```

```
// Display list box
```

```
....
```

```
// Get selected employee from list
```

```
Employee selectedEmployee = (Employee) list.getSelectedItemAt();
```

```
return selectedEmployee;
```

[Example 12-7](#) demonstrates the use of partial object reading. It reads only the last name and primary key for the employee data. This reduces the amount of data read from the database.

### **Example 12-7 Optimization Through Partial Object Reading**

```
/* Read all the employees from the database, ask the user to choose one and return
it. This uses partial object reading to read just the last names of the employees.
Since TopLink automatically includes the primary key of the object, the full
object can easily be read for editing */
```

```
List list;
```

```
// Fetch data from database and add to list box
```

```
ReadAllQuery query = new ReadAllQuery(Employee.class);
```

```
query.addPartialAttribute("lastName");
```

```
// The next line avoids a query exception
```

```
query.dontMaintainCache();
```

```
Vector employees = (Vector) session.executeQuery(query);
```

```
list.addAll(employees);
```

```
// Display list box
```

```
....
```

```
// Get selected employee from list
```

```
Employee selectedEmployee = (Employee) session.readObject(list.getSelectedItemAt());
```

```
return selectedEmployee;
```

**12.12.9.1.2 Report Query** Report query lets you retrieve data from a set of objects and their related objects. Report query supports database reporting functions and features.

For more information, see [Section 108.5.2, "Report Query Results"](#).

[Example 12-8](#) demonstrates the use of report query to read only the last name of the employees. This reduces the amount of data read from the database compared to the code in [Example 12-6](#), and avoids instantiating employee instances.

### **Example 12-8 Optimization Through Report Query**

```
/* Read all the employees from the database, ask the user to choose one and return
it. The report query is used to read just the last name of the employees. Then the
primary key stored in the report query result to read the real object */
```

```
List list;
```

```
// Fetch data from database and add to list box
```

```
ExpressionBuilder builder = new ExpressionBuilder();
```

```
ReportQuery query = new ReportQuery (Employee.class, builder);
```

```
query.addAttribute("lastName");
```

```
query.retrievePrimaryKeys();
```

```
Vector reportRows = (Vector) session.executeQuery(query);
```

```
list.addAll(reportRows);
```

```
// Display list box
....

// Get selected employee from list
ReportQueryResult result = (ReportQueryResult) list.getSelectedItem();
Employee selectedEmployee =
    (Employee)result.readobject(Employee.Class,session);
```

Although the differences between the unoptimized example ([Example 12-6](#)) and the report query optimization in [Example 12-8](#) appear to be minor, report queries offer a substantial performance improvement.

**12.12.9.1.3 Fetch Groups** Fetch groups are similar to partial object reading, but does allow caching of the objects read. For objects with many attributes or reference attributes to complex graphs (or both), you can define a fetch group that determines what attributes are returned when an object is read. Because TopLink will automatically execute additional queries when the `get` method is called for attributes not in the fetch group, ensure that the unfetched data is not required: refetching data can become a performance issue.

For more information about querying with fetch groups, see [Section 111.3, "Using Queries with Fetch Groups"](#).

[Example 12-9](#) demonstrates the use of a static fetch group.

**Example 12-9 Configuring a Query with a FetchGroup Using the FetchGroupManager**

```
// Create static fetch group at the descriptor level
FetchGroup group = new FetchGroup("nameOnly");
group.addAttribute("firstName");
group.addAttribute("lastName");
descriptor.getFetchGroupManager().addFetchGroup(group);

// Use static fetch group at query level
ReadAllQuery query = new ReadAllQuery(Employee.class);
query.setFetchGroupName("nameOnly");

/* Only Employee attributes firstName and lastName are fetched.
   If you call the Employee get method for any other attribute, TopLink executes
   another query to retrieve all unfetched attribute values. Thereafter,
   calling that get method will return the value directly from the object */
```

### 12.12.9.2 Reading Case 2: Batch Reading Objects

The way your application reads data from the database affects performance. For example, reading a collection of rows from the database is significantly faster than reading each row individually.

A common performance challenge is to read a collection of objects that have a one-to-one reference to another object. This typically requires one read operation to read in the source rows, and one call for each target row in the one-to-one relationship.

To reduce the number of read operations required, use join and batch reading. [Example 12-10](#) illustrates the unoptimized code required to retrieve a collection of objects with a one-to-one reference to another object. [Example 12-11](#) and [Example 12-12](#) illustrate the use of joins and batch reading to improve efficiency.

**Example 12-10 No Optimization**

```
/* Read all the employees, and collect their address' cities. This takes N + 1
   queries if not optimized */
```

```

// Read all the employees from the database. This requires 1 SQL call
Vector employees = session.readAllObjects(Employee.class,
    new ExpressionBuilder().get("lastName").equal("Smith"));

//SQL: Select * from Employee where l_name = 'Smith'

// Iterate over employees and get their addresses.
// This requires N SQL calls
Enumeration enum = employees.elements();
Vector cities = new Vector();
while(enum.hasMoreElements()) {
    Employee employee = (Employee) enum.nextElement();
    cities.addElement(employee.getAddress().getCity());
}
//SQL: Select * from Address where address_id = 123, etc

```

### **Example 12–11 Optimization Through Joining**

```

/* Read all the employees; collect their address' cities. Although the code
is almost identical because joining optimization is used it takes only 1
query */

// Read all the employees from the database using joining.
// This requires 1 SQL call
ReadAllQuery query = new ReadAllQuery(Employee.class);
ExpressionBuilder builder = query.getExpressionBuilder();
query.setSelectionCriteria(builder.get("lastName").equal("Smith"));
query.addJoinedAttribute("address");
Vector employees = session.executeQuery(query);

/* SQL: Select E.*, A.* from Employee E, Address A where E.l_name = 'Smith' and
E.address_id = A.address_id Iterate over employees and get their addresses.
The previous SQL already read all the addresses, so no SQL is required */
Enumeration enum = employees.elements();
Vector cities = new Vector();
while (enum.hasMoreElements()) {
    Employee employee = (Employee) enum.nextElement();
    cities.addElement(employee.getAddress().getCity());
}

```

### **Example 12–12 Optimization Through Batch Reading**

```

/* Read all the employees; collect their address' cities. Although the code
is almost identical because batch reading optimization is used it takes only
2 queries */

// Read all the employees from the database, using batch reading.
// This requires 1 SQL call, note that only the employees are read
ReadAllQuery query = new ReadAllQuery(Employee.class);
ExpressionBuilder builder = query.getExpressionBuilder();
query.setSelectionCriteria(builder.get("lastName").equal("Smith"));
query.addBatchReadAttribute("address");
Vector employees = (Vector)session.executeQuery(query);

// SQL: Select * from Employee where l_name = 'Smith'

// Iterate over employees and get their addresses.
// The first address accessed will cause all the addresses
// to be read in a single SQL call
Enumeration enum = employees.elements();

```

```

Vector cities = new Vector();
while (enum.hasMoreElements()) {
    Employee employee = (Employee) enum.nextElement();
    cities.addElement(employee.getAddress().getCity());
    // SQL: Select distinct A.* from Employee E, Address A
    // where E.l_name = 'Smith' and E.address_id = A.address_i
}

```

Because the two-phase approach to the query ([Example 12-11](#) and [Example 12-12](#)) accesses the database only twice, it is significantly faster than the approach illustrated in [Example 12-10](#).

Joins offer a significant performance increase under most circumstances. Batch reading offers a further performance advantage in that it allows for delayed loading through value holders, and has much better performance where the target objects are shared.

For example, if employees in [Example 12-10](#), [Example 12-11](#), and [Example 12-12](#) are at the same address, batch reading reads much less data than joining, because batch reading uses a SQL DISTINCT call to filter duplicate data.

Batch reading and joining are available for one-to-one, one-to-many, many-to-many, direct collection, direct map and aggregate collection mappings. Note that one-to-many joining will return a large amount of duplicate data and so is normally less efficient than batch reading.

### 12.12.9.3 Reading Case 3: Using Complex Custom SQL Queries

TopLink provides a high-level query mechanism. However, if your application requires a complex query, a direct SQL or stored procedure call may be the best solution.

For more information about executing SQL calls, see [Section 108.9.1.1, "SQLCall"](#).

### 12.12.9.4 Reading Case 4: Using View Objects

Some application operations require information from several objects rather than from just one. This can be difficult to implement, and resource-intensive. [Example 12-13](#) illustrates unoptimized code that reads information from several objects.

#### **Example 12-13 No Optimization**

```

/* Gather the information to report on an employee and return the summary of the
information. In this situation, a hash table is used to hold the report
information. Notice that this reads a lot of objects from the database, but
uses very little of the information contained in the objects. This may take 5
queries and read in a large number of objects */

```

```

public Hashtable reportOnEmployee(String employeeName) {
    Vector projects, associations;
    Hashtable report = new Hashtable();
    // Retrieve employee from database
    Employee employee = session.readObject(Employee.class,
        new ExpressionBuilder.get("lastName").equal(employeeName));
    // Get all the projects affiliated with the employee
    projects = session.readAllObjects(Project.class,
        "SELECT P.* FROM PROJECT P," +
        "EMPLOYEE E WHERE P.MEMBER_ID = E.EMP_ID AND E.L_NAME = " +
        employeeName);
    // Get all the associations affiliated with the employee
    associations = session.readAllObjects(Association.class, "SELECT A.* " +
        "FROM ASSOC A, EMPLOYEE E WHERE A.MEMBER_ID = E.EMP_ID AND E.L_NAME = "
        + employeeName);
}

```

```

        report.put("firstName", employee.getFirstName());
        report.put("lastName", employee.getLastName());
        report.put("manager", employee.getManager());
        report.put("city", employee.getAddress().getCity());
        report.put("projects", projects);
        report.put("associations", associations);
        return report;
    }

```

To improve application performance in these situations, define a new read-only object to encapsulate this information, and map it to a view on the database. To set the object to be read-only, configure its descriptor as read-only (see [Section 119.3, "Configuring Read-Only Descriptors"](#)).

#### **Example 12–14 Optimization Through View Object**

```

CREATE VIEW NAMED EMPLOYEE_VIEW AS (SELECT F_NAME = E.F_NAME, L_NAME = E.L_
NAME, EMP_ID = E.EMP_ID, MANAGER_NAME = E.NAME, CITY = A.CITY, NAME = E.NAME
FROM EMPLOYEE E, EMPLOYEE M, ADDRESS A
WHERE E.MANAGER_ID = M.EMP_ID
AND E.ADDRESS_ID = A.ADDRESS_ID)

```

Define a descriptor for the `EmployeeReport` class as follows:

- Define the descriptor as usual, but specify the `tableName` as `EMPLOYEE_VIEW`.
- Map only the attributes required for the report. In the case of the `numberOfProjects` and `associations`, use a transformation mapping to retrieve the required data.

You can now query the report from the database in the same way as any other object enabled by TopLink.

#### **Example 12–15 View the Report from Example 12–14**

```

/* Return the report for the employee */
public EmployeeReport reportOnEmployee(String employeeName) {
    EmployeeReport report;
    report = (EmployeeReport) session.readObject(EmployeeReport.class,
        new ExpressionBuilder.get("lastName").equal(employeeName));
    return report;
}

```

---

**WARNING:** Allowing an unverified SQL string to be passed into methods (for example: `readAllObjects(Class class, String sql)` and `readObject(Class class, String sql)` method) makes your application vulnerable to SQL injection attacks.

---

### **12.12.9.5 Reading Case 5: Inheritance Subclass Outer-Joining**

If you have an inheritance hierarchy that spans multiple tables and frequently query for the root class, consider using outer joining. This allows an outer-joining to be used for queries against an inheritance superclass that can read all of its subclasses in a single query instead of multiple queries.

Note that on some databases, the outer joins may be less efficient than the default multiple queries mechanism.

For more information about inheritance, see [Section 16.2.2, "Descriptors and Inheritance"](#).

For more information about querying on inheritance, see [Section 111.6, "Querying on an Inheritance Hierarchy"](#).

## 12.12.10 Write Optimization Examples

TopLink provides the write optimization features listed in [Table 12–14](#).

This section includes the following write optimization examples:

- [Writing Case: Batch Writes](#)

**Table 12–14 Write Optimization Features**

Feature	Effect on Performance
Unit of work	<p>Improves performance by updating only the changed fields and objects.</p> <p>Minimizes the amount of tracking required (which can be expensive) by registering only those objects that will change.</p> <p>For more information, see <a href="#">Chapter 113, "Introduction to TopLink Transactions"</a>.</p> <p>Note: The unit of work supports marking classes as read-only (see <a href="#">Section 119.3, "Configuring Read-Only Descriptors"</a> and <a href="#">Section 115.2, "Declaring Read-Only Classes"</a>). This avoids tracking of objects that do not change.</p>
Batch writing	<p>Lets you group all insert, update, and delete commands from a transaction into a single database call. This dramatically reduces the number of calls to the database (see <a href="#">Section 12.12.10.1.2, "Batch Writing and Parameterized SQL"</a>).</p>
Parameterized SQL	<p>Improves performance for frequently executed SQL statements (see <a href="#">Section 12.12.1, "How to Use Parameterized SQL and Prepared Statement Caching for Optimization"</a>).</p>
Sequence number preallocation	<p>Dramatically improves insert performance (see <a href="#">Section 12.12.10.1.3, "Sequence Number Preallocation"</a>).</p>
Multiprocessing	<p>Splitting a batch job across threads lets you synchronize reads from a cursor stream and use parallel units of work for performance improvements even on a single machine (see <a href="#">Section 12.12.10.1.4, "Multiprocessing"</a>).</p>
<i>Does exist</i> alternatives	<p>The <i>does exist</i> call on write object can be avoided in certain situations by checking the cache for <i>does exist</i>, or assuming the existence of the object (see <a href="#">Section 117.7, "Configuring Existence Checking at the Project Level"</a> or <a href="#">Section 119.17, "Configuring Cache Existence Checking at the Descriptor Level"</a> and <a href="#">Section 115.1.3, "How to Use Registration and Existence Checking"</a>).</p>
Change Tracking	<p>Improves writing and transactional read performance (see <a href="#">Section 113.2.3, "Unit of Work and Change Policy"</a> and <a href="#">Section 119.30, "Configuring Change Policy"</a>).</p>
Isolated Client Sessions	<p>For write-only, or non-cached (isolated) objects, the unit of work isolation level should be set to isolated-always to avoid caching overhead when not caching (see <a href="#">Section 102.2.7, "Cache Isolation"</a>).</p>

### 12.12.10.1 Writing Case: Batch Writes

The most common write performance problem occurs when a batch job inserts a large volume of data into the database. For example, consider a batch job that loads a large amount of data from one database, and then migrates the data into another. The following objects are involved:

- Simple individual objects with no relationships.



- Objects that use generated sequence numbers as their primary key.
- Objects that have an address that also uses a sequence number.

The batch job loads 10,000 employee records from the first database and inserts them into the target database. With no optimization, the batch job reads all the records from the source database, acquires a unit of work from the target database, registers all objects, and commits the unit of work.

#### **Example 12–16 No Optimization**

```

/* Read all the employees, acquire a unit of work, and register them */

// Read all the employees from the database. This requires 1 SQL call,
// but will be very memory intensive as 10,000 objects will be read
Vector employees = sourceSession.readAllObjects(Employee.class);

//SQL: Select * from Employee

// Acquire a unit of work and register the employees
UnitOfWork uow = targetSession.acquireUnitOfWork();
uow.registerAllObjects(employees);
uow.commit();

// SQL: Begin transaction
// SQL: Update Sequence set count = count + 1 where name = 'EMP'
// SQL: Select count from Sequence
// SQL: ... repeat this 10,000 times + 10,000 times for the addresses ...
// SQL: Commit transaction
// SQL: Begin transaction
// SQL: Insert into Address (...) values (...)
// SQL: ... repeat this 10,000 times
// SQL: Insert into Employee (...) values (...)
// SQL: ... repeat this 10,000 times
// SQL: Commit transaction

```

This batch job performs poorly, because it requires 60,000 SQL executions. It also reads huge amounts of data into memory, which can raise memory performance issues. TopLink offers several optimization features to improve the performance of this batch job.

To improve this operation, do the following:

- Use TopLink batch read operations and cursor support (see [Section 12.12.10.1.1, "Cursors"](#)).
- Use batch writing or parameterized batch writing to write to the database (see [Section 12.12.10.1.2, "Batch Writing and Parameterized SQL"](#)).  
If your database does not support batch writing, use parameterized SQL to implement the write query.
- Implement sequence number preallocation (see [Section 12.12.10.1.3, "Sequence Number Preallocation"](#)).
- Implement multiprocessing (see [Section 12.12.10.1.4, "Multiprocessing"](#)).

**12.12.10.1.1 Cursors** To optimize the query in [Example 12–16](#), use a cursored stream to read the Employees from the source database. You can also employ a weak identity map instead of a hard or soft cache identity map in both the source and target databases.

To address the potential for memory problems, use the `releasePrevious` method after each read to stream the cursor in groups of 100. Register each batch of 100 employees in a new unit of work and commit them.

Although this does not reduce the amount of executed SQL, it does address potential out-of-memory issues. When your system runs out of memory, the result is performance degradation that increases over time, and excessive disk activity caused by memory swapping on disk.

For more information, see [Section 12.12.7, "How to Use Cursored Streams and Scrollable Cursors for Optimization"](#).

**12.12.10.1.2 Batch Writing and Parameterized SQL** Batch writing lets you combine a group of SQL statements into a single statement and send it to the database as a single database execution. This feature reduces the communication time between the application and the server, and substantially improves performance.

You can enable batch writing alone (dynamic batch writing) using `Login` method `useBatchWriting`. If you add batch writing to [Example 12–16](#), you execute each batch of 100 employees as a single SQL execution. This reduces the number of SQL executions from 20,200 to 300.

You can also enable batch writing and parameterized SQL (parameterized batch writing) and prepared statement caching. Parameterized SQL avoids the prepare component of SQL execution. This improves write performance because it avoids the prepare cost of an SQL execution. For parameterized batch writing you would get one statement per Employee, and one for Address: this reduces the number of SQL executions from 20,200 to 400. Although this is more than dynamic batch writing alone, parameterized batch writing also avoids all parsing, so it is much more efficient overall.

Although parameterized SQL avoids the prepare component of SQL execution, it does not reduce the number of executions. Because of this, parameterized SQL alone may not offer as big of a gain as batch writing. However, if your database does not support batch writing, parameterized SQL will improve performance. If you add parameterized SQL in [Example 12–16](#), you must still execute 20,200 SQL executions, but parameterized SQL reduces the number of SQL PREPAREs to 4.

For more information, see [Section 12.11.3, "How to Use Batch Writing for Optimization"](#).

**12.12.10.1.3 Sequence Number Preallocation** SQL select calls are more resource-intensive than SQL modify calls, so you can realize large performance gains by reducing the number of select calls you issue. The code in [Example 12–16](#) uses the select calls to acquire sequence numbers. You can substantially improve performance if you use sequence number preallocation.

In TopLink, you can configure the sequence preallocation size on the login object (the default size is 50). [Example 12–16](#) uses a preallocation size of 1 to demonstrate this point. If you stream the data in batches of 100 as suggested in [Section 12.12.10.1.1, "Cursors"](#), set the sequence preallocation size to 100. Because employees and addresses in the example both use sequence numbering, you further improve performance by letting them share the same sequence. If you set the preallocation size to 200, this reduces the number of SQL execution from 60,000 to 20,200.

For more information about sequencing preallocation, see [Section 18.2.3, "Sequencing and Preallocation Size"](#).

**12.12.10.1.4 Multiprocessing** You can use multiple processes or multiple machines to split the batch job into several smaller jobs. In this example, splitting the batch job across threads enables you to synchronize reads from the cursored stream, and use parallel Units of Work on a single machine.

This leads to a performance increase, even if the machine has only a single processor, because it takes advantage of the wait times inherent in SQL execution. While one thread waits for a response from the server, another thread uses the waiting cycles to process its own database operation.

[Example 12–17](#) illustrates the optimized code for this example. Note that it does not illustrate multiprocessing.

**Example 12–17 Fully Optimized**

```

/* Read each batch of employees, acquire a unit of work, and register them */
targetSession.getLogin().useBatchWriting();
targetSession.getLogin().setSequencePreallocationSize(200);
targetSession.getLogin().bindAllParameters();
targetSession.getLogin().cacheAllStatements();
targetSession.getLogin().setMaxBatchWritingSize(200);

// Read all the employees from the database into a stream.
// This requires 1 SQL call, but none of the rows will be fetched.
ReadAllQuery query = new ReadAllQuery(Employee.class);
query.useCursoredStream();
CursoredStream stream;
stream = (CursoredStream) sourceSession.executeQuery(query);
//SQL: Select * from Employee. Process each batch
while (! stream.atEnd()) {
    Vector employees = stream.read(100);
    // Acquire a unit of work to register the employees
    UnitOfWork uow = targetSession.acquireUnitOfWork();
    uow.registerAllObjects(employees);
    uow.commit();
}
//SQL: Begin transaction
//SQL: Update Sequence set count = count + 200 where name = 'SEQ'
//SQL: Select count from Sequence where name = 'SEQ'
//SQL: Commit transaction
//SQL: Begin transaction
//BEGIN BATCH SQL: Insert into Address (...) values (...)
//... repeat this 100 times
//Insert into Employee (...) values (...)
//... repeat this 100 times
//END BATCH SQL:
//SQL: Commit transactionJava optimization

```

## 12.13 Optimizing the Unit of Work

For best performance when using a unit of work, consider the following tips:

- Register objects with a unit of work only if objects are eligible for change. If you register objects that will not change, the unit of work needlessly clones and processes those objects.
- Avoid the cost of existence checking when you are registering a new or existing object (see [Section 115.1.3](#), "How to Use Registration and Existence Checking").

- Avoid the cost of change set calculation on a class you know will not change by telling the unit of work that the class is read-only (see [Section 115.2, "Declaring Read-Only Classes"](#)).
- Avoid the cost of change set calculation on an object read by a `ReadAllQuery` in a unit of work that you do not intend to change by unregistering the object (see [Section 115.1.5, "How to Unregister Working Clones"](#)).
- Before using conforming queries, be sure that it is necessary. For alternatives, see [Section 115.4, "Using Conforming Queries and Descriptors"](#).
- Enable weaving and change tracking to greatly improve transactional performance. For more information, see [Section 12.14, "Optimizing Using Weaving"](#).

If your performance measurements show that you have a performance problem during unit of work commit, consider using object level or attribute level change tracking, depending on the type of objects involved and how they typically change. For more information, see [Section 113.2.3, "Unit of Work and Change Policy"](#).

## 12.14 Optimizing Using Weaving

Oracle recommends that you enable weaving to improve performance. Transactional performance in particular can be greatly improved through the use of weaving and change tracking.

In addition to using weaving to transparently configure lazy loading (indirection) and change tracking, TopLink uses weaving to make numerous internal optimizations.

For more information, see [Section 2.10, "Using Weaving"](#).

## 12.15 Optimizing the Application Server and Database Optimization

Configuring your application server and database correctly can have a big impact on performance and scalability. Ensure that you correctly optimize these key components of your application in addition to your TopLink application and persistence.

For your application or Java EE server, ensure your memory, thread pool and connection pool sizes are sufficient for your server's expected load, and that your JVM has been configured optimally.

Ensure that your database has been configured correctly for optimal performance and its expected load.

## 12.16 Optimizing Storage and Retrieval of Binary Data in XML

When working with Java API for XML Web Services (JAX-WS), you can use XML binary attachments to optimize the storage and retrieval of binary data in XML. Rather than storing the data as a base64 BLOB, you can optimize it by sending the data as a Multipurpose Internet Mail Extensions (MIME) attachment in order to retrieve it on the other end.

To make the use of XML binary attachments, register an instance of an `oracle.toplink.ox.attachment.XMLAttachmentMarshaller` or `XMLAttachmentUnmarshaller` with the binding framework. During a marshal operation, binary data will be handed into the `XMLAttachmentMarshaller`, which will be required to provide an ID that you can use at a later time to retrieve the data.

TopLink runtime supports MtOM and SwaRef-style attachments.

TopLink provides support for the following Java types as attachments:

- `java.awt.Image`
- `javax.activation.DataHandler`
- `javax.mail.internet.MimeMultipart`
- `javax.xml.transform.Source`
- `byte[]`
- `Byte[]`

You can generate schema and mappings based on JAXB classes for these types.

You can configure which mappings will be treated as attachments and set the MIME types of those attachments. You perform configurations using the following JAXB annotations:

- `XmlAttachmentRef`—Used on a `DataHandler` to indicate that this should be mapped to a `swaRef` in the XML schema. This means it should be treated as a `SwaRef` attachment.
- `XmlMimeType`—Specifies the expected MIME type of the mapping. When used on a `byte` array, this value should be passed into the `XMLAttachmentMarshaller` during a marshal operation. During schema generation, this will result in an `expectedContentType` attribute being added to the related element.
- `XmlInlineBinaryData`—Indicates that this binary field should always be written inline as `base64Binary` and never treated as an attachment.

For information on JAXB annotations, see Chapter 8 of the specification at <http://jcp.org/aboutJava/communityprocess/pfd/jsr222/index.html>.

Additionally, you have to set the schema type on a mapping going to binary if it is to be considered an attachment: it is either `base64Binary` or `swaRef`.

---



---

**Note:** TopLink lets you override treating an object as an attachment on a per-mapping basis.

---



---

Consider the following examples.

**Example 12–18 Using SwaRef**

```
public class Employee {

    @XmlAttachmentRef
    public DataHandler photo;
    ...
}
```

The preceding code yields the following XML schema type:

```
<xs:complexType name="employee">
  <xs:sequence>
    <xs:element name="photo" type="xs:swaRef"/>
  </xs:sequence>
</xs:complexType>
```

The XML would look as follows:

```
<employee>
  <photo>attachment_id</photo>
</employee>
```

**Example 12–19 Using MtOM Without mimeType**

```
public class Employee {
    public java.awt.Image photo;
    ...
}
```

The preceding code generates the following XML schema type:

```
<xs:complexType name="employee">
  <xs:sequence>
    <xs:element name="photo" type="base64Binary"/>
  </xs:sequence>
</xs:complexType>
```

The XML would look as follows:

```
<employee>
  <photo>
    <xop:Include href="attachment_id"/>
  </photo>
</employee>
```

**Example 12–20 Using MtOM with mimeType**

```
public class Employee {
    @XmlMimeType("image/jpeg")
    public java.awt.Image photo;
    ...
}
```

The preceding code generates the following XML schema type:

```
<xs:complexType name="employee">
  <xs:sequence>
    <xs:element name="photo"
      ns:expectedContentTypes="image/jpeg"
      type="xs:base64Binary"/>
  </xs:sequence>
</xs:complexType>
```

The XML would look as follows:

```
<employee>
  <photo>
    <xop:Include href="attachment_id"/>
  </photo>
</employee>
```

**Example 12–21 Using Binary Object with Forced Inline**

```
public class Employee {
    @XmlInlineBinaryData
    public java.awt.Image photo;
    ...
}
```

The preceding code generates the following XML schema type:

```
<xs:complexType name="employee">
  <xs:sequence>
    <xs:element name="photo" type="xs:base64Binary"/>
  </xs:sequence>
</xs:complexType>
```

The XML would look as follows:

```
<employee>
  <photo>ASWUHF13234230IJEUFHEIUFWE134DF03IR3298RY== </photo>
</employee>
```

If you are not using JAXB, use the `oracle.toplink.ox.mappings.XMLBinaryDataMapping` and `XMLBinaryDataCollectionMapping` API to handle binary data. For more information, see [Section 53.12, "XML Binary Data Mapping"](#) and [Section 53.13, "XML Binary Data Collection Mapping"](#).

### 12.16.1 How to Use an Attachment Marshaller and Unmarshaller

You implement `TopLink XMLAttachmentMarshaller` and `XMLAttachmentUnmarshaller` interfaces to add and retrieve various types of XML attachments. An `XMLMarshaller` holds an instance of `XMLAttachmentMarshaller`, and `XMLUnmarshaller`—an instance of `XMLAttachmentUnmarshaller`.

You set and obtain an attachment marshaller and unmarshaller using the following corresponding `XMLMarshaller` and `XMLUnmarshaller` methods:

- `setAttachmentMarshaller(XMLAttachmentMarshaller am)`
- `getAttachmentMarshaller()`
- `setAttachmentUnmarshaller(XMLAttachmentUnmarshaller au)`
- `getAttachmentUnmarshaller()`

[Example 12-22](#) shows how to use an attachment marshaller in your application.

#### **Example 12-22 Using an Attachment Marshaller**

```
...
XMLMarshaller marshaller = context.createMarshaller();
XMLAttachmentMarshaller am = new EmployeeAttachmentMarshaller();
marshaller.setAttachmentMarshaller(am);
...
```

For the preceding example to be valid, the XML schema type should be set to `swaRef`.

For more information, see [Section 47.1.1.3.1, "How to Use TopLink XMLContext"](#).





---

## Customizing the TopLink Application

There are multiple ways to customize your TopLink application, ranging from creating custom data types to using EclipseLink JPA extensions.

This chapter includes the following sections:

- [Introduction to Customization](#)
- [Creating Custom Data Types](#)
- [Using Public Source](#)
- [Using the Session Customizer Class](#)
- [Using the Descriptor Customizer Class](#)
- [Using the Descriptor Amendment Methods](#)
- [Using EclipseLink JPA Extensions](#)

### 13.1 Introduction to Customization

By design, TopLink can adapt to a variety of relational and nonrelational data sources.

To integrate TopLink with a data source that is not directly supported by the TopLink API, Oracle recommends that you use an EIS project (see [Chapter 71, "Introduction to EIS Projects"](#)) or a XML project (see [Chapter 47, "Introduction to XML Projects"](#)).

Using an EIS project, you can integrate your TopLink-enabled application with any nonrelational data source that supports a JCA adapter and any supported EIS record type, including indexed, mapped, or XML. If no JCA adapter exists for your target data source, you can concentrate your integration efforts on creating an adapter. Simultaneously, you can build your application according to JCA specifications. Although this still requires custom development effort, it is more efficient than trying to extend TopLink classes and provides you with a JCA adapter that you can leverage in any other project (making it a better value).

Using an XML project, you can integrate your TopLink-enabled application with Web services or other XML-message based designs.

The remainder of this chapter describes other customization options provided by the TopLink API.

### 13.2 Creating Custom Data Types

TopLink provides support for all the most common Java data types. [Table 13–1](#) lists the TopLink mapping extensions that you can use to support custom data types. You can

also create your object converter to allow conversion between a data type and your own Java type.

**Table 13–1 Mapping Extensions for Custom Data Types**

Extension	Description
Object type converter (see <a href="#">Section 17.2.6.3</a> , "Object Type Converter")	An extension of direct and direct collection mappings that lets you match a fixed number of data values to Java objects. Use this converter when the values in the schema differ from those in Java
Serialized object converter (see <a href="#">Section 17.2.6.1</a> , "Serialized Object Converter")	An extension of direct and direct collection mappings that lets you map serializable objects, such as multimedia data, to a binary format in a data source, such as a base64 element in an XML document or Binary Large Object (BLOB) field in a database
Type conversion converter (see <a href="#">Section 17.2.6.2</a> , "Type Conversion Converter")	An extension of direct and direct collection mappings that lets you explicitly map a data source type to a Java type. For example, a <code>java.util.Date</code> in Java can be mapped to a <code>java.sql.Date</code> in the data source.
Simple type translator (see <a href="#">Section 17.2.6.4</a> , "Simple Type Translator")	An extension of direct and direct collection mappings that lets you automatically translate an XML element value to an appropriate Java type based on the element's <code>&lt;type&gt;</code> attribute as defined in your XML schema.

### 13.3 Using Public Source

The source code to most public classes is available in `<TOPLINK_HOME>\jlib\toplink-src.zip`.

This is provided for debugging purposes.

### 13.4 Using the Session Customizer Class

You can customize a session at run time by specifying a session customizer—a Java class that implements the `oracle.toplink.tools.sessionconfiguration.SessionCustomizer` interface.

For more information, see the following:

- [Section 87.2.3](#), "Session Customization"
- [Section 89.8](#), "Configuring a Session Customizer Class"
- Persistence unit property `eclipselink.session.customizer` in "EclipseLink JPA Properties for Customization and Validation" table of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#How\\_to\\_Use\\_the\\_Persistence\\_Unit\\_Properties\\_for\\_Customization\\_and\\_Validation](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#How_to_Use_the_Persistence_Unit_Properties_for_Customization_and_Validation)

### 13.5 Using the Descriptor Customizer Class

You can customize a descriptor at run time by specifying a descriptor customizer—a Java class that implements the `oracle.toplink.tools.sessionconfiguration.DescriptorCustomizer` interface.

For more information, see the following:

- [Section 16.2.6](#), "Descriptor Customization"

- Section 119.34, "Configuring a Descriptor Customizer Class"
- "How to Use the @Customizer Annotation" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#How\\_to\\_Use\\_the\\_.40Customizer\\_Annotation](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#How_to_Use_the_.40Customizer_Annotation)
- Persistence unit property `eclipselink.descriptor.customizer.<ENTITY>` in "EclipseLink JPA Properties for Customization and Validation" table of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#How\\_to\\_Use\\_the\\_Persistence\\_Unit\\_Properties\\_for\\_Customization\\_and\\_Validation](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#How_to_Use_the_Persistence_Unit_Properties_for_Customization_and_Validation)

## 13.6 Using the Descriptor Amendment Methods

To customize descriptors, you can use their amendment methods.

For more information, see the following:

- Section 16.2.7, "Amendment and After-Load Methods"
- Section 119.35, "Configuring Amendment Methods"

## 13.7 Using EclipseLink JPA Extensions

If you are developing a EclipseLink JPA application, use EclipseLink JPA metadata annotations and XML extensions for customization.

For more information, see "Using EclipseLink JPA Extensions" chapter of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29).



# Part V

---

## Mapping and Configuration Overview

This part describes how to use TopLink to map persistent objects to a data source and how to capture that information for use with the TopLink run-time component. It contains the following chapter:

- [Chapter 14, "Introduction to TopLink Mapping and Configuration"](#)

This chapter introduces the metadata, contained in the descriptor, used by TopLink to generate SQL statements that create, read, modify, and delete objects.



---

# Introduction to TopLink Mapping and Configuration

TopLink uses metadata (see [Section 2.9, "Working with TopLink Metadata"](#)) to describe how objects relate to a data source representation. Your mapping and configuration activities construct this metadata.

After creating the metadata, you can use it in any number of applications by referencing the metadata from a session (see [Chapter 87, "Introduction to TopLink Sessions"](#)). The TopLink runtime uses this metadata in all persistence and data transformation operations.

This chapter includes the following section:

- [Mapping and Configuration Concepts](#)

## 14.1 Mapping and Configuration Concepts

This section describes concepts unique to TopLink mapping and configuration, including the following:

- [Projects](#)
- [Descriptors](#)
- [Mappings](#)

### 14.1.1 Projects

The `Project` class is the primary container in which TopLink stores its mapping and configuration metadata. A project relates a set of object classes to a data source at the data model level.

A project contains a descriptor (see [Section 14.1.2, "Descriptors"](#)) for each class and each descriptor contains a mapping (see [Section 14.1.3, "Mappings"](#)) for each data member that TopLink should persist or transform.

Using Oracle JDeveloper TopLink Editor or TopLink Workbench, you can export mapping and configuration metadata into a deployment XML file called `project`. For more information, see [Section 116.3, "Exporting Project Information"](#).

After creating the project XML file, you must associate it with a session so that TopLink can use it at run time. For more information, see [Section 89.2, "Configuring a Primary Mapping Project"](#).

For Enterprise JavaBeans (EJB) applications where there is no session, deploy the project XML file to the target application server. In this context, the project XML file is also known as the deployment XML file.

For more information, see the following:

- [Section 9.1.2, "sessions.xml File"](#)
- [Section 9.1.1, "project.xml File"](#)
- [Chapter 15, "Introduction to Projects"](#).

## 14.1.2 Descriptors

Descriptors describe how a Java class relates to a data source representation. They relate object classes to the data source at the data model level. For example, persistent class attributes may map to database columns.

TopLink uses descriptors to store the information that describes how an instance of a particular class can be represented in a data source (see [Section 14.1.3, "Mappings"](#)). Most descriptor information can be defined by Oracle JDeveloper TopLink Editor or TopLink Workbench, then read from the project XML file at run time.

See [Chapter 16, "Introduction to Descriptors"](#) for more information.

## 14.1.3 Mappings

Mappings describe how individual object attributes relate to a data source representation. Mappings can involve a complex transformation or a direct entry.

TopLink uses mappings to determine how to transform data between object and data source representation. Most mapping information can be defined by Oracle JDeveloper TopLink Editor or TopLink Workbench, then read from the project XML file at run time. Mappings are owned by descriptors (see [Section 14.1.2, "Descriptors"](#)).

See [Chapter 17, "Introduction to Mappings"](#) for more information.



# Part VI

---

## Projects

This part describes the TopLink artifact used to contain mapping and data source-specific information. It contains the following chapter:

- [Chapter 15, "Introduction to Projects"](#)

This chapter describes each of the different TopLink project types and important project concepts.

For information on specific project types, see the following parts:

- [Part IX, "Relational Projects"](#) (for both relational and object-relational data type applications)
- [Part XIV, "XML Projects"](#)
- [Part XVII, "EIS Projects"](#)



## Introduction to Projects

A TopLink project encapsulates both mapping metadata and, where relevant, data source access information. The project is the primary object used by TopLink at run time. Each session (excluding the session broker) in a deployed application references a single project.

This chapter includes the following sections:

- [TopLink Project Types](#)
- [Project Concepts](#)
- [Project API](#)
- [XML Namespaces Overview](#)

### 15.1 TopLink Project Types

Table 15–1 lists the project types available in TopLink, classifies each as basic or advanced, and indicates how to create each.

**Table 15–1** TopLink Project Types

Project Type	Description	Oracle JDeveloper	TopLink Workbench	Java
Relational (see <a href="#">Chapter 18, "Introduction to Relational Projects"</a> )	A project for transactional persistence of Java objects to a relational database or an object-relational data type database accessed using Java Database Connectivity (JDBC). Supports TopLink queries and expressions.	✓	✓	✓
EIS (see <a href="#">Chapter 71, "Introduction to EIS Projects"</a> )	A project for transactional persistence of Java objects to a nonrelational data source accessed using a Java EE Connector Architecture (JCA) adapter and any supported EIS record type, including indexed, mapped, or XML. Supports TopLink queries and expressions.	✓	✓	✓
XML (see <a href="#">Chapter 47, "Introduction to XML Projects"</a> )	A project for nontransactional, nonpersistent (in-memory) conversion between Java objects and XML schema (XSD)-based documents using Java Architecture for XML Binding (JAXB). Does not support TopLink queries and expressions.	✓	✓	✓

For more information, see the following:

- [Chapter 116, "Creating a Project"](#)
- [Chapter 117, "Configuring a Project"](#)
- [Chapter 87, "Introduction to TopLink Sessions"](#)

## 15.2 Project Concepts

This section describes concepts unique to TopLink projects, including the following:

- [Project Architecture](#)
- [Relational and Nonrelational Projects](#)
- [Persistent and Nonpersistent Projects](#)
- [Projects and Login](#)
- [Projects and Platforms](#)
- [Projects and Sequencing](#)
- [XML Namespaces](#)

### 15.2.1 Project Architecture

The project type you choose determines the type of descriptors and mappings you can use. There is a project type for each data source type that TopLink supports.

Table 15–2 summarizes the relationship between project, descriptor, and mappings.

**Table 15–2 Project, Descriptor, and Mapping Support**

Project	Descriptor	Mapping
Relational (see <a href="#">Chapter 18</a> , "Introduction to Relational Projects")	The following: <ul style="list-style-type: none"> <li>■ Relational (see <a href="#">Section 21.1</a>, "Relational Descriptors")</li> <li>■ Object-relational data type (see <a href="#">Section 24.1</a>, "Object-Relational Data Type Descriptors")</li> </ul>	The following: <ul style="list-style-type: none"> <li>■ Relational (see <a href="#">Section 17.4</a>, "Relational Mappings")</li> <li>■ Object-relational data type (see <a href="#">Section 17.5</a>, "Object-Relational Data Type Mappings")</li> </ul>
EIS (see <a href="#">Chapter 71</a> , "Introduction to EIS Projects")	EIS (see <a href="#">Section 74.1</a> , "EIS Descriptor Concepts")	EIS (see <a href="#">Section 17.7</a> , "EIS Mappings")
XML (see <a href="#">Chapter 47</a> , "Introduction to XML Projects")	XML (see <a href="#">Section 50.1</a> , "XML Descriptor Concepts")	XML (see <a href="#">Section 17.6</a> , "XML Mappings")

### 15.2.2 Relational and Nonrelational Projects

TopLink supports both relational and nonrelational projects.

Relational projects persist Java objects to a relational database.

Nonrelational projects persist Java objects to another (nonrelational) type of data source, or perform nonpersistent (see [Section 15.2.3](#), "Persistent and Nonpersistent Projects") data conversion. For example, to persist Java objects to an EIS data source by using a JCA adapter, use an EIS project. To perform nonpersistent (in-memory) conversions between Java objects and XML elements, use an XML project.

### 15.2.3 Persistent and Nonpersistent Projects

TopLink supports projects you use for applications that require persistent storage of Java objects. For example, use a relational project to persist Java objects to a relational database, or an EIS project to persist Java objects to an EIS data source by way of a JCA adapter.

TopLink also supports projects you use for applications that require only nonpersistent (in-memory) data conversion. For example, use an XML project to perform nonpersistent (in-memory) conversion between Java objects and XML elements.

## 15.2.4 Projects and Login

The login (if any) associated with a project determines how the TopLink runtime connects to the project's data source.

A login includes details of data source access, such as authentication, use of connection pools, and use of external transaction controllers. A login owns a platform.

A platform includes options specific to a particular data source, such as binding, use of native SQL, use of batch writing, and sequencing. For more information about platforms, see [Section 15.2.5, "Projects and Platforms"](#).

For projects that do not persist to a data source, a login is not required. For projects that do persist to a data source, a login is always required.

In TopLink Workbench, the project type determines the type of login that the project uses, if applicable.

You can use a login in a variety of roles. A login's role determines where and how you create it. The login role you choose depends on the type of project you are creating and how you intend to use the login:

- [POJO Session Role](#)
- [CMP Deployment Role](#)
- [Development Role](#)

### 15.2.4.1 POJO Session Role

You create a session login in the `sessions.xml` file for TopLink applications that do not use container-managed persistence (CMP).

Typically, the TopLink runtime instantiates a project when you load a session from the `sessions.xml` file (see [Chapter 90, "Acquiring and Using Sessions at Run Time"](#)). The runtime also instantiates a login and its platform based on the information in the `sessions.xml` file.

The TopLink runtime uses the information in the session login whenever you perform a persistence operation using the session in your POJO TopLink application.

In this case, you do not configure a deployment login (see [Section 15.2.4.2, "CMP Deployment Role"](#)).

If you are using TopLink Workbench and your login is based on a relational database platform, you must also configure a development login (see [Section 15.2.4.3, "Development Role"](#)).

If a `sessions.xml` file contains a login, this login overrides any other login definition.

There is a session login type for each project type that persists to a data source. For a complete list of login types available, see [Section 96.1.2, "Data Source Login Types"](#).

For information on configuring a session login, see [Section 89.3, "Configuring a Session Login"](#).

### 15.2.4.2 CMP Deployment Role

You create a deployment login in the `project.xml` file (also known as the `toplink-ejb-jar.xml` file) for a TopLink-enabled CMP application.

When you deploy your TopLink-enabled CMP application with its `toplink-ejb-jar.xml` file, the application server or EJB container uses the

information in the deployment login whenever your business logic performs a persistence operation from within an entity bean with container-managed persistence.

In this case, you do not configure a session login (see [Section 15.2.4.1, "POJO Session Role"](#)). In fact, there is no `session.xml` file at all (see [Section 9.1.2.4, "CMP Applications and Session Metadata"](#)).

If you are using TopLink Workbench and your login is based on a relational database platform, you must also configure a development login (see [Section 15.2.4.3, "Development Role"](#)).

For information on creating a deployment login, see [Section 20.5, "Configuring Development and Deployment Logins"](#).

### 15.2.4.3 Development Role

Using TopLink Workbench, you create a development login in the TopLink Workbench project file when your project is based on a relational database platform.

TopLink Workbench uses the information in the development login whenever you perform a data source operation from within TopLink Workbench, for example, whenever you read or write schema information from or to a data store during application development. The development login information is never written to a `sessions.xml` or `project.xml` file.

The development login is never used when you deploy your application: it is overridden by either the `sessions.xml` file (see [Section 15.2.4.1, "POJO Session Role"](#)) or the `project.xml` file (see [Section 15.2.4.2, "CMP Deployment Role"](#)).

For more information on creating a development login, see [Section 20.5, "Configuring Development and Deployment Logins"](#).

## 15.2.5 Projects and Platforms

The platform (if any) associated with a project tells the TopLink runtime what specific type of data source a project uses.

A platform includes options specific to a particular data source, such as binding, use of native SQL, use of batch writing, and sequencing.

A login includes details of data source access, such as authentication, use of connection pools, and use of external transaction controllers. A login owns a platform. For more information about logins, see [Section 15.2.4, "Projects and Login"](#).

For projects that do not persist to a data source, a platform is not required. For projects that do persist to a data source, a platform is always required.

In TopLink Workbench, the project type determines the type of platform that the project uses, if applicable.

There is a platform type for each project type that persists to a data source. For a complete list of platform types available, see [Section 96.1.3, "Data Source Platform Types"](#).

## 15.2.6 Projects and Sequencing

An essential part of maintaining object identity (see [Section 102.2.1, "Cache Type and Object Identity"](#)) is sequencing: managing the assignment of unique values to distinguish one instance from another.

Projects have different sequencing requirements, depending on their types:

- For relational projects, you typically obtain object identifier values from a separate sequence table (or database object) dedicated to managing object identifier values (see [Section 18.2, "Sequencing in Relational Projects"](#)).
- For EIS projects, you typically use a returning policy (see [Section 119.27, "Configuring Returning Policy"](#)) to obtain object identifier values managed by the EIS data source.
- For XML projects, because you are simply performing transformations between objects and XML documents, sequencing is not an issue.

To configure sequencing, you must configure the following:

- how to obtain sequence values (see [Section 15.2.6.1, "Configuring How to Obtain Sequence Values"](#)), and
- where to write sequence values when an instance of a descriptor's reference class is created (see [Section 15.2.6.2, "Configuring Where to Write Sequence Values"](#)).

Depending on the type of sequencing you use and the architecture of your application, you may consider using a sequence connection pool. For more information, see [Section 96.1.6.4, "Sequence Connection Pools"](#).

### 15.2.6.1 Configuring How to Obtain Sequence Values

To determine how TopLink obtains sequence values, you configure TopLink sequencing at the project or session level, depending on the type of project you are building, as follows:

- In a CMP project, you do not configure a session directly: in this case, you must configure sequences at the project level (see [Section 20.3, "Configuring Sequencing at the Project Level"](#)).
- In a POJO project, you can configure a session directly: in this case, you can use session-level sequence configuration instead of project-level sequence configuration or to override project level sequence configuration on a session-by-session basis, if required (see [Section 98.4, "Configuring Sequencing at the Session Level"](#)).

### 15.2.6.2 Configuring Where to Write Sequence Values

To tell TopLink into which table and column to write the sequence value when an instance of a descriptor's reference class is created, you configure TopLink sequencing at the descriptor level (see [Section 23.3, "Configuring Sequencing at the Descriptor Level"](#)).

## 15.2.7 XML Namespaces

As defined in <http://www.w3.org/TR/REC-xml-names/>, an XML namespace is a collection of names, identified by a URI reference, which are used in XML documents as element types and attribute names. To promote reusability and modularity, XML document constructs should have universal names, whose scope extends beyond their containing document. XML namespaces are the mechanism which accomplishes this.

XML namespaces are applicable in projects that reference an XML schema: EIS projects that use XML records (see [Section 72.2, "Creating an EIS Project with XML Records"](#)) and XML projects (see [Section 47.1, "XML Project Concepts"](#)).

For more information, see [Section 15.4, "XML Namespaces Overview"](#).

## 15.3 Project API

This section describes the following:

- [Project Inheritance Hierarchy](#)

### 15.3.1 Project Inheritance Hierarchy

There is only one type of project: `oracle.toplink.sessions.Project`.

## 15.4 XML Namespaces Overview

As defined in <http://www.w3.org/TR/REC-xml-names/>, an XML namespace is a collection of names, identified by a URI reference, which are used in XML documents as element types and attribute names. To promote reusability and modularity, XML document constructs should have universal names, whose scope extends beyond their containing document. XML namespaces are the mechanism which accomplishes this.

XML namespaces are applicable in projects that reference an XML schema: EIS projects that use XML records (see [Section 72.2, "Creating an EIS Project with XML Records"](#)) and XML projects (see [Section 47.1, "XML Project Concepts"](#)).

You can use Oracle JDeveloper or TopLink Workbench to configure the XML schema namespace for your project.

This section describes the following:

- [TopLink Workbench Namespace Resolution](#)
- [Element and Attribute Form Options](#)
- [TopLink Runtime Namespace Resolution](#)

### 15.4.1 TopLink Workbench Namespace Resolution

Using TopLink Workbench, you can configure the XML schema namespace for your project. For more information, see [Section 5.6.5, "How to Configure XML Schema Namespace"](#).

### 15.4.2 Element and Attribute Form Options

The `xsd:schema` element provides attributes that you can use to specify how elements and attributes should be qualified by namespace.

This section describes the consequences of the following combinations of element and attribute form configuration:

- [Element Form Default Qualified and Attribute Form Default Unqualified](#)
- [Element and Attribute Form Default Unqualified](#)
- [Element and Attribute Form Default Qualified](#)

#### 15.4.2.1 Element Form Default Qualified and Attribute Form Default Unqualified

[Example 15–1](#) shows an XML schema in which a target namespace is set. It is coded with `elementFormDefault` set to `qualified` and `attributeFormDefault` set to `unqualified`. This means all elements must be namespace qualified and globally declared attributes must be namespace qualified and locally defined attributes must not be namespace qualified.



**Example 15–1 XML Schema with Element Form Default Qualified and Attribute Form Default Unqualified**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  xmlns="urn:namespace-example"
  targetNamespace="urn:namespace-example">
  <xsd:element name="customer" type="customer-type"/>
  <xsd:complexType name="customer-type">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element ref="date-of-birth"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:integer"/>
  </xsd:complexType>
  <xsd:element name="date-of-birth" type="xsd:date"/>
</xsd:schema>
```

[Example 15–2](#) shows an XML document that conforms to this XML schema.

**Example 15–2 XML Document**

```
<?xml version="1.0" encoding="UTF-8"?>
<ns:customer xmlns:ns="urn:namespace-example" id="1">
  <ns:name>Jane Doe</ns:name>
  <ns:date-of-birth>1975-02-21</ns:date-of-birth>
</ns:customer>
```

[Example 15–3](#) shows the Java code for a Customer class `XMLDescriptor` and XML mappings for its attributes to illustrate how this schema configuration affects the XPath expressions you specify for default root element and mappings (for more information, see [Chapter 52, "Configuring an XML Descriptor"](#) and [Chapter 54, "Configuring an XML Mapping"](#)).

**Example 15–3 XML Descriptors and Mappings**

```
NamespaceResolver namespaceResolver = new NamespaceResolver();
namespaceResolver.put("ns", "urn:namespace-example");

XMLDescriptor customerDescriptor = new XMLDescriptor();
customerDescriptor.setJavaClass(Customer.class);
customerDescriptor.setDefaultRootElement("ns:customer");
customerDescriptor.setNamespaceResolver(namespaceResolver);

XMLDirectMapping idMapping = new XMLDirectMapping();
idMapping.setAttributeName("id");
idMapping.setXPath("@id");
customerDescriptor.addMapping(idMapping);

XMLDirectMapping nameMapping = new XMLDirectMapping();
nameMapping.setAttributeName("name");
nameMapping.setXPath("ns:name/text()");
customerDescriptor.addMapping(nameMapping);

XMLDirectMapping birthDateMapping = new XMLDirectMapping();
birthDateMapping.setAttributeName("birthDate");
birthDateMapping.setXPath("ns:date-of-birth/text()");
customerDescriptor.addMapping(birthDateMapping);
```

**15.4.2.2 Element and Attribute Form Default Unqualified**

[Example 15–4](#) shows an XML schema in which a target namespace is set. It is coded with `elementFormDefault` and `attributeFormDefault` set to `unqualified`.

This means that globally defined nodes must be namespace qualified and locally defined nodes must not be namespace qualified.

#### **Example 15–4 XML Schema with Element and Attribute Form Default Unqualified**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="unqualified"
  attributeFormDefault="unqualified"
  xmlns="urn:namespace-example"
  targetNamespace="urn:namespace-example">
  <xsd:element name="customer" type="customer-type"/>
  <xsd:complexType name="customer-type">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element ref="date-of-birth"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:integer"/>
  </xsd:complexType>
  <xsd:element name="date-of-birth" type="xsd:date"/>
</xsd:schema>
```

Example 15–5 shows an XML document that conforms to this XML schema.

#### **Example 15–5 XML Document**

```
<?xml version="1.0" encoding="UTF-8"?>
<ns:customer xmlns:ns="urn:namespace-example" id="1">
  <name>Jane Doe</name>
  <ns:date-of-birth>1975-02-21</ns:date-of-birth>
</ns:customer>
```

Example 15–6 shows the Java code for a Customer class XMLDescriptor and XML mappings for its attributes to illustrate how this schema configuration affects the XPath expressions you specify for default root element and mappings (for more information, see [Chapter 52, "Configuring an XML Descriptor"](#) and [Chapter 54, "Configuring an XML Mapping"](#)).

#### **Example 15–6 XML Descriptors and Mappings**

```
NamespaceResolver namespaceResolver = new NamespaceResolver();
namespaceResolver.put("ns", "urn:namespace-example");

XMLDescriptor customerDescriptor = new XMLDescriptor();
customerDescriptor.setJavaClass(Customer.class);
customerDescriptor.setDefaultRootElement("ns:customer");
customerDescriptor.setNamespaceResolver(namespaceResolver);

XMLDirectMapping idMapping = new XMLDirectMapping();
idMapping.setAttributeName("id");
idMapping.setXPath("@id");
customerDescriptor.addMapping(idMapping);

XMLDirectMapping nameMapping = new XMLDirectMapping();
nameMapping.setAttributeName("name");
nameMapping.setXPath("name/text()");
customerDescriptor.addMapping(nameMapping);

XMLDirectMapping birthDateMapping = new XMLDirectMapping();
birthDateMapping.setAttributeName("birthDate");
birthDateMapping.setXPath("ns:date-of-birth/text()");
customerDescriptor.addMapping(birthDateMapping);
```

### 15.4.2.3 Element and Attribute Form Default Qualified

[Example 15-7](#) shows an XML schema in which a target namespace is set. It is coded with `elementFormDefault` and `attributeFormDefault` set to `qualified`. This means that all nodes must be namespace qualified.

#### **Example 15-7 XML Schema with Element and Attribute Form Default Qualified**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="qualified"
  xmlns="urn:namespace-example"
  targetNamespace="urn:namespace-example">
  <xsd:element name="customer" type="customer-type"/>
  <xsd:complexType name="customer-type">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element ref="date-of-birth"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:integer"/>
  </xsd:complexType>
  <xsd:element name="date-of-birth" type="xsd:date"/>
</xsd:schema>
```

[Example 15-8](#) shows an XML document that conforms to this XML schema.

#### **Example 15-8 XML Document**

```
<?xml version="1.0" encoding="UTF-8"?>
<ns:customer xmlns:ns="urn:namespace-example" ns:id="1">
  <ns:name>Jane Doe</ns:name>
  <ns:date-of-birth>1975-02-21</ns:date-of-birth>
</ns:customer>
```

[Example 15-9](#) shows the Java code for a `Customer` class `XMLDescriptor` and XML mappings for its attributes to illustrate how this schema configuration affects the XPaths you specify for default root element and mappings (for more information, see [Chapter 52, "Configuring an XML Descriptor"](#) and [Chapter 54, "Configuring an XML Mapping"](#)).

#### **Example 15-9 XML Descriptors and Mappings**

```
NamespaceResolver namespaceResolver = new NamespaceResolver();
namespaceResolver.put("ns", "urn:namespace-example");

XMLDescriptor customerDescriptor = new XMLDescriptor();
customerDescriptor.setJavaClass(Customer.class);
customerDescriptor.setDefaultRootElement("ns:customer");
customerDescriptor.setNamespaceResolver(namespaceResolver);

XMLDirectMapping idMapping = new XMLDirectMapping();
idMapping.setAttributeName("id");
idMapping.setXPath("@ns:id");
customerDescriptor.addMapping(idMapping);

XMLDirectMapping nameMapping = new XMLDirectMapping();
nameMapping.setAttributeName("name");
nameMapping.setXPath("ns:name/text()");
customerDescriptor.addMapping(nameMapping);

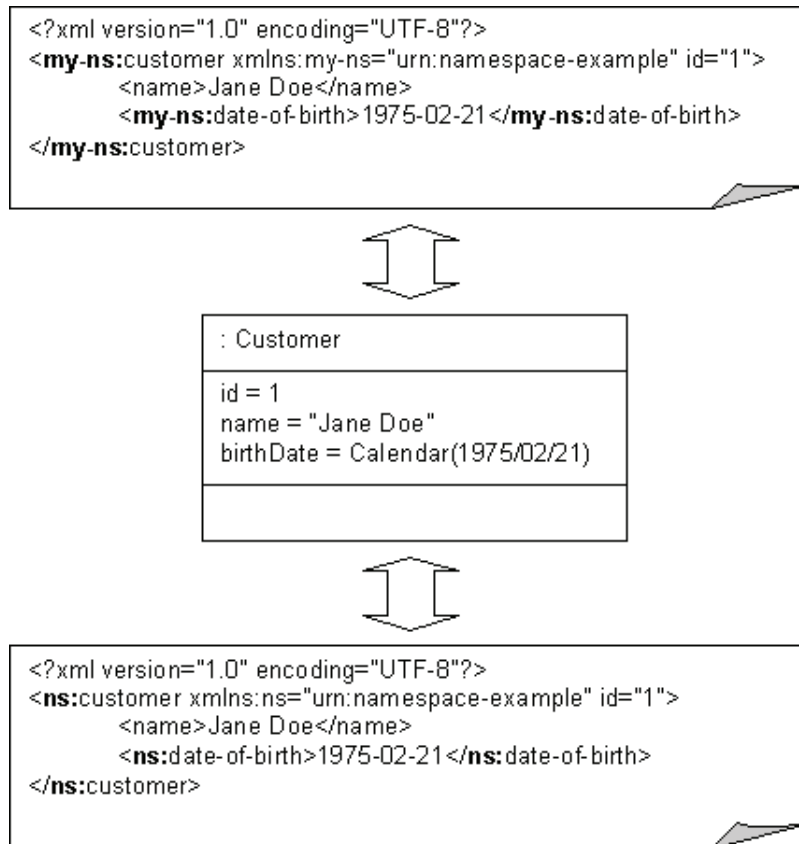
XMLDirectMapping birthDateMapping = new XMLDirectMapping();
birthDateMapping.setAttributeName("birthDate");
birthDateMapping.setXPath("ns:date-of-birth/text()");
customerDescriptor.addMapping(birthDateMapping);
```

### 15.4.3 TopLink Runtime Namespace Resolution

It is common for an XML document to include one or more namespaces. TopLink supports this using its `NamespaceResolver`. The namespace resolver maintains pairs of namespace prefixes and Uniform Resource Identifiers (URIs). TopLink uses these prefixes in conjunction with the XPath statements you specify on EIS mappings to XML records and XML mappings.

Although TopLink captures namespace prefixes in the XPath statements for mappings (if applicable), the input document is not required to use the same namespace prefixes. As [Example 15-1](#) shows, TopLink will use the namespace prefixes specified in the mapping when creating new documents.

**Figure 15-1** *Namespaces in TopLink*



# Part VII

---

## Descriptors

This part describes the TopLink artifact used to describe persistent objects. It contains the following chapter:

- [Chapter 16, "Introduction to Descriptors"](#)

This chapter describes each of the different TopLink descriptor types and important descriptor concepts.

For information on specific descriptor types, see the following parts:

- [Part X, "Relational Descriptors"](#)
- [Part XI, "Object-Relational Data Type Descriptors"](#)
- [Part XV, "XML Descriptors"](#)
- [Part XVIII, "EIS Descriptors"](#)



## Introduction to Descriptors

TopLink uses descriptors to store the information that describes how an instance of a particular class can be represented by a data source. Descriptors own mappings that associate class instance variables with a data source and transformation routines that are used to store and retrieve values. As such, the descriptor acts as the connection between a Java object and its data source representation.

This chapter includes the following sections:

- [Descriptor Types](#)
- [Descriptor Concepts](#)
- [Descriptors and Inheritance](#)
- [Descriptors and Locking](#)
- [Descriptor API](#)

### 16.1 Descriptor Types

Table 16–1 lists the descriptor types you use to describe the classes in your object model and classifies them as basic or advanced.

**Table 16–1** *TopLink Descriptor Types*

Descriptor Type	Description	Oracle JDeveloper	TopLink Workbench	Java
Relational (see <a href="#">Section 21.1</a> , "Relational Descriptors")	Describes Java objects that you map to tables in a relational database. Applicable to all relational databases that TopLink supports.	✓	✓	✓
Object-relational (see <a href="#">Section 24.1</a> , "Object-Relational Data Type Descriptors")	Describes Java objects that you map to tables in a relational database that provides special database data types that correspond more closely to object types. Applicable only to the relational databases that TopLink supports that provide these special data types.			✓
EIS (see <a href="#">Section 74.1</a> , "EIS Descriptor Concepts")	Describes Java objects that you map to an EIS data source by way of a JCA adapter.	✓	✓	✓
XML (see <a href="#">Section 50.1</a> , "XML Descriptor Concepts")	Describes Java objects that you map, in memory, to complex types in XML documents defined by an XML schema document (XSD).	✓	✓	✓

For more information, see the following:

- [Chapter 118](#), "Creating a Descriptor"
- [Chapter 119](#), "Configuring a Descriptor"

## 16.2 Descriptor Concepts

This section introduces descriptor concepts unique to TopLink, including the following:

- [Descriptor Architecture](#)
- [Descriptors and Inheritance](#)
- [Descriptors and CMP and BMP](#)
- [Fetch Groups](#)
- [Descriptors and Aggregation](#)
- [Descriptor Customization](#)
- [Amendment and After-Load Methods](#)
- [Descriptor Event Manager](#)
- [Descriptor Query Manager](#)
- [Descriptors and Sequencing](#)
- [Descriptors and Locking](#)
- [Default Root Element](#)

### 16.2.1 Descriptor Architecture

A **descriptor** stores all the information describing how an instance of a particular object class can be represented in a data source.

TopLink descriptors contain the following information:

- The persistent Java class it describes and the corresponding data source (database tables, XML complex type, or EIS interaction)
- A collection of mappings, which describe how the attributes and relationships for that class are stored in the database
- The primary key information (or equivalent) of the data source
- A list of query keys (or aliases) for field names
- Information for sequence numbers
- A set of optional properties for tailoring the behavior of the descriptor, including support for caching refresh options, identity maps, optimistic locking, the event manager, and the query manager

There is a descriptor type for each data source type that TopLink supports. In some cases, multiple descriptor types are valid for the same data source type. The type of descriptor you use determines the type of mappings that you can define.

[Table 16–2](#) summarizes the relationship between project, descriptor, and mappings.



**Table 16–2 Project, Descriptor, and Mapping Support**

Project	Descriptor	Mapping
Relational (see Chapter 18, "Introduction to Relational Projects")	The following: <ul style="list-style-type: none"> <li>Relational (see Section 21.1, "Relational Descriptors")</li> <li>Object-relational data type (see Section 24.1, "Object-Relational Data Type Descriptors")</li> </ul>	The following: <ul style="list-style-type: none"> <li>Relational (see Section 17.4, "Relational Mappings")</li> <li>Object-relational data type (see Section 17.5, "Object-Relational Data Type Mappings")</li> </ul>
EIS (see Chapter 71, "Introduction to EIS Projects")	EIS (see Section 74.1, "EIS Descriptor Concepts")	EIS (see Section 17.7, "EIS Mappings")
XML (see Chapter 47, "Introduction to XML Projects")	XML (see Section 50.1, "XML Descriptor Concepts")	XML (see Section 17.6, "XML Mappings")

## 16.2.2 Descriptors and Inheritance

**Inheritance** describes how a derived (child) class inherits the characteristics of its superclass (parent). You can use descriptors to describe the inheritance relationships between classes in relational, EIS, and XML projects.

In the descriptor for a child class, you can override mappings that have been specified in the descriptor for a parent class, or map attributes that have not been mapped at all in the parent class descriptor.

For more information, see [Section 16.3, "Descriptors and Inheritance"](#).

## 16.2.3 Descriptors and CMP and BMP

You can use descriptors to describe the characteristics of entity beans with container-managed or bean-managed persistence.

When mapping enterprise beans, you create a descriptor for the bean class: you do not create a descriptor for the local interface, remote interface, home class, or primary key class.

When using TopLink Workbench, you must define the project with the correct entity bean type (such as entity beans with container-managed or bean-managed persistence) and import the `ejb-jar.xml` file for the beans into the TopLink Workbench project.

For CMP projects, you use the `ejb-jar.xml` file to define the bean's mapped attributes. A descriptor of a bean with container-managed persistence contains a CMP policy used to configure CMP-specific options.

---



---

**Note:** For EJB 3.0 projects, you can use annotations to define the bean's mapped attributes.

---



---

This section describes the following:

- [Nondeferred Changes](#)
- [Creating a New Entity Bean and `ejbCreate` / `ejbPostCreate` Methods](#)
- [Inheritance](#)

### 16.2.3.1 Nondeferred Changes

By default, TopLink defers all changes until commit time: this is the most efficient approach that produces the least number of data source interactions.

Alternatively, you can configure an entity bean's descriptor for nondeferred changes. This means that as you change the persistent fields of the entity bean, TopLink CMP modifies the relational schema immediately.

Using nondeferred changes, you can achieve backward compatibility with the native behavior of some EJB containers. You can also accommodate advanced applications that rely on the database and entity changes being synchronized for such things as triggers or stored procedures based on transient state within the transaction, deletion and creation of rows with the same primary key, or other complex queries that depend on transient transaction state.

Nondeferred changes have the disadvantage of being the least efficient approach: they produce the greatest number of data source interactions.

When you configure TopLink CMP to support nondeferred changes, TopLink will continue to handle constraints for mapped relationships among entity beans with the same deferral setting. However, you are responsible for handling any errors that result from making changes to a class that is not deferred, but related to a class that is deferred when a constraint exists between these two classes.

---

---

**Note:** When you configure a descriptor for nondeferred changes, TopLink CMP does not apply nondeferred changes to dependent objects. Dependent objects are subject to default deferred changes: the relational schema is not modified until commit.

---

---

For more information, see [Section 119.18, "Configuring a Descriptor with EJB CMP and BMP Information"](#).

### 16.2.3.2 Creating a New Entity Bean and `ejbCreate` / `ejbPostCreate` Methods

When you create a new entity bean, by default, the bean's life cycle can be thought of as follows:

1. `ejbCreate` method:

After the insert, the EJB container retrieves the primary key allocated by the database for the created instance.

For a relational project:

- a. `INSERT INTO ...`
- b. `SELECT FROM ...`

For an EIS project:

- a. Write object ...
- b. Find object ...

2. `ejbPostCreate` method:

The EJB container updates container-managed relationship (CMR) fields. The EJB container needs the primary key obtained in the `ejbCreate` method.

For a relational project:

- a. `UPDATE SET ...`

For an EIS project:

- a. Write object ...

However, if you have non-null foreign key constraints in your database, doing a data source modification after the `ejbCreate` method executes can cause problems. To get around this, some application servers, such as, for example, OC4J, allow you to create new objects after the `ejbPostCreate` method executes, and rely on the container to resolve the foreign key constraint.

For more information, see [Section 119.18, "Configuring a Descriptor with EJB CMP and BMP Information"](#).

### 16.2.3.3 Inheritance

`TopLink` allows you to configure inheritance for CMP descriptors, with some reservations.

For more information, see [Section 16.3.5, "Inheritance and CMP and BMP"](#).

## 16.2.4 Fetch Groups

By default, when you execute an object-level read query for a particular object class, `TopLink` returns all the persistent attributes mapped in the object's descriptor. With this single query, all the object's persistent attributes are defined, and calling their `get` methods returns the value directly from the object.

When you are interested in only some of the attributes of an object, it may be more efficient to return only a subset of the object's attributes using a fetch group with which you can define a subset of an object's attributes and associate the fetch group with either a `ReadObjectQuery` or `ReadAllQuery` query.

For more information, see the following:

- [Section 119.33, "Configuring Fetch Groups"](#)
- [Section 108.7.1.6, "Fetch Groups and Object-Level Read Queries"](#)

## 16.2.5 Descriptors and Aggregation

Two objects—a source (parent or owning) object and a target (child or owned) object—are related by aggregation if there is a strict one-to-one relationship between them, and all the attributes of the target object can be retrieved from the same data source representation as the source object. This means that if the source object exists, then the target object must also exist, and if the source object is destroyed, then the target object is also destroyed.

In this case, the descriptors for the source and target objects must be designated to reflect this relationship.

In EJB 3.0, an aggregate is known as an embeddable. In the EJB 3.0 specification, an embeddable may not contain another embeddable (that is, the EJB 3.0 specification does not support nested aggregates).

For more information, see the following:

- [Section 21.2, "Aggregate and Composite Descriptors in Relational Projects"](#)
- [Section 74.2, "EIS Descriptors and Aggregation"](#)
- [Section 50.1.1.1, "Composite Descriptors in XML Projects"](#)

## 16.2.6 Descriptor Customization

You can customize a descriptor at run time by specifying a descriptor customizer—a Java class that implements the

`oracle.toplink.tools.sessionconfiguration.DescriptorCustomizer` interface and provides a default (zero-argument) constructor.

You use a descriptor customizer to customize a descriptor at run time through code API similar to how you use an amendment method to customize a descriptor (see [Section 16.2.7, "Amendment and After-Load Methods"](#)).

For more information, see [Section 119.34, "Configuring a Descriptor Customizer Class"](#).

## 16.2.7 Amendment and After-Load Methods

Using TopLink Workbench, you can associate a static Java method that is called when a descriptor is loaded at run time. This method can amend the run-time descriptor instance through the descriptor Java code API. Use this method to make some advanced configuration options that may not be currently supported by TopLink Workbench.

You can only modify descriptors before the session has been connected; you should not modify descriptors after the session has been connected.

For more information, see [Section 119.35, "Configuring Amendment Methods"](#).

## 16.2.8 Descriptor Event Manager

In relational and EIS projects, TopLink raises various instances of `DescriptorEvent` (see [Table 119-26](#) and [Table 119-28](#)) during the persistence life cycle. Each descriptor owns an instance of `DescriptorEventManager` that is responsible for receiving these events and dispatching them to the descriptor event handlers registered with it.

Using a descriptor event handler, you can execute your own application specific logic whenever descriptor events occur, allowing you to take customized action at various points in the persistence life-cycle. For example, using a descriptor event handler, you can do the following:

- Synchronize persistent objects with other systems, services, and frameworks.
- Maintain nonpersistent attributes of which TopLink is not aware.
- Notify other objects in the application when the persistent state of an object changes.
- Implement complex mappings or optimizations not directly supported by TopLink mappings.

For more information, see the following:

- [Section 119.24, "Configuring a Domain Object Method as an Event Handler"](#)
- [Section 119.25, "Configuring a Descriptor Event Listener as an Event Handler"](#)

## 16.2.9 Descriptor Query Manager

Each relational and EIS descriptor provides an instance of `DescriptorQueryManager` that you can use to configure the following:

- named queries (see [Section 119.7, "Configuring Named Queries at the Descriptor Level"](#))
- custom default queries for basic persistence operations (see [Section 108.13.2, "How to Configure Default Query Implementations"](#))

- additional join expressions (see [Section 108.13.3, "How to Configure Additional Join Expressions"](#))

For more information on using the query manager, see [Section 108.13, "Descriptor Query Manager Queries"](#).

### 16.2.10 Descriptors and Sequencing

An essential part of maintaining object identity is managing the assignment of unique values (that is, a specific sequence) to distinguish one object instance from another. For more information, see [Section 15.2.6, "Projects and Sequencing"](#).

Sequencing options you configure at the project (or session) level determine the type of sequencing that TopLink uses. In a CMP project, you typically configure the sequence type at the project level (see [Section 20.3, "Configuring Sequencing at the Project Level"](#)). In a POJO project, you can use session-level sequence configuration to override project-level sequence configuration, on a session-by-session basis, if required (see [Section 98.4, "Configuring Sequencing at the Session Level"](#)).

After configuring the sequence type, for each descriptor's reference class, you must associate one attribute, typically the attribute used as the primary key (see [Section 119.2, "Configuring Primary Keys"](#)), with its own sequence (see [Section 23.3, "Configuring Sequencing at the Descriptor Level"](#)).

### 16.2.11 Descriptors and Locking

You can configure a descriptor with any of the following locking policies to control concurrent access to a domain object:

- **Optimistic**—All users have read access to the data. When a user attempts to make a change, the application checks to ensure the data has not changed since the user read the data (see [Section 16.4.1, "Optimistic Version Locking Policies"](#) and [Section 16.4.4, "Optimistic Field Locking Policies"](#)).
- **Pessimistic**—The first user who accesses the data with the purpose of updating it locks the data until completing the update (see [Section 16.4.5, "Pessimistic Locking Policy"](#)).
- **No locking**—The application does not prevent users overwriting each other's changes.

Oracle recommends using optimistic locking for most types of applications to ensure that users do not overwrite each other's changes.

For more information, see the following:

- [Section 16.4, "Descriptors and Locking"](#)
- [Section 119.26, "Configuring Locking Policy"](#)

### 16.2.12 Default Root Element

You configure EIS root descriptors ([Section 76.3, "Configuring Default Root Element"](#)) and XML descriptors ([Section 52.4, "Configuring Default Root Element"](#)) with a default root element so that the TopLink runtime knows the data source data type associated with the class the descriptor describes.

---



---

**Note:** The undefined document root element of a referenced object is ignored during marshalling with an any collection mapping and object mapping.

---



---

This section describes what a default root element is and how TopLink uses it. Consider the `Customer` and `Address` classes and their mappings, shown in [Example 16-1](#).

### Example 16-1 Customer and Address Classes

**Class:** `Customer`

**Default Root:** `customer`

**Attributes and Mappings:**

<code>name:String</code>	Direct Mapping to	<code>name/text()</code>
<code>billingAddress:Address</code>	Composite Object Mapping to	<code>billing-address</code>
<code>shippingAddress:Address</code>	Composite Object Mapping to	<code>shipping-address</code>

**Class:** `Address`

**Default Root:** `address`

**Attributes and Mappings:**

<code>street:String</code>	Direct Mapping to	<code>street/text()</code>
<code>city:String</code>	Direct Mapping to	<code>city/text()</code>

These classes correspond to the XML schema, shown in [Example 16-2](#).

### Example 16-2 Customer and Address Schema

```
<xsd:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xsd:complexType name="address-type">
    <xsd:sequence>
      <element name="street" type="xsd:string"/>
      <element name="city" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="customer" type="customer-type"/>
  <xsd:complexType name="customer-type">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="billing-address" type="address-type"/>
      <xsd:element name="shipping-address" type="address-type"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

When an instance of the `Customer` class is persisted to XML, the TopLink runtime performs the following:

1. Gets the default root element.

The `Customer` class instance corresponds to the root of the XML document. The TopLink runtime uses the default root element specified on the descriptor (`customer`) to start the XML document. TopLink then uses the mappings on the descriptor to marshal the object's attributes:

```
<customer>
  <name>...</name>
</customer>
```

2. When the TopLink runtime encounters an object attribute such as `billingAddress`, it checks the mapping associated with it to determine with what element (`billing-address`) to continue:

```
<customer>
  <name>...</name>
```

```

        <billing-address/>
    </customer>

```

The TopLink runtime checks the mapping's reference descriptor (Address) to determine what attributes to persist:

```

<customer>
  <name>...</name>
  <billing-address>
    <street>...</street>
    <city>...</city>
  </billing-address>
</customer>

```

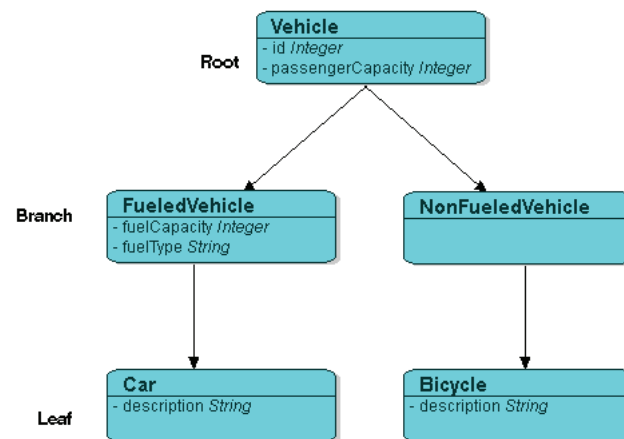
## 16.3 Descriptors and Inheritance

Inheritance describes how a derived class inherits the characteristics of its superclass. You can use descriptors to describe the inheritance relationships between classes in relational, EIS, and XML projects.

Figure 16–1 illustrates the `Vehicle` object model—a typical Java inheritance hierarchy. The root class `Vehicle` contains two branch classes: `FueledVehicle` and `NonFueledVehicle`. Each branch class contains a leaf class: `Car` and `Bicycle`, respectively.

**Figure 16–1 Example Inheritance Hierarchy**

Java Inheritance Hierarchy:



TopLink recognizes the following three types of classes in an inheritance hierarchy:

1. The root class stores information about *all* instantiable classes in its subclass hierarchy. By default, queries performed on the root class return instances of the root class and its instantiable subclasses. However, the root class can be configured so queries on it return only instances of itself, without instances of its subclasses.

For example, the `Vehicle` class in Figure 16–1 is a root class.

2. Branch classes have a persistent superclass and also have subclasses. By default, queries performed on the branch class return instances of the branch class and any of its subclasses. However, as with the root class, the branch class can be configured so queries on it return only instances of itself without instances of its subclasses.

For example, the `FueledVehicle` class in [Figure 16-1](#) is a branch class.

3. Leaf classes have a persistent superclass in the hierarchy but do not have subclasses. Queries performed on the leaf class can only return instances of the leaf class.

For example, the `Car` class in [Figure 16-1](#) is a leaf class.

In the descriptor for a child class, you can override mappings that have been specified in the descriptor for a parent class, or map attributes that have not been mapped at all in the parent class descriptor.

This section includes information on the following topics:

- [How to Specify a Class Indicator](#)
- [Inheritance and Primary Keys](#)
- [Single and Multi-Table Inheritance](#)
- [Aggregate and Composite Descriptors and Inheritance](#)
- [Inheritance and CMP and BMP](#)

For more information about configuring inheritance for a parent (root) class descriptor, see [Section 119.21, "Configuring Inheritance for a Parent \(Root\) Descriptor"](#).

For more information about configuring inheritance for a child (branch or leaf) class descriptor, see [Section 119.20, "Configuring Inheritance for a Child \(Branch or Leaf\) Class Descriptor"](#).

### 16.3.1 How to Specify a Class Indicator

When configuring inheritance, you configure the root class descriptor with the means of determining which subclasses it should instantiate.

You can do this in one of the following ways:

- [Using Class Indicator Fields](#)
- [Using Class Extraction Methods](#)

---



---

**Note:** All leaf classes in the hierarchy must have a class indicator and they must have the same type of class indicator (field or class extraction method).

---



---

#### 16.3.1.1 Using Class Indicator Fields

You can use a persistent attribute of a class to indicate which subclass should be instantiated. For example, in a relational descriptor, you can use a class indicator field in the root class table. The indicator field should not have an associated direct mapping unless it is set to read-only.

---



---

**Note:** If the indicator field is part of the primary key, define a write-only transformation mapping for the indicator field (see [Chapter 39, "Configuring a Relational Transformation Mapping"](#)).

---



---

You can use strings or numbers as values in the class indicator field.

The root class descriptor must specify how the value in the class indicator field translates into the class to be instantiated.



One approach is to configure the root class descriptor with a class indicator dictionary: a collection of key-values that associates a simple key, stored in the class indicator field, with a class to instantiate. [Table 16–3](#) illustrates the class indicator dictionary for the `Vehicle` class' subclasses, as shown in [Figure 16–1](#).

**Table 16–3 Class Indicator Dictionary for the Vehicle Class**

Key	Value
F	FueledVehicle
N	NonFueledVehicle
C	Car
B	Bicycle

Another approach is to simply use the class name itself as the value stored in the class indicator field. This avoids having to define unique indicators for each class at the expense of a slightly larger key value (depending on the length of your class names).

### 16.3.1.2 Using Class Extraction Methods

You can define a Java method to compute the class indicator based on any available information in the object's data source record. Such a method is called a class extraction method.

Using a class extraction method, you do not need to include an explicit class indicator field in your data model and you can handle relationships that are too complex to describe using class indicator fields.

A class extraction method must have the following characteristics:

- it must be defined on the root descriptor's class;
- it must be static;
- it must take a `Record` as an argument;
- it must return the `java.lang.Class` object to use for the `Record` passed in.

You may also need to define only-instances and with-all-subclasses expressions (see [Section 16.3.1.2.1, "Specifying Expressions for Only-Instances and With-All-Subclasses"](#)).

For example, [Table 16–4](#) lists the rows in the `EMPLOYEE` table. The `Employee` class is the base class. `Director`, `Manager`, `Programmer`, and `TechWriter` classes each derive from the `Employee` class. However, in your application, instances of `Manager`, `Programmer`, and `TechWriter` classes must be represented as `Employee` instances and instances of `Director` must be represented as `Director` instances. Because there is no a one-to-one correspondence between class and `JOB_TYPE` field value, the `JOB_TYPE` field alone cannot serve as a class indicator field (see [Section 16.3.1.1, "Using Class Indicator Fields"](#)). To resolve this issue, you could use the class extraction method, shown in [Example 16–3](#).

**Table 16–4 EMPLOYEE Table**

ID	NAME	JOB_TYPE	JOB_TITLE
732	Bob Jones	1	Manager
733	Sarah Smith	3	Technical Writer
734	Ben Ng	2	Director

**Table 16–4 (Cont.) EMPLOYEE Table**

ID	NAME	JOB_TYPE	JOB_TITLE
735	Sally Johnson	3	Programmer

**Example 16–3 Class Extraction Method**

```

...
// If the JOB_TYPE field value in record equals 2, return the Director class.
// Return the Employee class for all other JOB_TYPE field values

public static Class getClassFromRecord(Record record) {
    if (record.get("JOB_TYPE").equals(new Integer(2)) {
        return Director.class;
    }
    else {
        return Employee.class;
    }
}

```

When configuring inheritance using a class extraction method, Oracle TopLink does not generate SQL for queries on the root class.

**16.3.1.2.1 Specifying Expressions for Only-Instances and With-All-Subclasses** If you use a class extraction method (see [Section 16.3.1.2, "Using Class Extraction Methods"](#)), you must provide TopLink with expressions to correctly filter sibling instances for all classes that share a common table (see [Section 119.22, "Configuring Inheritance Expressions for a Parent \(Root\) Class Descriptor"](#)).

## 16.3.2 Inheritance and Primary Keys

For relational and EIS projects, TopLink assumes that all of the classes in an inheritance hierarchy have the same primary key, as set in the root descriptor.

For more information, see the following:

- [Section 21.3.1, "Inheritance and Primary Keys in Relational Projects"](#)
- [Section 74.3.1, "Inheritance and Primary Keys in EIS Projects"](#)

## 16.3.3 Single and Multi-Table Inheritance

In a relational project, you can map your inheritance hierarchy to a single table (see [Section 21.3.2.1, "Single-Table Inheritance"](#)) or to multiple tables (see [Section 21.3.2.2, "Multi-Table Inheritance"](#)).

## 16.3.4 Aggregate and Composite Descriptors and Inheritance

You can designate relational descriptors as aggregates, and EIS descriptors as composites. XML descriptors are always composites (see [Section 50.1.1, "XML Descriptors and Aggregation"](#)).

When configuring inheritance for a relational aggregate descriptor, all the descriptors in the inheritance tree must be aggregates. The descriptors for aggregate and non-aggregate classes cannot exist in the same inheritance tree.

Similarly, when configuring inheritance for an EIS composite descriptor, all the descriptors in the inheritance tree must be composites. The descriptors for composite and noncomposite classes cannot exist in the same inheritance tree.

When configuring inheritance for an XML descriptor, because all XML descriptors are composites, descriptor type does not restrict inheritance.

### 16.3.5 Inheritance and CMP and BMP

Although inheritance is a standard tool in object-oriented modeling, the EJB specifications prior to 3.0 contain only general information regarding inheritance. You should fully understand this information before implementing EJB inheritance. Be aware of the fact that future EJB specifications may dictate inheritance guidelines not supported by all application servers.

## 16.4 Descriptors and Locking

This section describes the various types of locking policy that TopLink supports, including the following:

- [Optimistic Version Locking Policies](#)
- [Optimistic Version Locking Policies and Cascading](#)
- [Optimistic Locking and Rollbacks](#)
- [Optimistic Field Locking Policies](#)
- [Pessimistic Locking Policy](#)
- [Locking in a Three-Tier Application](#)

For more information, see [Section 119.26, "Configuring Locking Policy"](#).

### 16.4.1 Optimistic Version Locking Policies

With optimistic locking, all users have read access to the data. When a user attempts to make a change, the application checks to ensure the data has not changed since the user read the data.

Optimistic version locking policies enforce optimistic locking by using a version field (also known as a write-lock field) that you provide in the reference class that TopLink updates each time an object change is committed.

TopLink caches the value of this version field as it reads an object from the data source. When the client attempts to write the object, TopLink compares the cached version value with the current version value in the data source in the following way:

- If the values are the same, TopLink updates the version field in the object and commits the changes to the data source.
- If the values are different, the write operation is disallowed because another client must have updated the object since this client initially read it.

TopLink provides the following version-based optimistic locking policies:

- `VersionLockingPolicy`: requires a *numeric* version field; TopLink updates the version field by incrementing its value by one.
- `TimestampLockingPolicy`: requires a *timestamp* version field; TopLink updates the version field by inserting a new timestamp (this policy can be configured to get the time from the data source or locally; by default, the policy gets the time from the data source).

---

---

**Note:** In general, Oracle recommends numeric version locking because of the following:

- accessing the timestamp from the data source can have a negative impact on performance;
  - time stamp locking is limited to the precision that the database stores for timestamps.
- 
- 

Whenever any update fails because optimistic locking has been violated, TopLink throws an `OptimisticLockException`. This should be handled by the application when performing any database modification. The application must notify the client of the locking contention, refresh the object, and have the client reapply its changes.

You can choose to store the version value in the object as a mapped attribute, or in the cache. In three-tier applications, you typically store the version value in the object to ensure it is passed to the client when updated (see [Section 16.4.6, "Locking in a Three-Tier Application"](#)).

If you store the version value in the cache, you do not need to map it. If you do map the version field, you must configure the mapping as read-only (see [Section 121.2, "Configuring Read-Only Mappings"](#)).

To ensure that the parent object's version field is updated whenever a privately owned child object is modified, consider [Section 16.4.2, "Optimistic Version Locking Policies and Cascading"](#).

When using optimistic version locking with the unit of work, consider [Section 115.11, "Using Optimistic Read Locking with the `forceUpdateToVersionField` Method"](#).

If you are using a stored procedure to update or delete an object, your database may not return the row-count required to detect an optimistic lock failure, so your stored procedure is responsible for checking the optimistic lock version and throwing an error if they do not match. Only version locking is directly supported with a `StoredProcedureCall`. Because timestamp and field locking require two versions of the same field to be passed to the call, an SQL call that uses an `##` parameter to access the translation row could be used for other locking policies. For more information, see [Section 109.5, "Using a `StoredProcedureCall`"](#) and [Section 109.6, "Using a `StoredFunctionCall`"](#).

## 16.4.2 Optimistic Version Locking Policies and Cascading

If your database schema is such that both a parent object and its privately owned child object are stored in the same table, then if you update the child object, the parent object's version field will be updated.

However, if the parent and its privately owned child are stored in separate tables, then changing the child will not, by default, update the parent's version field.

To ensure that the parent object's version field is updated in this case, you can either manually update the parent object's version field (see [Section 115.11, "Using Optimistic Read Locking with the `forceUpdateToVersionField` Method"](#)) or, if you are using a `VersionLockingPolicy`, you can configure TopLink to automatically cascade the child object's version field update to the parent (see [Section 119.26.2.2, "Configuring Optimistic Locking Policy Cascading"](#)).

After you enable optimistic version locking cascading, when a privately owned child object is modified, TopLink will traverse the privately owned foreign reference mappings, updating all the parent objects back to the root.

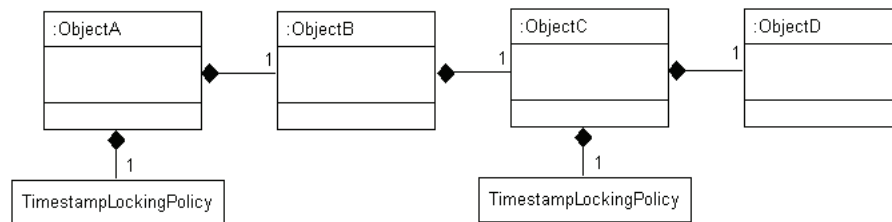
Optimistic version locking cascading is only applied if the child object is registered in a unit of work.

TopLink supports optimistic version locking cascading for:

- object changes in privately owned one-to-one and one-to-many mappings
- relationship changes (adding or removing) in the following collection mappings (privately owned or not):
  - direct collection
  - one-to-many
  - many-to-many
  - aggregate collection

Consider the example object graph shown in [Figure 16–2](#)

**Figure 16–2 Optimistic Version Locking Policies and Cascading Example**



In this example, `ObjectA` privately owns `ObjectB`, and `ObjectB` privately owns `ObjectC`, and `ObjectC` privately owns `ObjectD`.

Suppose you register `ObjectB` in a unit of work, modify an `ObjectB` field, and commit the unit of work. In this case, `ObjectB` checks the cache for `ObjectA` and, if not present, queries the database for `ObjectA`. `ObjectB` then notifies `ObjectA` of its change. `ObjectA` forces an update on its version optimistic locking field even though it has no changes to its corresponding table.

Suppose you register `ObjectA` in a unit of work, access its `ObjectB` to access its `ObjectC` to access its `ObjectD`, modify an `ObjectD` field, and commit the unit of work. In this case, `ObjectD` notifies `ObjectC` of its changes. `ObjectC` forces an update on its version optimistic locking field even though it has no changes to its corresponding table. `ObjectC` then notifies `ObjectB` of the `ObjectD` change. `ObjectB` then notifies `ObjectA` of the `ObjectD` change. `ObjectA` forces an update on its version optimistic locking field even though it has no changes to its corresponding table.

### 16.4.3 Optimistic Locking and Rollbacks

With optimistic locking, use the `UnitOfWork` method `commitAndResumeOnFailure` (see [Section 115.6, "Resuming a Unit of Work After Commit"](#)) to rollback a locked object's value, if you store the optimistic lock versions in the cache.

If you store the locked versions in an object, you must refresh the objects (or their versions) on a failure. Alternatively, you can acquire a new unit of work on the failure and reapply any changes into the new unit of work.

## 16.4.4 Optimistic Field Locking Policies

Optimistic field locking policies enforce optimistic locking by using one or more of the fields that currently exist in the table to determine if the object has changed since the client read the object.

The unit of work caches the original state of the object when you first read the object or register it with the unit of work. At commit time, the unit of work compares the original values of the lock fields with their current values on the data source during the update. If any of the lock field's values have changed, an optimistic lock exception is thrown.

TopLink provides the following optimistic field locking policies:

- **AllFieldsLockingPolicy:** For update and delete operations, TopLink compares all the fields of the object with all the fields in the data source. If the original value of any fields differ from that in the data source, the write operation is disallowed.

For example, if you changed a customer's last name, TopLink might produce SQL similar to the following:

```
UPDATE CUSTOMER SET LNAME='new last name' WHERE ID=7 AND LNAME='old last name'
AND FNAME='Donald' AND B_DAY='1972' AND CREDIT_RATING='A+' AND EYE_COLOR='Blue'
```

The main disadvantage of this field locking policy is that it is not the most efficient, especially if the changed object has many attributes.

---



---

**Note:** This comparison is only on a per table basis. If an update operation is performed on an object that is mapped to multiple tables (multiple table inheritance), then only the changed fields for each table changed appear in the *where* clause.

---



---

- **ChangedFieldsLockingPolicy:** For update operations, TopLink compares only the fields of the object that have changed with the corresponding fields in the data source. If the original value of any such field differs from that in the data source, the write operation is disallowed. TopLink does not make any field comparisons for deletes.

The main advantage of this field locking policy is that it allows concurrent updates of different fields. For example, if one thread updates a customer's last name and another thread updates the same customer's credit rating, and you configure the *Customer* descriptor with *ChangedFieldsLockingPolicy*, then TopLink might produce SQL like:

```
// Unit of work 1
UPDATE CUSTOMER SET LNAME='new name' WHERE ID=7 AND LNAME='old name'
// Unit of work 2
UPDATE CUSTOMER SET CREDIT_RATING='B' WHERE ID=7 AND CREDIT_RATING='A+'
```

- **SelectedFieldsLockingPolicy:** For update and delete operations, TopLink compares only the selected fields of the object with the corresponding fields in the data source. If the cached value of any such field differs from that in the data source, the write operation is disallowed.

For example, if you select *Customer* attributes *LNAME* and *CREDIT\_RATING*, then at run time, TopLink might produce SQL like:

```
UPDATE CUSTOMER SET LNAME='new name' WHERE ID=7 AND LNAME='old name' AND
CREDIT_RATING='A+'
```

Whenever any update fails because optimistic locking has been violated, TopLink throws an *OptimisticLockException*. This should be handled by the application

when performing any database modification. The application must notify the client of the locking contention, refresh the object, and have the client reapply its changes.

When using field locking policies, a unit of work must be employed for updating the data source.

---

---

**Note:** You cannot use an instance of `FieldsLockingPolicy` if you are using `AttributeChangeTrackingPolicy` (see [Section 113.2.3.3, "Attribute Change Tracking Policy"](#)).

---

---

## 16.4.5 Pessimistic Locking Policy

With pessimistic locking, the first user who accesses the data with the purpose of updating it locks the data until completing the update.

When using a pessimistic locking policy, you can configure the policy to either fail immediately or to wait until the read lock is acquired.

You can use a pessimistic locking policy only in a project with a container-managed persistence type (see [Section 117.5, "Configuring Persistence Type"](#)) and with descriptors that have EJB information (see [Section 119.18, "Configuring a Descriptor with EJB CMP and BMP Information"](#)).

You can also use pessimistic locking (but not a pessimistic locking policy) at the query level (see [Section 119.7.1.9, "Configuring Named Query Options"](#)).

TopLink provides an optimization for pessimistic locking when this locking is used with entity beans with container-managed persistence: if you set your query to pessimistic locking and run the query in its own new transaction (which will end after the execution of the finder), then TopLink overrides the locking setting and does not append `FOR UPDATE` to the SQL. However, the use of this optimization may produce an undesirable result if the pessimistic lock query has been customized by the user with a SQL string that includes `FOR UPDATE`. In this case, if the conditions for the optimization are present, the query will be reset to nonpessimistic locking, but the SQL will remain the same resulting in the locking setting of the query conflicting with the query's SQL string. To avoid this problem, you can take one of the following two approaches:

- Use EJB QL or a TopLink's expression (see [Chapter 110, "Introduction to TopLink Expressions"](#)) for the selection criteria. This will give TopLink control over the SQL generation.
- Place the finder in a transaction to eliminate conditions for the optimization.

## 16.4.6 Locking in a Three-Tier Application

If you are building a three-tier application, in order to correctly lock an object, you must obtain the lock before the object is sent to client for editing.

### 16.4.6.1 Optimistic Locking in a Three-Tier Application

If you are using optimistic locking, you have the following two choices for locking objects correctly:

1. Map the optimistic lock field in your object as not read-only and pass the version to the client on the read and back to the server on the update.

You must define a non-read-only mapping for the version field and make the optimistic locking policy store the version value in the object, not the cache (in

TopLink Workbench, this is done on the Locking tab by unchecking **Store Version in Cache** (see [Section 119.26.1, "How to Configure Locking Policy Using TopLink Workbench"](#))).

Ensure that the original version value is sent to the client when it reads the object for the update. The client must then pass the original version value back with the update information, and this version must be set into the object to be updated after it is registered/read in the new unit of work on the server.

2. Hold the unit of work for the duration of the interaction with the client.

Either through a stateful session bean, or in an HTTP session, store the unit of work used to read the object for the update for the duration of the client interaction.

You must read the object through this unit of work before passing it to the client for the update. This ensures that the version value stored in the unit of work cache or in the unit of work clone will be the original value.

This same unit of work must be used for the update.

The first option is more commonly used, and is required if developing a stateless application.

#### 16.4.6.2 Pessimistic Locking in a Three-Tier Application

If you are using pessimistic locking, you must use the unit of work to start a database transaction before the object is read. You must hold this unit of work and database transaction while the client is editing the object and until the client updates the object. You must use this same unit of work to update the object. If you are building a three-tier Web application (where it is not normally desirable to hold a database transaction open across client interactions), optimistic locking is normally more desirable than pessimistic locking (see [Section 16.4.6.1, "Optimistic Locking in a Three-Tier Application"](#)).

## 16.5 Descriptor API

The descriptor API can be used to define, or amend TopLink descriptors through Java code. The descriptor API classes are mainly in the `oracle.toplink.descriptors` package. These include the following classes:

- `ClassDescriptor` (abstract generic descriptor API)
- `RelationalDescriptor` (relational project-specific API)
- `DescriptorEventManager` (event API)
- `DescriptorQueryManager` (query API)
- `InheritancePolicy`
- `InterfacePolicy`
- `ReturningPolicy`
- Locking policies (various optimistic locking policies)

For object-relational data type, EIS, and XML projects, descriptor classes are in the `oracle.toplink.objectrelational`, `oracle.toplink.eis`, and `oracle.toplink.ox` packages, respectively.

This section describes the important descriptor classes in the Oracle TopLink Foundation Library, including the [Descriptor Inheritance Hierarchy](#).



## 16.5.1 Descriptor Inheritance Hierarchy

[Example 16–4](#) illustrates the descriptor types that derive from class `oracle.toplink.descriptors.ClassDescriptor`.

### **Example 16–4** *Descriptor Inheritance Hierarchy*

```
class oracle.toplink.descriptors.ClassDescriptor
  class oracle.toplink.descriptors.RelationalDescriptor
    class oracle.toplink.objectrelational.ObjectRelationalDescriptor
  class oracle.toplink.eis.EISDescriptor
  class oracle.toplink.ox.XMLDescriptor
```



# Part VIII

---

## Mappings

This part provides general mapping information. It contains the following chapter:

- [Chapter 17, "Introduction to Mappings"](#)

This chapter describes each of the different TopLink mapping types and important mapping concepts.

For information on specific mapping types, see the following parts:

- [Part XII, "Relational Mappings"](#)
- [Part XIII, "Object-Relational Data Type Mappings"](#)
- [Part XVI, "XML Mappings"](#)
- [Part XIX, "EIS Mappings"](#)



---



---

## Introduction to Mappings

TopLink can transform data between an object representation and a representation specific to a data source. This transformation is called mapping and it is the core of a TopLink project.

A mapping corresponds to a single data member of a domain object. It associates the object data member with its data source representation and defines the means of performing the two-way conversion between object and data source.

This chapter includes the following sections:

- [Mapping Types](#)
- [Mapping Concepts](#)
- [Mapping API](#)
- [Relational Mappings](#)
- [Object-Relational Data Type Mappings](#)
- [XML Mappings](#)
- [EIS Mappings](#)

### 17.1 Mapping Types

Table 17-1 describes the mapping types that TopLink supports.

**Table 17-1** *TopLink Mapping Types*

Type	Description	Oracle JDeveloper	TopLink Workbench	Java
Relational (see <a href="#">Section 17.4</a> , "Relational Mappings")	Mappings that transform any object data member type to a corresponding relational database (SQL) data source representation in any supported relational database. Relational mappings allow you to map an object model into a relational data model.	✓	✓	✓

**Table 17–1 (Cont.) TopLink Mapping Types**

Type	Description	Oracle JDeveloper	TopLink Workbench	Java
Object-relational data type (see <a href="#">Section 17.5, "Object-Relational Data Type Mappings"</a> )	Mappings that transform certain object data member types to structured data source representations optimized for storage in specialized object-relational data type databases, such as Oracle Database. Object-relational data type mappings let you map an object model into an object-relational data type data model.			✓
EIS (see <a href="#">Section 17.7, "EIS Mappings"</a> )	Mappings that transform object data members to the EIS record format defined by the object's descriptor.	✓	✓	✓
XML (see <a href="#">Section 17.6, "XML Mappings"</a> )	Mappings that transform object data members to the XML elements of an XML document whose structure is defined by an XML schema document (XSD).	✓	✓	✓

For more information, see the following:

- [Chapter 120, "Creating a Mapping"](#)
- [Chapter 121, "Configuring a Mapping"](#)

## 17.2 Mapping Concepts

This section describes concepts unique to TopLink mappings, including the following:

- [Mapping Architecture](#) (applicable to relational and nonrelational mappings)
- [Example Mapping](#) (applicable to relational and nonrelational mappings)
- [Automatic Mappings](#)
  - [JPA Automapping](#) (applicable to relational mappings)
  - [Automapping with TopLink Workbench at Development Time](#) (applicable to relational and nonrelational mappings)
  - [Default Mapping in EJB 2.n CMP Projects Using OC4J at Run Time](#) (applicable to relational mappings)
  - [JAXB Project Generation at Development Time](#) (applicable to XML mappings)
- [Indirection \(Lazy Loading\)](#) (applicable to relational mappings)
  - [Value Holder Indirection](#)
  - [Transparent Indirect Container Indirection](#)
  - [Proxy Indirection](#)
  - [Weaved Indirection](#)
  - [Indirection and JPA](#)
  - [Indirection and EJB 2.n CMP](#)
  - [Indirection, Serialization, and Detachment](#)
- [Method Accessors and Attribute Accessors](#) (applicable to relational and nonrelational mappings)
- [Mapping Converters and Transformers](#)

- [Serialized Object Converter](#) (applicable to relational and nonrelational mappings)
- [Type Conversion Converter](#) (applicable to relational and nonrelational mappings)
- [Object Type Converter](#) (applicable to XML mappings)
- [Simple Type Translator](#) (applicable to XML mappings)
- [Transformation Mappings](#) (applicable to relational and nonrelational mappings)
- [Mappings and XPath](#) (applicable to XML mappings)
- [Mappings and xsd:list and xsd:union Types](#) (applicable to XML mappings)
- [Mappings and the jaxb:class Customization](#) (applicable to XML mappings)
- [Mappings and JAXB Typesafe Enumerations](#) (applicable to XML mappings)

## 17.2.1 Mapping Architecture

To define a mapping, you draw upon the following components:

- The data representation specific to the data source (such as a relational database table or schema-defined XML element) in which you store the object's data.
- A descriptor for a particular object class.
- An object class to map.

---



---

**Note:** A mapping is the same regardless of whether your project is persistent or nonpersistent.

---



---

For an example of a typical TopLink mapping, see [Section 17.2.2, "Example Mapping"](#).

The type of data source you define in your TopLink project determines the type of mappings you can use and how you configure them. In a persistent project, you use mappings to persist to a data source. In a nonpersistent project, you use mappings simply to transform between the object format and some other data representation (such as XML). For more information about data source and project types, see [Section 15.1, "TopLink Project Types"](#).

A descriptor represents a particular domain object: it describes the object's class. It owns mappings: one mapping for each of the class data members that you intend to persist or transform in memory.

---



---

**Note:** Persistence is applicable at the descriptor level.

---



---

For more information about descriptors, see [Chapter 16, "Introduction to Descriptors"](#).

TopLink provides mappings to handle a wide variety of data types and data representations. For more information, see [Section 17.1, "Mapping Types"](#).

All mappings are subclasses of the `oracle.toplink.mappings.DatabaseMapping` class. For more information about the mapping API, see [Section 17.3, "Mapping API"](#).

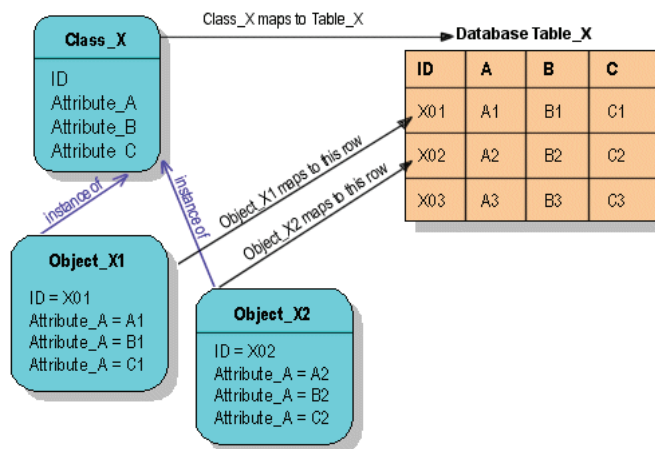
## 17.2.2 Example Mapping

Although TopLink supports more complex mappings, most TopLink classes map to a single database table or XML element that defines the type of information available in the class. Each object instance of a given class maps to a single row comprising the object's attributes, plus an identifier (the primary key) that uniquely identifies the object.

Figure 17–1 illustrates the simplest database mapping case in which:

- **Table\_X** in the database represents **Class\_X**.
- **Object\_X1** and **Object\_X2** are instances of **Class\_X**.
- Individual rows in **Table\_X** represent **Object\_X1** and **Object\_X2**, as well as any other instances of **Class\_X**.

**Figure 17–1 How Classes and Objects Map to a Database Table**



TopLink provides you with the tools to build these mappings, from the simple mappings illustrated in Figure 17–1, to complex mappings.

For an additional example of a relational mapping, see Figure 27–1, "Direct-to-Field Mapping".

For an example of a nonrelational mapping, see Figure 53–34, "XML Transformation Mappings".

## 17.2.3 Automatic Mappings

Typically, you use Oracle JDeveloper TopLink Editor or TopLink Workbench to define mappings on a class-by-class and data-member-by-data-member basis manually (see Section 120.2, "Creating Mappings Manually During Development").

Alternatively, you can take advantage of the following:

- [JPA Automapping](#)
- [Automapping with Oracle JDeveloper at Development Time](#)
- [Automapping with TopLink Workbench at Development Time](#)
- [Default Mapping in EJB 2.n CMP Projects Using OC4J at Run Time](#)
- [JAXB Project Generation at Development Time](#)



### 17.2.3.1 JPA Automapping

To configure automapping in a JPA project, you just need to annotate your persistence classes with `@Entity` and define their primary key with `@Id` (or define the list of entities and their primary key fields in your `orm.xml`) and the EclipseLink JPA persistence provider will automatically map all unmapped properties. You can also configure `persistence.xml` properties to automatically create or replace the corresponding database tables. For more information, see "Introduction to EclipseLink JPA" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Introduction\\_to\\_EclipseLink\\_JPA\\_%28ELUG%29](http://wiki.eclipse.org/Introduction_to_EclipseLink_JPA_%28ELUG%29)

### 17.2.3.2 Automapping with Oracle JDeveloper at Development Time

You can use Oracle JDeveloper TopLink Editor **Automap** feature to automatically define default mappings for every class and data member in your project (see [Section 120.3, "Creating Mappings Automatically During Development"](#)).

### 17.2.3.3 Automapping with TopLink Workbench at Development Time

You can use TopLink Workbench **Automap** feature to automatically define default mappings for every class and data member in your project (see [Section 120.3, "Creating Mappings Automatically During Development"](#)).

TopLink Workbench automapping is available for all project types and assumes that both the object model and database schema are already defined.

### 17.2.3.4 Default Mapping in EJB 2.n CMP Projects Using OC4J at Run Time

**Default mapping** is a relational persistence framework term that refers to making the framework automatically generate the object descriptor metadata (including such things as mappings, login data, database platform, locking, and foreign keys).

---

---

**Note:** You can apply default mapping to relational projects only.

---

---

TopLink can also optionally create or drop-and-create the tables associated with the entities during the deployment. This means that TopLink can handle the whole deployment process with the minimum requirements: a compliant EAR file and a valid data source. This frees you from creating tables and specifying mappings before you deploy your application.

TopLink default mapping supports the following features:

- Direct-to-field mapping support for standard CMP (that is, not dependent object) fields.
- Serialized object mapping support for dependent objects.
- One-to-one, one-to-many, and many-to-many mappings support for CMR fields.
- Self-referencing, unidirectional and bidirectional relationship mappings (see [Section 27.2.1, "Directionality"](#)).
- Optimistic version locking for each entity.
- Automatic table drop and create, platform-specified supported types, default size and subsizes, and database-reserved keywords.
- EJB QL (EJB query language) queries, such as **finder** and **ejbSelect**.

- Unknown primary key class case (primary key-class type `java.lang.Object.class`).

Default mapping is available for CMP relational projects deployed to OC4J configured to use TopLink as the persistence manager. In this configuration, the EJB container provides the entity bean descriptor data (from `ejb-jar.xml`) required by the persistence manager to generate the persistence descriptor file.

If a `toplink-ejb-jar.xml` descriptor file is not present, TopLink, working as the OC4J persistence manager, generates a default persistence descriptor file for any CMP project during deployment. In this case, TopLink applies default mappings and, optionally, automatic table generation. The generated descriptor file includes the following:

- Mapping for each entity CMP and CMR field.
- Optimistic locking, foreign keys, target foreign key, and relation table.
- Transparent indirection (lazy loading) for relationships.
- Database login and platform metadata.

If a `toplink-ejb-jar.xml` descriptor file is present and specified in the `orion-ejb-jar.xml` file, TopLink does not apply default mapping: it honours the mappings specified in the `toplink-ejb-jar.xml` file. In this case, you can still configure automatic table generation.

For more information, see [Section 9.9.1.3, "Configuring default-mapping Properties"](#).

### 17.2.3.5 JAXB Project Generation at Development Time

JAXB provides an API and a tool that allow automatic two-way mapping between XML documents and Java objects. The JAXB compiler generates all the Java classes and mappings based on the provided Document Type Definition (DTD) and a schema definition.

For more information on JAXB, see *Architecture for XML Binding (JAXB): A Primer* at <http://java.sun.com/developer/technicalArticles/xml/jaxb/index.html>

For more information on XML projects, see [Chapter 47, "Introduction to XML Projects"](#).

For more information on XML mappings, see [Chapter 53, "Introduction to XML Mappings"](#).

## 17.2.4 Indirection (Lazy Loading)

By default, when TopLink retrieves a persistent object, it retrieves all of the dependent objects to which it refers. When you configure indirection (also known as lazy reading, lazy loading, and just-in-time reading) for an attribute mapped with a relationship mapping, TopLink uses an indirection object as a place holder for the referenced object: TopLink defers reading the dependent object until you access that specific attribute. This can result in a significant performance improvement, especially if the application is interested only in the contents of the retrieved object, rather than the objects to which it is related.

Oracle strongly recommends using indirection for all relationship mappings. Not only does this let you optimize data source access, but it also allows TopLink to optimize the unit of work processing, cache access, and concurrency.

---



---

**Note:** The use of indirection is especially important for providing a proper maintenance of bidirectional relationships (see [Section 2.14.3.4, "Maintaining Bidirectional Relationships"](#)). In this case, you must use indirection. If you are operating with collections, you must use transparent indirection (see [Section 17.2.4.2, "Transparent Indirect Container Indirection"](#)).

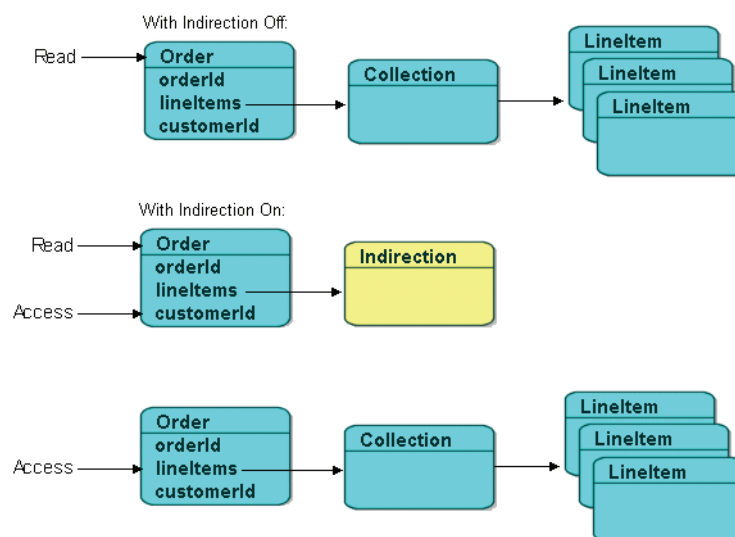
---



---

Figure 17–2 shows an indirection example. Without indirection, reading the `Order` object also reads the dependent collection of `LineItem` objects. With indirection, reading the `Order` object does not read the dependent collection of `LineItem` objects: the `lineItems` attribute refers to an indirection object. You can access other attributes (such as `customerId`), but `TopLink` reads the dependent `LineItem` objects only if and when you access the `lineItems` attribute.

**Figure 17–2 TopLink Indirection**



TopLink supports the following types of indirection:

- [Value Holder Indirection](#)
- [Transparent Indirect Container Indirection](#)
- [Proxy Indirection](#)

When using indirection with CMP, the version of EJB and application server you use affects how indirection is configured and what types of indirection are applicable (see [Section 17.2.4.6, "Indirection and EJB 2.n CMP"](#)).

When using indirection with an object that your application serializes, you must consider the effect of any untriggered indirection objects at deserialization time (see [Section 17.2.4.7, "Indirection, Serialization, and Detachment"](#)).

For information on configuring indirection, see [Section 121.3, "Configuring Indirection \(Lazy Loading\)"](#).

#### 17.2.4.1 Value Holder Indirection

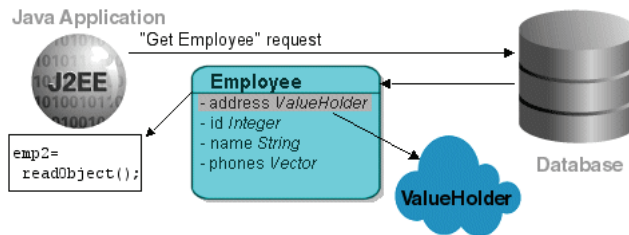
Persistent classes that use indirection must replace relationship attributes with value holder attributes. A value holder is an instance of a class that implements the

ValueHolderInterface interface, such as ValueHolder. This object stores the information necessary to retrieve the object it is replacing from the database. If the application does not access the value holder, the replaced object is never read from the database.

To obtain the object that the value holder replaces, use the `getValue` and `setValue` methods of the ValueHolderInterface. A convenient way of using these methods is to hide the `getValue` and `setValue` methods of the ValueHolderInterface inside `get` and `set` methods, as shown in the following illustrations.

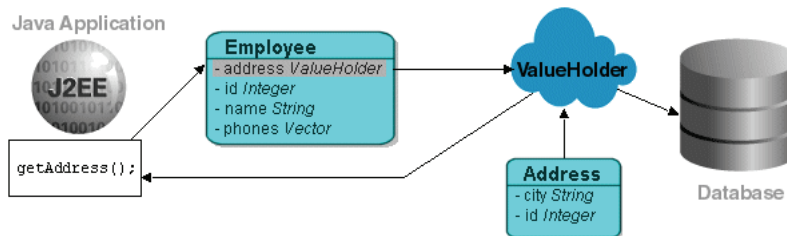
Figure 17-3 shows the Employee object being read from the database. The Address object is not read and will not be created unless it is accessed.

**Figure 17-3 Address Object Not Read**



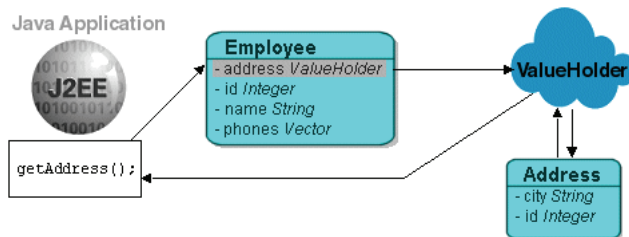
The first time the address is accessed, as in Figure 17-4, the ValueHolder reads and returns the Address object.

**Figure 17-4 Initial Request**



Subsequent requests for the address do not access the database, as shown in Figure 17-5.

**Figure 17-5 Subsequent Requests**



If you are using method access (Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"), the `get` and `set` methods specified in the mapping must access the instance of ValueHolderInterface, rather than the object referenced by the value holder. The application should not use these getter and setter, but use the getter and setter that hide the usage of value holders. For more

information, see [Section 121.3.2.2, "Configuring Value Holder Indirection with Method Accessing"](#).

For JPA entities or POJO classes that you configure for weaving, TopLink weaves value holder indirection for one-to-one mappings. If you want TopLink to weave change tracking and your application includes collection mappings (one-to-many or many-to-many), then you must configure all collection mappings to use transparent indirect container indirection only (you may not configure your collection mappings to use eager loading nor value holder indirection).

#### 17.2.4.2 Transparent Indirect Container Indirection

Transparent indirect container (see [Section 121.14, "Configuring Container Policy"](#)) indirection lets you declare any relationship attribute of a persistent class that holds a collection of related objects as any of the following:

- `java.util.Collection`
- `java.util.Hastable`
- `java.util.List`
- `java.util.Map`
- `java.util.Set`
- `java.util.Vector`

TopLink will use an indirection object that implements the appropriate interface and also performs just-in-time reading of the related objects. When using transparent indirection, you do not have to declare the attributes as `ValueHolderInterface`.

Newly created collection mappings use transparent indirection by default if their attribute *is not* a `ValueHolderInterface`.

For JPA entities or POJO classes that you configure for weaving, TopLink weaves value holder indirection for one-to-one mappings. If you want TopLink to weave change tracking and your application includes collection mappings (one-to-many or many-to-many), then you must configure all collection mappings to use transparent indirect container indirection only (you may not configure your collection mappings to use eager loading nor value holder indirection).

You can configure TopLink to automatically weave transparent indirect container indirection for JPA entities and Plain Old Java Object (POJO) classes. For more information, see the following:

- [Section 2.10, "Using Weaving"](#)
- "Using EclipseLink JPA Weaving" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#Using\\_EclipseLink\\_JPA\\_Weaving](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#Using_EclipseLink_JPA_Weaving)

#### 17.2.4.3 Proxy Indirection

Introduced in JDK 1.3, the Java class `Proxy` lets you use dynamic proxy objects as place-holders for a defined interface. Certain TopLink mappings (see [Table 121-4, "Mapping Support for Indirection"](#)) can be configured to use proxy indirection, which gives you the benefits of TopLink indirection without the need to include TopLink classes in your domain model. Proxy indirection is to one-to-one relationship mappings as indirect containers are to collection mappings.

To use proxy indirection, your domain model must satisfy all of the following criteria:

- The target class of the one-to-one relationship must implement a public interface.

- The one-to-one attribute on the source class must be of the `interface` type.
- If you employ method accessing (Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"), then the getter and setter methods must use the interface.

Before using proxy indirection, be aware of the restrictions it places on how you use the unit of work (see Section 17.2.4.3.1, "Proxy Indirection Restrictions").

To configure proxy indirection, you can use Oracle JDeveloper TopLink Editor, TopLink Workbench (see Section 121.3.1, "How to Configure Indirection Using TopLink Workbench"), or Java in an amendment method (see Section 121.3.2.5, "Configuring Proxy Indirection").

**17.2.4.3.1 Proxy Indirection Restrictions** Proxy objects in Java are only able to intercept messages sent. If a primitive operation such as `==`, `instanceof`, or `getClass` is used on a proxy, it will not be intercepted. This limitation can require the application to be somewhat aware of the usage of proxy objects.

You cannot register the target of a proxy indirection implementation with a unit of work. Instead, first register the source object with the unit of work. This lets you retrieve a target object clone with a call to a getter on the source object clone.

For example:

```
UnitOfWork uow = session.acquireUnitOfWork();
Employee emp = (Employee)session.readObject(Employee.class);

// Register the source object
Employee empClone = (Employee)uow.registerObject(emp);

// All of source object's relationships are cloned when source object is cloned
Address addressClone = empClone.getAddress();
addressClone.setCity("Toronto");
```

For more information about clones and the unit of work, see Chapter 113, "Introduction to TopLink Transactions".

#### 17.2.4.4 Weaved Indirection

For JPA entities or POJO classes that you configure for weaving, TopLink weaves value holder indirection for one-to-one mappings. If you want TopLink to weave change tracking and your application includes collection mappings (one-to-many or many-to-many), then you must configure all collection mappings to use transparent indirect container indirection only (you may not configure your collection mappings to use eager loading nor value holder indirection).

For more information, see Section 2.10, "Using Weaving".

#### 17.2.4.5 Indirection and JPA

When you set mapping annotation attribute `fetch` to `lazy`, the EclipseLink JPA persistence provider uses indirection.

By default, one-to-many and many-to-many relationships are lazy and use transparent indirection, while one-to-one and many-to-one relationships are not lazy.

If you set one-to-one or many-to-one relationships to lazy, and you enable weaving, the EclipseLink JPA persistence provider will use weaving to enable value holder indirection for these relationships.

For more information, see the following:

- Section 17.2.4.4, "Weaved Indirection"

- [Section 2.10, "Using Weaving"](#)

#### 17.2.4.6 Indirection and EJB 2.n CMP

When using indirection (lazy loading) with EJB 2.n, how TopLink handles indirection depends on the EJB version and application server you are using.

In addition, you cannot use proxy indirection (see [Section 17.2.4.3, "Proxy Indirection"](#)) for relationships to an enterprise bean, because EJB do not directly implement their remote or local interfaces.

When using indirection with an enterprise bean that your application serializes, you must consider the effect of any untriggered indirection objects at deserialization time (see [Section 17.2.4.7, "Indirection, Serialization, and Detachment"](#)).

When using CMP with any of the application servers for which TopLink provides CMP integration (see [Section 8.1, "Introduction to the Application Server Support"](#)), TopLink uses code generation to automatically configure the usage of value holder indirection ([Section 17.2.4.1, "Value Holder Indirection"](#)) for all CMR.

#### 17.2.4.7 Indirection, Serialization, and Detachment

When using indirection (lazy loading), it is likely that a graph of persistent objects will contain untriggered indirection objects. Because indirection objects are transient and do not survive serialization between one JVM and another, untriggered indirection objects will trigger an error if the relationship is accessed after deserialization.

The application must ensure that any indirect relationships that will be required after deserialization have been instantiated before serialization. This can be done through accessing the get method for any relationship using `ValueHolder` or weaved indirection, and by sending the `size` method to any relationship using transparent indirection. If the application desired the relationships to be always instantiated on serialization, you could overwrite the serialization `writeObject` method in the persistent class to first instantiate the desired relationships. Use caution for objects with many or deep relationships to avoid serializing large object graphs: ideally, only the relationships required by the client should be instantiated.

When serializing JPA entities, any lazy relationships that have not been instantiated prior to serialization will trigger errors if they are accessed. If weaving is used on the server, and the entities are serialized to a client, the same weaved classes must exist on the client, either through static weaving of the jar, or through launching the client JVM using the TopLink agent.

For more information, see the following:

- [Section 2.10, "Using Weaving"](#)
- [Section 115.5, "Merging Changes in Working Copy Clones"](#)

### 17.2.5 Method Accessors and Attribute Accessors

By default, TopLink uses direct access to access public attributes. Using TopLink, you can configure field access at the project level (see [Section 117.4, "Configuring Method or Direct Field Access at the Project Level"](#)) and at the mapping level ([Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"](#)).

### 17.2.6 Mapping Converters and Transformers

If existing TopLink mappings do not meet your needs, you can create custom mappings using mapping extensions. These extensions include the following:

- Serialized Object Converter
- Type Conversion Converter
- Object Type Converter
- Simple Type Translator
- Transformation Mappings

---

**Note:** You can use the mapping converters and transformers regardless of whether your data source is relational or nonrelational.

---

### 17.2.6.1 Serialized Object Converter

The serialized object converter is an extension of direct and direct collection mappings that lets you map complex objects into binary fields through Java object serialization. Serialized objects are normally stored in RAW or Binary Large Object (BLOB) fields in the database, or HEX or BASE64 elements in an XML document.

Figure 17–6 shows an example of a direct-to-field mappings that uses a serialized object converter. The attribute `jobDescription` contains a formatted text document that is stored in the `JOB_DESC` field of the database.

**Figure 17–6 Serialized Object Converter (relational)**

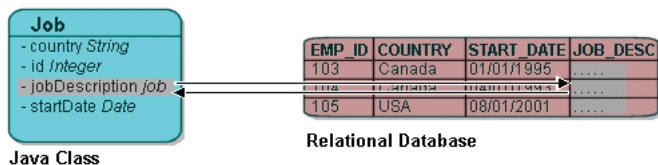
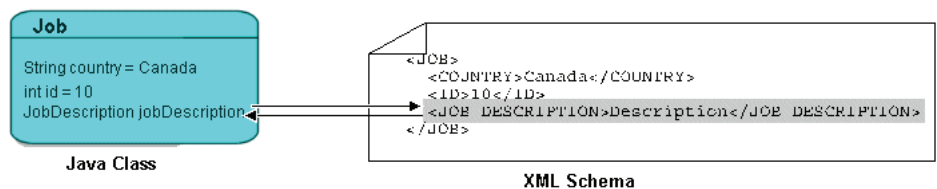


Figure 17–8 demonstrates an example of a nonrelational mapping that uses a serialized object converter. The attribute `jobDescription` contains a formatted text document that TopLink stores in the `JOB DESCRIPTION` element of an XML schema.

**Figure 17–7 Serialized Object Converter (nonrelational)**



The serialized object converter relies on the Java serializer. Before you map a domain object with the serialized object converter, ensure that the domain object implements the `java.io.Serializable` interface (or inherits that implementation) and marks all nonserializable fields transient.

For more information, see [Section 121.9, "Configuring a Serialized Object Converter"](#).

### 17.2.6.2 Type Conversion Converter

The type conversion converter is an extension of direct and direct collection mappings that lets you explicitly map a data source type to a Java type. For example, a `Number`



in the data source can be mapped to a `String` in Java, or a `java.util.Date` in Java can be mapped to a `java.sql.Date` in the data source.

Figure 17–8 illustrates a type conversion mapping (relational). Because the `java.util.Date` class is stored by default as a `Timestamp` in the database, it must first be converted to an explicit database type such as `java.sql.Date` (required only for DB2—most other databases have a single date data type that can store any date or time).

**Figure 17–8 Type Conversion Mapping (relational)**

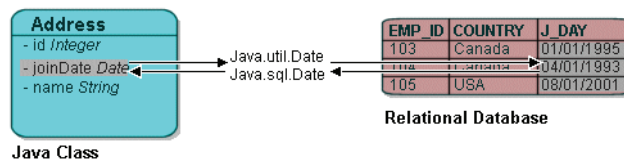
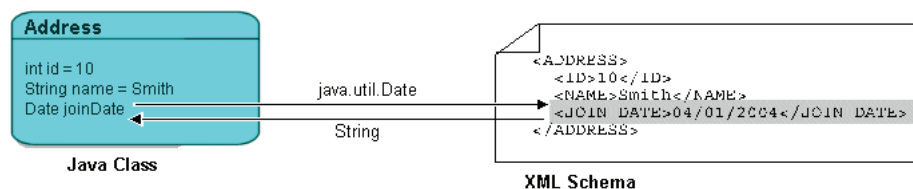


Figure 17–9 illustrates a type conversion mapping (nonrelational). `java.util.Date` object is mapped to a `String` in a XML schema.

**Figure 17–9 Type Conversion Mapping (nonrelational)**



You can use a type conversion converter to specify the specific database type when that type must be handled specially for the database. This includes support for the special Oracle JDBC binding options required for `NCHAR`, `NVARCHAR2`, and `NCLOB` fields as well as the special Oracle Thin JDBC insert and update requirements for handling `BLOB` and `CLOB` fields greater than 5K.

TopLink uses the `NCharacter`, `NClob` and `NString` types in the `oracle.toplink.platform.database.oracle` package as the converter data type to support the `NCHAR`, `NCLOB` and `NVARCHAR2` types. TopLink uses the `java.sql.Blob` and `Clob` types as the converter data type to support `BLOB` and `CLOB` values greater than 5K.

You can configure a type conversion converter to map a data source time type (such as `TIMESTAMP`) to a `java.lang.String` provided that the `String` value conforms to the following formats:

- `YYYY/MM/DD HH:MM:SS`
- `YY/MM/DD HH:MM:SS`
- `YYYY-MM-DD HH:MM:SS`
- `YY-MM-DD HH:MM:SS`

For more complex `String` to `TIMESTAMP` type conversion, consider a transformation mapping (see Section 17.2.6.5, "Transformation Mappings").

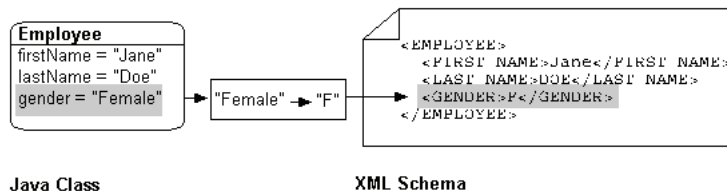
For more information, see Section 121.10, "Configuring a Type Conversion Converter".

### 17.2.6.3 Object Type Converter

The object type converter is an extension of direct and direct collection mappings that lets you match a fixed number of XML values to Java objects. Use this converter when the values in the schema differ from those in Java.

Figure 17–10 illustrates an object type conversion between the `Employee` attribute `gender` and the XML element `gender`. If the value of the Java object attribute is `Female`, TopLink stores it in the XML element as `F`.

**Figure 17–10 Object Type XML Converter**



For more information, see [Section 121.11, "Configuring an Object Type Converter"](#).

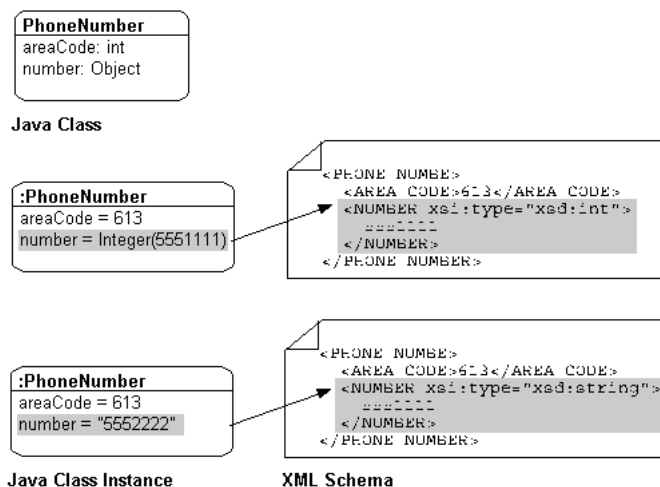
### 17.2.6.4 Simple Type Translator

The simple type translator is an extension of direct and direct collection mappings that lets you automatically translate an XML element value to an appropriate Java type based on the element's `<type>` attribute as defined in your XML schema.

You can use a simple type translator only when the mapping's XPath goes to a text node. You cannot use a simple type translator if the mapping's XPath goes to an attribute.

Using a simple type translator, you can make the XML document preserve type information. This is useful when your object model specifies generic object attributes such as `java.lang.Object` and `java.io.Serializable`, since they do not trigger specific type conversions in TopLink as do specific object attributes such as `java.lang.Integer` or `java.util.Calendar`.

Figure 17–11 illustrates a type translation XML mapping for the number attribute of the `PhoneNumber` class. Notice that the Java attribute is not specific enough to preserve the typing. The simple type translator adds the type information to the resulting document to preserve the typing.

**Figure 17–11 Simple Type Translator**

By default, TopLink uses built-in read and write conversion pairs (see [Section 17.2.6.4.1, "Default Read Conversions"](#) and [Section 17.2.6.4.2, "Default Write Conversions"](#)).

You can override this behavior by specifying and configuring your own simple type translator, for example, to write XML binary data as Base64.

For more information, see [Section 121.12, "Configuring a Simple Type Translator"](#).

**17.2.6.4.1 Default Read Conversions** [Table 17–2](#) lists the built-in conversion pairs for reading XML elements. When the schema `<type>` attribute is specified and the simple type translator is enabled, the value read is converted to the corresponding Java type.

**Table 17–2 Simple Type Translator Read Conversions**

Schema Type	Java Type
base64Binary	Byte[]
boolean	Boolean
byte	Byte
date	Calendar
dateTime	Calendar
double	Double
float	Float
hexBinary	Byte[]
int	int
integer	BigInteger
long	Long
short	Short
string	String
time	Calendar
unsignedByte	Short

**Table 17–2 (Cont.) Simple Type Translator Read Conversions**

Schema Type	Java Type
unsignedInt	Long
unsignedShort	Integer

**17.2.6.4.2 Default Write Conversions** Table 17–3 lists the built-in conversion pairs for writing XML. When a Java class attribute is of a type in Table 17–3 and the simple type translator is enabled, the corresponding schema type is specified on the element written.

**Table 17–3 Simple Type Translator Write Conversions**

Java Type	Schema Type
Byte[]	hexBinary
BigInteger	integer
Boolean	boolean
Byte	byte
Calendar	dateTime
GregorianCalendar	dateTime
Double	double
Float	float
Integer	int
Long	long
int	int
short	short
String	string

### 17.2.6.5 Transformation Mappings

In some special circumstances, existing mapping types and their default Java to data source type handling may be insufficient. In these special cases, you can consider using a transformation mapping to perform specialized translations between how a value is represented in Java and in the data source.

A transformation mapping is made up of the following two components:

- attribute transformer (see [Section 121.15, "Configuring Attribute Transformer"](#)): performs the object attribute transformation at read (unmarshal) time;
- field transformer (see [Section 121.16, "Configuring Field Transformer Associations"](#)): performs the object attribute-to-field transformation at write (marshal) time;

You can implement a transformer as either a separate class or as a method on your domain object.

Within your implementation of the attribute and field transformer, you can take whatever actions are necessary to transform your application data to suit your data source, and vice versa.

For more information, see the following:

- Section 27.13, "Transformation Mapping"
- Section 77.9, "EIS Transformation Mapping"
- Section 53.9, "XML Transformation Mapping"

## 17.2.7 Mappings and XPath

TopLink uses XPath statements to efficiently map the attributes of a Java object in EIS mappings to XML records, and in XML mappings to XML documents. When you create such a mapping, you can specify the following:

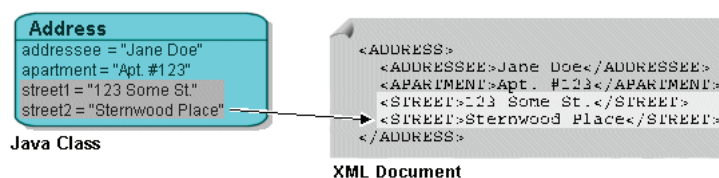
- XPath by Position
- XPath by Path and Name
- XPath by Name
- Self XPath

### 17.2.7.1 XPath by Position

In a relational database table, columns are uniquely identified by name. In an XML document, elements are uniquely identified by name and position. [Figure 17–12](#) illustrates mapping to an XML document in which the first instance of the `street` element stores apartment information and the second instance of the `street` element stores street information. [Figure 17–12](#) shows that TopLink XML mappings preserve the order in which mappings are persisted and allow you to map Java object attributes to XML elements by position using an XPath like `street[2]/text()`.

Other XML technologies only recognize the name of XML elements (not their position) and force you to store the simple values from elements with the same name in a collection.

**Figure 17–12 Mapping to an XML Document by Position**

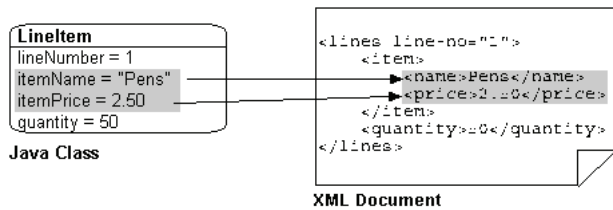


### 17.2.7.2 XPath by Path and Name

In an XML document, attributes and elements are uniquely identified by a combination of name and path. [Figure 17–13](#) illustrates that TopLink XML mappings can uniquely identify an XML element by name and path using an XPath such as `item/name/text()`. TopLink does not require a formal object relationship between XML elements `lines` and `item`.

Other XML technologies force you to provide an object relationship for every level of nesting, resulting in the inclusion of many XML elements and classes simply to organize the data to satisfy this restriction. This produces an unnecessarily large object model that does not properly reflect the domain space.

**Figure 17-13 Mapping to an XML Document by Path and Name**

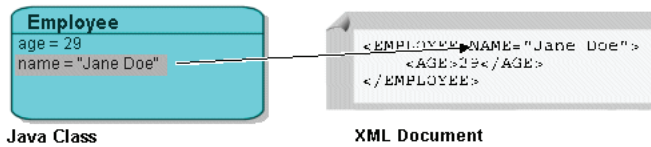


### 17.2.7.3 XPath by Name

For simple XML documents, TopLink XML mappings can correctly place data in an XML document given an XPath of only an attribute or element name.

Figure 17-14 illustrates mapping to a simple XML document by name. You can map Java object attribute name to XML attribute name by specifying an XPath of only @NAME. Similarly, you can map Java object attribute age to XML text node AGE by specifying an XPath of only AGE.

**Figure 17-14 Mapping to a Simple XML Document by Name**



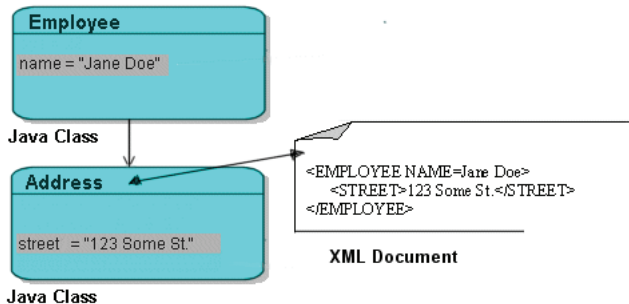
Specifying an XPath by name provides the worst performance of the XPath mapping options. Oracle recommends that you use XPath by position (see Section 17.2.7.1, "XPath by Position") or XPath by path and name (see Section 17.2.7.2, "XPath by Path and Name") instead.

### 17.2.7.4 Self XPath

For composite relationships, TopLink XML mappings can place data in the parent's element rather than an element nested within it given the self XPath (" . ").

Figure 17-15 illustrates mapping to an XML document using the self XPath.

**Figure 17-15 Mapping to a XML Document Using Self XPath**



Note that in the preceding example represented by Figure 17-15, name attribute of the Employee class is mapped using the @name annotation.

Using the self XPath, you can make TopLink perform all read and write operations in the parent's element and not an element nested within it (see [Section 17.2.9, "Mappings and the JAXB: class Customization"](#)).

## 17.2.8 Mappings and xsd:list and xsd:union Types

TopLink supports mapping to `xsd:list` and `xsd:union` types in EIS mappings to XML records and XML mappings to XML documents, as [Table 17-4](#) shows.

**Table 17-4 TopLink Support for xsd:list and xsd:union Types**

XSD	EIS Direct Mapping XML Direct Mapping	EIS Composite Direct Collection Mapping XML Composite Direct Collection Mapping
Mapping an xsd:union Type	✓	
Mapping an xsd:list Type	✓	✓
Mapping a List of Unions		✓
Mapping a Union of Lists	✓	✓
Mapping a Union of Unions	✓	

### 17.2.8.1 Mapping an xsd:union Type

Use an `EISDirectMapping` (with XML records), an `XMLDirectMapping` or their subclasses to map a Java attribute to an `xsd:union` type, such as the following:

```
<xsd:simpleType name="size-type">
  <xsd:union memberTypes="xsd:decimal xsd:string"/>
</xsd:simpleType>
```

When TopLink marshalls (writes) an object to XML, it uses its default conversion pairs to convert from the Java type to the appropriate `xsd` type.

In the case where the `memberTypes` map to the same Java type, TopLink marshalls using the first `memberType` in the union which allows a successful conversion. For example, if you map a Java type of `byte []` to an `xsd:union` with `memberTypes` of `hexBinary` and `base64Binary`, then TopLink marshalls using the first `memberType`: `hexBinary`.

You can customize the default conversion pairs to control the Java type to `xsd` type conversion using `XMLField` method `addConversion` and configuring your mapping with that `XMLField` using `EISDirectMapping` or `XMLDirectMapping` method `setField`. For example, if the `memberTypes` were `xsd:date` and `xsd:time` and the Java attribute was of type `java.util.Date` instead of the JAXB 1.0 standard `java.util.Calendar`, you can modify the conversion pair for `xsd:date` to be `java.util.Date`.

When TopLink unmarshalls (reads) XML into an object, it tries each `memberType` in the order specified in the XSD until the first successful conversion is made.

If your XML document specifies the `xsi:type` attribute on an element, then TopLink converts according to the `xsi:type` instead of trying the `memberTypes`.

For more information, see [Section 53.3.5, "Mapping to a Union Field with an XML Direct Mapping"](#). The same applies to an `EISDirectMapping` with XML records (see [Section 77.3, "EIS Direct Mapping"](#)).

### 17.2.8.2 Mapping an xsd:list Type

You can map a Java attribute to an `xsd:list` type, such as:

```
<xsd:simpleType name="sizes">
  <xsd:list itemType="xsd:int"/>
</xsd:simpleType>
```

If you represent the `xsd:list` in your object model as a Java `List` type, use an `EISCompositeDirectCollectionMapping` (with XML records), an `XMLCompositeDirectCollectionMapping` or their subclasses and use mapping method `useCollectionClass` to specify the `List` type of the Java attribute.

If you represent the list in your object model as a `String` of white space delimited tokens (for example, "aaa bbb ccc"), use an `EISDirectMapping` (with XML records), an `XMLDirectMapping` or their subclasses to map this Java attribute to an `xsd:list` (for example, `<item>aaa bbb ccc</item>`).

In either case, you can configure whether or not the mapping unmarshalls (writes) the list to a single node, like `<item>aaa bbb ccc</item>`, or to multiple nodes, such as the following:

```
<item>aaa</item>
<item>bbb</item>
<item>ccc</item>
```

For more information on mapping to an `xsd:list` type using an `XMLCompositeDirectCollectionMapping` or its subclasses, see the following:

- [Section 53.4.3, "Mapping to a Single Text Node with an XML Composite Direct Collection Mapping"](#)
- [Section 53.4.4, "Mapping to a Single Attribute with an XML Composite Direct Collection Mapping"](#)
- [Section 53.4.7, "Specifying the Content Type of a Collection with an XML Composite Direct Collection Mapping"](#)

The same applies to an `EISCompositeDirectCollectionMapping` (with XML records).

For more information about mapping to an `xsd:list` type using an `XMLDirectMapping` or its subclasses, see [Section 53.3.4, "Mapping to a List Field with an XML Direct Mapping"](#). The same applies to an `EISDirectMapping` with XML records (see [Section 77.3, "EIS Direct Mapping"](#)).

### 17.2.8.3 Mapping a List of Unions

Use an `EISCompositeDirectCollectionMapping` (with XML records), an `XMLCompositeDirectCollectionMapping` or their subclasses to map a Java attribute to an `xsd:list` that contains `xsd:union` types, such as:

```
<xsd:element name="listOfUnions" type="listOfUnions"/>
  <xsd:simpleType name="listOfUnions">
    <xsd:list>
      <xsd:simpleType>
        <xsd:union memberTypes="xsd:date xsd:integer"/>
      </xsd:simpleType>
    </xsd:list>
  </xsd:simpleType>
```

When `TopLink` marshalls (writes) an object to XML, it does not rely on a single `xsd:list itemType`. Instead, for each item in the list, `TopLink` tries each `memberType` until the first successful conversion.

For more information, see [Section 53.4.5, "Mapping to a List of Unions with an XML Composite Direct Collection Mapping"](#). The same applies to an `EISCompositeDirectCollectionMapping` with XML records (see [Section 77.4, "EIS Composite Direct Collection Mapping"](#)).



### 17.2.8.4 Mapping a Union of Lists

You can map a Java attribute to an `xsd:union` type whose `memberTypes` are `xsd:list` types where each `xsd:list` contains items of a single type, such as:

```
<xsd:element name="listOfUnions" type="UnionOfLists"/>
  <xsd:simpleType name="UnionOfLists">
    <xsd:union memberTypes="xsd:double">
      <xsd:simpleType>
        <xsd:list itemType="xsd:date"/>
      </xsd:simpleType>
    <xsd:simpleType>
      <xsd:list itemType="xsd:integer"/>
    </xsd:simpleType>
  </xsd:union>
</xsd:simpleType>
```

Note that in this example, valid XML documents contain either all `xsd:double`, all `xsd:date`, or all `xsd:integer` values.

If you represent the list in your object model as a `String` of white space delimited tokens (for example, "aaa bbb ccc"), use an `EISDirectMapping` (with XML records) or an `XMLDirectMapping` to map this Java attribute to an `xsd:list` (for example, `<item>aaa bbb ccc</item>`).

If you represent the list in your object model as a Java `List` type, use an `EISCompositeDirectCollectionMapping` (with XML records), an `XMLCompositeDirectCollectionMapping` or their subclasses.

For more information, see the following:

- [Section 53.3.6, "Mapping to a Union of Lists with an XML Direct Mapping"](#). The same applies to an `EISDirectMapping` with XML records (see [Section 77.3, "EIS Direct Mapping"](#)).
- [Section 53.4.6, "Mapping to a Union of Lists with an XML Composite Direct Collection Mapping"](#). The same applies to an `EISCompositeDirectCollectionMapping` with XML records (see [Section 77.4, "EIS Composite Direct Collection Mapping"](#)).

### 17.2.8.5 Mapping a Union of Unions

Use an `EISDirectMapping` (with XML records), an `XMLDirectMapping` or their subclasses to map a Java attribute to an `xsd:union` that contains `xsd:union` types, such as the following:

```
<xsd:simpleType name="UnionOfUnions">
  <xsd:union>
    <xsd:simpleType>
      <xsd:union>
        <xsd:simpleType>
          <xsd:list itemType="xsd:date"/>
        </xsd:simpleType>
      <xsd:simpleType>
        <xsd:list itemType="xsd:integer"/>
      </xsd:simpleType>
    </xsd:union>
  </xsd:simpleType>
  <xsd:simpleType>
    <xsd:union>
      <xsd:simpleType>
        <xsd:list itemType="xsd:string"/>
      </xsd:simpleType>
      <xsd:simpleType>
        <xsd:list itemType="xsd:float"/>
      </xsd:simpleType>
    </xsd:union>
  </xsd:simpleType>
</xsd:simpleType>
```

```

        </xsd:union>
    </xsd:simpleType>
</xsd:union>
</xsd:simpleType>

```

Note that in this example, valid XML documents may contain any of `xsd:date`, `xsd:integer`, `xsd:string`, or `xsd:float`.

For more information, see [Section 53.3.7, "Mapping to a Union of Unions with an XML Direct Mapping"](#). The same applies to an `EISDirectMapping` with XML records (see [Section 77.3, "EIS Direct Mapping"](#)).

## 17.2.9 Mappings and the `jaxb:class` Customization

Using the `jaxb:class` customization, you can declaratively specify an application-specific subclass of a schema-derived implementation class. This lets you write your own classes that extend JAXB's generated implementation classes (see [Section 47.1.1.2.1, "Implementation Classes"](#)). The JAXB runtime binding framework can then access your subclasses.

When you create an EIS composite object mapping to XML records, or an XML composite object mapping to XML documents, you can configure the mapping's XPath ([Section 121.4, "Configuring XPath"](#)) to accommodate `jaxb:class` customizations with the following XSD structures:

- [all, choice, or sequence Structure](#)
- [group Structure](#)
- [sequence or choice Structure Containing a group](#)
- [group Structure Containing a sequence or choice](#)
- [group Structure Containing a group](#)

When mapping to `jaxb:class` customized structures, consider the limitations of TopLink support for this customization (see [Section 17.2.9.6, "Limitations of `jaxb:class` Customization Support"](#)).

### 17.2.9.1 all, choice, or sequence Structure

You can use the `jaxb:class` customization with an `all`, `choice`, or `sequence` structure. [Example 17-1](#) shows a `jaxb:class` customization of an `all` structure.

#### **Example 17-1** *jaxb:class Customization of an all Structure*

```

<xsd:element name="employee">
  <xsd:complexType>
    <xsd:all>
      <xsd:annotation>
        <xsd:appinfo>
          <jaxb:class name="period"/>
        </xsd:appinfo>
      </xsd:annotation>
      <xsd:element name="startDate" type="xsd:date"/>
      <xsd:element name="endDate" type="xsd:date"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>

```

This directs the JAXB compiler to create an inner class named `Period` in the owning element's class for the `all` structure. Use an `EISCompositeObjectMapping` (with XML records) or an `XMLCompositeObjectMapping` to map a Java attribute to this inner class.

For more information, see [Section 53.5, "XML Composite Object Mapping"](#). The same applies to an `EISCompositeObjectMapping` with XML records (see [Section 77.5, "EIS Composite Object Mapping"](#)).

### 17.2.9.2 group Structure

You can use the `jaxb:class` customization with a group structure, as [Example 17-2](#) shows.

#### **Example 17-2** *jaxb:class Customization of a group Structure*

```
<xsd:group name="G1">
  <xsd:annotation>
    <xsd:appinfo>
      <jaxb:class name="period" />
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="startDate" type="xsd:date" />
    <xsd:element name="endDate" type="xsd:date" />
  </xsd:sequence>
</xsd:group>

<xsd:element name="employee">
  <xsd:complexType>
    <xsd:group ref="G1" />
  </xsd:complexType>
</xsd:element>
```

This directs the JAXB compiler to create an external wrapper class named `Period` for the group structure. Use an `EISCompositeObjectMapping` (with XML records) or an `XMLCompositeObjectMapping` to map a Java attribute to this external wrapper class.

For more information, see [Section 53.5, "XML Composite Object Mapping"](#). The same applies to an `EISCompositeObjectMapping` with XML records (see [Section 77.5, "EIS Composite Object Mapping"](#)).

### 17.2.9.3 sequence or choice Structure Containing a group

You can use the `jaxb:class` customization with a sequence or choice structure that contains a group. [Example 17-3](#) shows a `jaxb:class` customization of a sequence structure containing a group structure.

#### **Example 17-3** *jaxb:class Customization of a sequence Structure Containing a group*

```
<xsd:element name="employee">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:annotation>
        <xsd:appinfo>
          <jaxb:class name="EmploymentInfo" />
        </xsd:appinfo>
      </xsd:annotation>
      <xsd:element name="id" type="xsd:int" />
      <xsd:group ref="G1" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:group name="G1">
  <xsd:annotation>
    <xsd:appinfo>
      <jaxb:class name="Period" />
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="startDate" type="xsd:date" />
    <xsd:element name="endDate" type="xsd:date" />
  </xsd:sequence>
</xsd:group>
```

```

        </xsd:appinfo>
    </xsd:annotation>
    <xsd:sequence>
        <xsd:element name="startDate" type="xsd:date" />
        <xsd:element name="endDate" type="xsd:date" />
    </xsd:sequence>
</xsd:group>

```

This directs the JAXB compiler to create an inner class named `EmploymentInfo` in the owning element's class for the `sequence` structure and an external wrapper class named `Period` for the `group` structure. The inner class references the external wrapper class. Use an `EISCompositeObjectMapping` (with XML records) or an `XMLCompositeObjectMapping` to map a Java attribute to this inner class.

For more information, see [Section 53.5, "XML Composite Object Mapping"](#). The same applies to an `EISCompositeObjectMapping` with XML records (see [Section 77.5, "EIS Composite Object Mapping"](#)).

#### 17.2.9.4 group Structure Containing a sequence or choice

You can use the `jaxb:class` customization with a `group` structure that contains a `sequence` or `choice`. [Example 17-4](#) shows a `jaxb:class` customization of a `group` structure containing a `sequence` structure.

##### **Example 17-4** *jaxb:class Customization of a group Structure Containing a sequence*

```

<xsd:group name="G1">
    <xsd:annotation>
        <xsd:appinfo>
            <jaxb:class name="EmploymentInfo" />
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:sequence>
        <xsd:annotation>
            <xsd:appinfo>
                <jaxb:class name="Period" />
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:element name="startDate" type="xsd:date" />
        <xsd:element name="endDate" type="xsd:date" />
    </xsd:sequence>
</xsd:group>

<xsd:element name="employee">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="id" type="xsd:int" />
            <xsd:group ref="G1" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

```

This directs the JAXB compiler to create an external wrapper class named `EmploymentInfo` for the `group` structure and an inner class named `Period` in the external wrapper class for the `sequence` structure. The owning element references the external wrapper class. Use an `EISCompositeObjectMapping` (with XML records) or an `XMLCompositeObjectMapping` to map a Java attribute to this external wrapper class.

For more information, see [Section 53.5, "XML Composite Object Mapping"](#). The same applies to an `EISCompositeObjectMapping` with XML records (see [Section 77.5, "EIS Composite Object Mapping"](#)).

### 17.2.9.5 group Structure Containing a group

You can use the `jaxb:class` customization with a `group` structure that contains another `group` structure, as [Example 17-5](#) shows.

#### **Example 17-5** *jaxb:class Customization of a group Structure Containing a group*

```
<xsd:group name="G1">
  <xsd:annotation>
    <xsd:appinfo>
      <jaxb:class name="EmploymentInfo"/>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="id" type="xsd:int"/>
    <xsd:group ref="G2"/>
  </xsd:sequence>
</xsd:group>

<xsd:group name="G2">
  <xsd:annotation>
    <xsd:appinfo>
      <jaxb:class name="Period"/>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="startDate" type="xsd:date"/>
    <xsd:element name="endDate" type="xsd:date"/>
  </xsd:sequence>
</xsd:group>

<xsd:element name="employee">
  <xsd:complexType>
    <xsd:group ref="G1"/>
  </xsd:complexType>
</xsd:element>
```

This directs the JAXB compiler to create a wrapper class named `EmploymentInfo` for the `group` structure that the owning element's class references and another wrapper class named `Period` for the `group` structure that the `EmploymentInfo` class references. Use an `EISCompositeObjectMapping` (with XML records) or an `XMLCompositeObjectMapping` to map a Java attribute to these wrapper classes.

For more information, see [Section 53.5, "XML Composite Object Mapping"](#). The same applies to an `EISCompositeObjectMapping` with XML records (see [Section 77.5, "EIS Composite Object Mapping"](#)).

### 17.2.9.6 Limitations of jaxb:class Customization Support

When mapping to `jaxb:class` customized structures, consider the following limitations:

- Unbounded structures are not supported.
- Partial validation is not supported.
- When mapping sequence elements to a composite object, the XML schema must order the elements so that the elements you map to the composite object are kept together.

The sequence structure forces all elements to occur in the order in which they are specified in the XML schema. Consider the XML schema shown in [Example 17-6](#). A valid XML instance must contain the sequence elements in the specified order:

```
street, customerName, city
```

In this example, you want to map the `customerName` attribute with a direct mapping and you want to map the `street` and `city` attributes to a composite `Address` object. Depending on the order in which you define the mappings, `TopLink` will marshal invalid XML document instances in the order

`customerName, street, city`

or

`street, city, customerName.`

**Example 17–6 XML Schema With Unsupported Sequence Element Order**

```
<xs:element name="customer">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="street" type="xs:string"/>
      <xs:element name="customerName" type="xs:string" />
      <xs:element name="city" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

To correct this problem, modify the XML schema to keep the elements you want to map to the composite object together (see [Example 17–7](#)) and define the mappings in the order specified by the XML schema.

**Example 17–7 XML Schema With Supported Sequence Element Order**

```
<xs:element name="customer">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="customerName" type="xs:string"/>
      <xs:element name="street" type="xs:string"/>
      <xs:element name="city" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

## 17.2.10 Mappings and JAXB Typesafe Enumerations

JAXB binds a typesafe enumeration class to a named simple type definition with a `baseType` that derives from `xsd:NCName` and has enumeration facets (see [Example 17–8](#)).

**Example 17–8 Schema Fragment with Typesafe Enumeration Declaration**

```
<simpleType name="NISTSchema-NCName-enumeration-1-Type">
  <restriction base="NCName">
    <enumeration value="qbandwidth-and.software-use.too"/>
    <enumeration value="_effort-disseminate_and-devices.com"/>
  </restriction>
</simpleType>
```

You can map a Java attribute to such an enumeration using the `JAXBTypesafeEnumConverter` with an `EISDirectMapping` or `EISCompositeDirectCollectionMapping` with XML records, or with an `XMLDirectMapping`, `XMLCompositeDirectCollectionMapping` or their subclasses with XML documents.

Oracle JDeveloper `TopLink` Editor and `TopLink` Workbench do not support the `JAXBTypesafeEnumConverter` directly: to configure a mapping with this converter,

you must use a descriptor amendment method (see [Section 121.13, "Configuring a JAXB Typesafe Enumeration Converter"](#)).

If you create a project and object model using the TopLink JAXB compiler (see [Section 48.2, "Creating an XML Project from an XML Schema"](#)), the compiler will create the type safe enumeration class and a class with descriptor amendment methods and register the required amendment methods automatically.

## 17.3 Mapping API

All the mapping classes are derived from the `DatabaseMapping` class.

**Table 17-5 Platform and Mapping Package Compatibility**

Platform	Mapping Package
DatabasePlatform	<code>oracle.toplink.mappings</code>
For relational projects	<code>oracle.toplink.xdb</code> <code>oracle.toplink.objectrelational</code>
EISPlatform	<code>oracle.toplink.eis.mappings</code>
For EIS projects	
XMLPlatform	<code>oracle.toplink.ox.mappings</code>
For XML projects	

## 17.4 Relational Mappings

A relational mapping transforms any object data member type to a corresponding relational database (SQL) data source representation in any supported relational database. Relational mappings allow you to map an object model into a relational data-model.

Relational mappings can also transform object data members that reference other domain objects that are stored in other tables in the database and are related through foreign keys.

Use relational mappings in relational projects. For more information, see [Section 18.1, "Building Relational Projects"](#).

For more information about relational mappings, see [Part XII, "Relational Mappings"](#)

## 17.5 Object-Relational Data Type Mappings

An object-relational data type mapping transforms certain object data member types to structured data source representations optimized for storage in specialized object-relational data type databases such as Oracle Database. Object-relational data type mappings allow you to map an object model into an object-relational data type data-model.

Use object-relational data type mappings in relational projects. For more information, see [Section 18.1, "Building Relational Projects"](#).

For more information about object-relational data type mappings, see [Part XIII, "Object-Relational Data Type Mappings"](#).

## 17.6 XML Mappings

An XML mapping transforms object data members to the XML elements of an XML file whose structure is defined by an XML schema document (XSD).

Use XML mappings in XML projects. For more information, see [Section 47.1, "XML Project Concepts"](#).

For more information about XML mappings, see [Part XVI, "XML Mappings"](#).

## 17.7 EIS Mappings

An EIS mapping transforms object data members to the EIS record format defined by the object's descriptor.

Use EIS mappings in EIS projects. For more information, see [Section 71.1, "EIS Project Concepts"](#).

For more information about EIS mappings, see [Part XIX, "EIS Mappings"](#).



# Part IX

---

## Relational Projects

This part describes relational projects.

This part contains the following chapters:

- [Chapter 18, "Introduction to Relational Projects"](#)  
This chapter provides an overview of relational projects.
- [Chapter 19, "Creating a Relational Project"](#)  
This chapter explains how to create a relational project.
- [Chapter 20, "Configuring a Relational Project"](#)  
This chapter explains how to configure project options specific to relational project.



---

---

## Introduction to Relational Projects

This chapter provides an overview of relational projects and focuses on building these projects for relational and object-relational data type databases.

This chapter includes the following sections:

- [Building Relational Projects](#)
- [Sequencing in Relational Projects](#)

For information on project concepts and features common to more than one type of TopLink projects, see [Chapter 15, "Introduction to Projects"](#).

### 18.1 Building Relational Projects

Use a relational project for transactional persistence of Java objects to a conventional *relational* database or to an *object-relational data type* database that supports data types specialized for object storage, both accessed using JDBC.

---

---

**Note:** If you are using TopLink Workbench, you must add your JDBC driver to the TopLink Workbench classpath. If you are using TopLink Workbench and `direct-to-XMLType` mappings (see [Section 27.4, "Direct-to-XMLType Mapping"](#)), you must add the Oracle Database `xdb.jar` file to the TopLink Workbench classpath.

For more information, see [Section 5.2, "Configuring the TopLink Workbench Environment"](#).

---

---

In a relational project, you can make full use of TopLink queries and expressions (see [Part XXIV, "Queries"](#)).

#### 18.1.1 How to Build Relational Projects for a Relational Database

Oracle JDeveloper TopLink Editor and TopLink Workbench provide complete support for creating relational projects that map Java objects to a conventional relational database accessed using JDBC.

[Table 18–1](#) describes the components of a relational project for a relational database.

**Table 18–1 Components of a Relational Project for a Relational Database**

Component	Supported Types
Data Source	For more information, see the following: <ul style="list-style-type: none"> <li>■ <a href="#">Section 96.1.2.1, "DatabaseLogin"</a></li> <li>■ <a href="#">Section 96.1.3.1, "Database Platforms"</a></li> </ul>
Descriptors	For more information, see <a href="#">Section 21.1, "Relational Descriptors"</a> .
Mappings	For more information, see the following: <ul style="list-style-type: none"> <li>■ <a href="#">Part VIII, "Mappings"</a></li> <li>■ <a href="#">Part XII, "Relational Mappings"</a></li> </ul>

For more information, see [Chapter 19, "Creating a Relational Project"](#).

### 18.1.2 How to Build Relational Projects for an Object-Relational Data Type Database

Oracle JDeveloper TopLink Editor and TopLink Workbench do not currently support relational projects for an object-relational data type database. You must create such a relational project in Java.

Using Java, you can create a relational project for transactional persistence of Java objects to an object-relational data type database that supports data types specialized for object storage (such as Oracle Database) accessed using JDBC.

When using TopLink to build a relational project for an object-relational data type database, consider the following:

- You must create a Java class and a TopLink `ObjectRelationalDescriptor` for each structured type (`Struct/object-type`).
- TopLink supports only arrays (`Varrays`) of basic types or arrays on structured types (`Struct/object-type`).  
TopLink does not support arrays of `Refs` or arrays of nested tables.
- TopLink supports only nested tables of `Refs`.  
TopLink does not support nested tables of basic types, structured types, or array types.

The general development process for building a relational project for an object-relational data type database is as follows:

1. Define structured object-types in the database.
2. Define tables of the structured object-types in the database.
3. Define the Java classes that will map to the structured object-types.
4. Create a relational project (see [Chapter 116, "Creating a Project"](#)).
5. Create an object-relational data type descriptor for each Java class (see [Section 25.2, "Creating an Object-Relational Data Type Descriptor"](#)).
6. Create object-relational data type mappings from each persistent field of each Java class to the corresponding object-types and object-type tables.

For more information, see the following:

- [Chapter 120, "Creating a Mapping"](#)
- [Chapter 41, "Configuring an Object-Relational Data Type Mapping"](#)

Table 18–2 describes the components of a relational project for an object-relational data type database.

**Table 18–2 Components of a Relational Project for an Object-Relational Data Type Database**

Component	Supported Types
Data Source	For more information, see the following: <ul style="list-style-type: none"> <li>Section 96.1.2.1, "DatabaseLogin"</li> <li>Section 96.1.3.1, "Database Platforms"</li> </ul>
Descriptors	For more information, see Section 24.1, "Object-Relational Data Type Descriptors".
Mappings	For more information, see the following: <ul style="list-style-type: none"> <li>Part VIII, "Mappings"</li> <li>Part XIII, "Object-Relational Data Type Mappings"</li> </ul>

For more information, see Chapter 19, "Creating a Relational Project".

## 18.2 Sequencing in Relational Projects

In an relational project, you store persistent objects for your application in database tables that represent the class of instantiated object. As Figure 18–1 shows, each row of the VEHICLE\_POOL table represents an instantiated object from that class, and the VEH\_ID column holds the primary key for each object.

**Figure 18–1 Sequencing Elements in a Class Database Table**

Table Name → VEHICLE\_POOL

Sequencing Column →

VEH_ID	COLOR	MAKE	MODEL	YEAR
1	red	Chevy	Malibu	2000
2	white	Ford	Focus	2001
3	white	Hyundai	Accent	2000
4	yellow	Dodge	3500 Tow Truck	1998
5	blue	Pontiac	Bonneville	2003
6	green	BMW	325i	2002

You configure TopLink sequencing at the project or session level (see Section 20.3, "Configuring Sequencing at the Project Level" or Section 98.4, "Configuring Sequencing at the Session Level") to tell TopLink how to obtain values for the primary key column: that is, what type of sequencing to use (see Section 18.2.2, "Sequencing Types").

You configure TopLink sequencing at the descriptor level (see Section 23.3, "Configuring Sequencing at the Descriptor Level") to tell TopLink into which table and column to write the sequence value when an instance of a descriptor's reference class is created.

---

**Note:** When choosing a column type for a primary key value, ensure that the type provides a suitable precision. For example, if you use a `TIMESTAMP` type but your database platform's `TIMESTAMP` is defined only to the second, then identical values may be returned for objects created within the same second.

---

This section describes the following:

- [Sequencing Configuration Options](#)
- [Sequencing Types](#)
- [Sequencing and Preallocation Size](#)
- [Sequencing with EJB 2.n Entity Beans with Container-Managed Persistence](#)

## 18.2.1 Sequencing Configuration Options

You can configure sequencing using either Oracle JDeveloper TopLink Editor, TopLink Workbench, or Java (but not both).

Oracle recommends using Oracle JDeveloper to configure sequencing. Using Oracle JDeveloper, you can easily configure the sequencing options applicable to most applications. For more information, see [Section 20.3, "Configuring Sequencing at the Project Level"](#) or [Section 98.4, "Configuring Sequencing at the Session Level"](#).

Using TopLink Workbench, create one sequence with a single preallocation size that applies to all descriptors that require sequencing. You can configure table sequencing (see [Section 18.2.2.1, "Table Sequencing"](#)) or native sequencing (see [Section 18.2.2.5, "Native Sequencing with an Oracle Database Platform"](#)). If you choose table sequencing, you can either use default table and column names or specify your own (see [Section 18.2.2.1.1, "Default Versus Custom Sequence Table"](#)).

Using Java, you can configure any sequence type that TopLink supports (see [Section 18.2.2, "Sequencing Types"](#)). You can create any number and combination of sequences per project. You can create a sequence object explicitly or use the platform default sequence (see [Section 18.2.2.4, "Default Sequencing"](#)). You can associate the same sequence with more than one descriptor or associate different sequences (and different sequence types) to various descriptors. You can configure a separate preallocation size for each descriptor's sequence. For more information, see [Section 98.4.2, "How to Configure Sequencing at the Session Level Using Java"](#).

## 18.2.2 Sequencing Types

TopLink supports the following sequence types:

- [Table Sequencing](#)
- [Unary Table Sequencing](#)
- [Query Sequencing](#)
- [Default Sequencing](#)
- [Native Sequencing with an Oracle Database Platform](#)
- [Native Sequencing with a Non-Oracle Database Platform](#)

### 18.2.2.1 Table Sequencing

With table sequencing, you create a single database table that includes sequencing information for one or more sequenced objects in the project. TopLink maintains this table to track sequence numbers for these object types.

As [Figure 18–2](#) shows, the table may contain sequencing information for more than one class that uses sequencing. The default table is called `SEQUENCE` and contains two columns:

- `SEQ_NAME`, which specifies the class type to which the selected row refers

- `SEQ_COUNT`, which specifies the highest sequence number currently allocated for the object represented in the selected row

**Figure 18–2 TopLink Table Sequence Table**

SEQUENCE	
SEQ_NAME	SEQ_COUNT
SEQ_V_POOL	350
SEQ_MACHINERY	800
SEQ_PURCH_ORDER	1550
SEQ_WORK_ORDER	2400

The rows of the `SEQUENCE` table represent each sequence object: one for each class that participates in sequencing or a single sequence object across several classes so that they can benefit from the same preallocation pool. When you configure sequencing at the descriptor level (see [Section 23.3, "Configuring Sequencing at the Descriptor Level"](#)), you specify the `SEQ_NAME` for the class. Add a row with that name to the `SEQUENCE` table and initialize the `SEQ_COUNT` column to the value 0.

Each time a new instance of a class is created, TopLink obtains the required sequence value. For efficiency, TopLink uses preallocation to reduce the number of table accesses required to obtain sequence values (see [Section 18.2.3, "Sequencing and Preallocation Size"](#)).

You can create the `SEQUENCE` table on the database in one of two ways:

- Use either Oracle JDeveloper TopLink Editor or TopLink Workbench to create the table. See [Section 5.5.3.4, "Generating Tables on the Database"](#) for more information.
- Use the TopLink table creator to create and update the table manually. See [Section 5.5.3.1, "Generating SQL Creation Scripts"](#) for more information.

You can configure table sequencing using Oracle JDeveloper, TopLink Workbench, or Java. For more information about configuring table sequencing, see [Section 20.3, "Configuring Sequencing at the Project Level"](#) or [Section 98.4, "Configuring Sequencing at the Session Level"](#).

**18.2.2.1 Default Versus Custom Sequence Table** In most cases, you implement table sequencing using the default table and column names. However, you may want to specify your own table and column names if the following holds true:

- You want to use an existing sequence table for sequencing.
- You do not want to use the default naming convention for the table and its columns.

### 18.2.2.2 Unary Table Sequencing

Although similar to table sequencing (see [Section 18.2.2.1, "Table Sequencing"](#)), with unary table sequencing, you create a separate sequence table for each sequenced object in the project.

As [Figure 18–3](#) shows, sequencing information appears in the table for a single class that uses sequencing. You can name the table anything you want but it must contain only one column named (by default) `SEQUENCE`.

**Figure 18–3 TopLink Unary Table Sequence Table**

EMP_SEQ
SEQUENCE
350

When you configure sequencing at the descriptor level, you specify the sequence name for the class: this is the name of the unary table sequence table. [Figure 18–3](#) shows a unary table sequence for the `Employee` class. The `Employee` class descriptor is configured (see [Section 23.3, "Configuring Sequencing at the Descriptor Level"](#)) with a sequence name of `EMP_SEQ` to match the unary table sequence table name. TopLink adds a row to this table and initializes the `SEQUENCE` column to the value 1.

Each time a new class is created, TopLink obtains the required sequence value from the single row of the unary sequence table corresponding to the class. For efficiency, TopLink uses preallocation to reduce the number of table accesses required to obtain sequence values (see [Section 18.2.3, "Sequencing and Preallocation Size"](#)).

You can create the unary table sequence table on the database in one of two ways:

- Use either Oracle JDeveloper TopLink Editor or TopLink Workbench to create the table. See [Section 5.5.3.4, "Generating Tables on the Database"](#) for more information.
- Use the TopLink table creator to create and update the table manually. See [Section 5.5.3.1, "Generating SQL Creation Scripts"](#) for more information.

Currently, you can only configure unary table sequencing in Java using the `UnaryTableSequence` class (for more information, see [Section 98.4.2, "How to Configure Sequencing at the Session Level Using Java"](#)).

### 18.2.2.3 Query Sequencing

With query sequencing, you can access a sequence resource using custom read (`ValueReadQuery`) and update (`DataModifyQuery`) queries and a preallocation size that you specify. This allows you to perform sequencing using stored procedures and allows you to access sequence resources that are not supported by the other sequencing types that TopLink provides.

Currently, you can only configure query sequencing in Java using the `QuerySequence` class (for more information, see [Section 98.4.2.3, "Configuring Query Sequencing"](#)).

### 18.2.2.4 Default Sequencing

The platform owned by a login is responsible for providing a default sequence instance appropriate for the platform type. For example, by default, a `DatabasePlatform` provides a table sequence using the default table and column names (see [Section 18.2.2.1, "Table Sequencing"](#)).

You can access this default sequence directly using `DatasourceLogin` method `getDefaultSequence`, or indirectly by using the `DefaultSequence` class, a wrapper for the platform default sequence.

If you associate a descriptor with a nonexistent sequence, the TopLink runtime will create an instance of `DefaultSequence` to provide sequencing for that descriptor. For more information, see [Section 23.3.2.3, "Configuring the Platform Default Sequence"](#).



The main purpose of the `DefaultSequence` is to allow a sequence to use a different pre-allocation size than the project default.

Currently, you can only make use of default sequencing in Java (for more information, see [Section 98.4.2.1, "Using the Platform Default Sequence"](#)).

### 18.2.2.5 Native Sequencing with an Oracle Database Platform

TopLink support for native sequencing with Oracle Databases is similar to table sequencing (see [Section 18.2.2.1, "Table Sequencing"](#)), except that TopLink does not maintain a table in the database. Instead, the database contains a sequence object that stores the current maximum number and preallocation size for sequenced objects. The sequence name configured at the descriptor level identifies the sequence object responsible for providing sequencing values for the descriptor's reference class.

You can configure native sequencing using Oracle JDeveloper, TopLink Workbench, or Java. For more information about configuring table sequencing, see [Section 20.3, "Configuring Sequencing at the Project Level"](#) or [Section 98.4, "Configuring Sequencing at the Session Level"](#).

**18.2.2.5.1 Understanding the Oracle SEQUENCE Object** The Oracle `SEQUENCE` object implements a strategy that closely resembles TopLink sequencing: it implements an `INCREMENT` construct that parallels the TopLink preallocation size, and a `sequence.nextval` construct that parallels the `SEQ_COUNT` field in the TopLink `SEQUENCE` table in table sequencing. This implementation enables TopLink to use the Oracle `SEQUENCE` object as if it were a TopLink `SEQUENCE` table, but eliminates the need for TopLink to create and maintain the table.

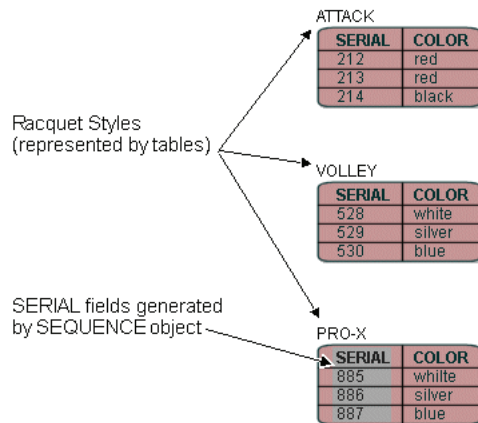
As with table sequencing, TopLink creates a pool of available numbers by requesting that the Oracle `SEQUENCE` object increment the `sequence.nextval` and return the result. Oracle adds the value, `INCREMENT`, to the `sequence.nextval`, and TopLink uses the result to build the sequencing pool.

The key difference between this process and the process involved in table sequencing is that TopLink is unaware of the `INCREMENT` construct on the `SEQUENCE` object. TopLink sequencing and the Oracle `SEQUENCE` object operate in isolation. To avoid sequencing errors in the application, set the TopLink preallocation size and the Oracle `SEQUENCE` object `INCREMENT` to the same value. Note that the Oracle sequence object must have a starting value equal to the preallocation size because when TopLink gets the next sequence value, it assume it has the previous preallocation size of values.

**18.2.2.5.2 Using SEQUENCE Objects** Your database administrator (DBA) must create a `SEQUENCE` object on the database for every sequencing series your application requires. If every class in your application requires its own sequence, the DBA creates a `SEQUENCE` object for every class; if you design several classes to share a sequence, the DBA need create only one `SEQUENCE` object for those classes.

For example, in [Figure 18-4](#), consider the case of a sporting goods manufacturer that manufactures three styles of tennis racquet. The data for these styles of racquet are stored in the database as follows:

- Each style of racquet has its own class table.
- Each manufactured racquet is an object represented by a line in the class table.
- The system assigns serial numbers to the racquets that use sequencing.

**Figure 18–4 Example of Database Tables–Racquet Information**

The manufacturer can do either of the following:

- *Use separate sequencing for each racquet style.* The DBA builds three separate SEQUENCE objects, perhaps called ATTACK\_SEQ, VOLLEY\_SEQ, and PROX\_SEQ. Each different racquet line has its own serial number series, and there may be duplication of serial numbers between the lines (for example: all three styles may include a racquet with serial number 1234).
- *Use a single sequencing series for all racquets.* The DBA builds a single SEQUENCE object (perhaps called RACQUET\_SEQ). The manufacturer assigns serial numbers to racquets as they are produced, without regard for the style of racquet.

### 18.2.2.6 Native Sequencing with a Non-Oracle Database Platform

Several databases support a type of native sequencing in which the database management system generates the sequence numbers.

When you create a database table for a class that uses native sequencing, include a primary key column, and set the column type as follows:

- For Sybase and Microsoft SQL Server databases, set the primary key field to the type `IDENTITY`.
- For IBM Informix databases, set the primary key field to the type `SERIAL`.
- For IBM DB2 databases, set the primary key field to the type `IDENTITY`.

When you insert a new object into the table, TopLink populates the object before insertion into the table, but does not include the sequence number. As the database inserts the object into its table, the database automatically populates the primary key field with a value equal to the primary key of the previous object plus 1.

At this point, and before the transaction closes, TopLink reads back the primary key for the new object so that the object has an identity in the TopLink cache.

---

**Note:** This type of sequencing does not support preallocation, so the preallocation size must be set to 1. To take advantage of sequence preallocation, Oracle recommends that you use table sequencing on these databases instead of native sequencing.

---

If your database provides native sequencing, but TopLink does not directly support it, you may be able to access the native sequence object using a query sequence and stored procedures. For more information, see [Section 18.2.2.3, "Query Sequencing"](#).

You can configure native sequencing using Oracle JDeveloper, TopLink Workbench, or Java. For more information about configuring table sequencing, see [Section 20.3, "Configuring Sequencing at the Project Level"](#) or [Section 98.4, "Configuring Sequencing at the Session Level"](#).

### 18.2.3 Sequencing and Preallocation Size

To improve sequencing efficiency, TopLink lets you preallocate sequence numbers. Preallocation enables TopLink to build a pool of available sequence numbers that are assigned to new objects as they are created and inserted into the database. TopLink assigns numbers from the sequence pool until the pool is empty.

The preallocation size specifies the size of the pool of available numbers. Preallocation improves sequencing efficiency by substantially reducing the number of database accesses required by sequencing. By default, TopLink sets preallocation size to 50. You can specify preallocation size either in Oracle JDeveloper TopLink Editor, TopLink Workbench, or as part of the session login.

Preallocation size configuration applies to table sequencing and Oracle native sequencing. In Oracle native sequencing, the sequence preallocation size must match the Oracle sequence object increment size. Preallocation is not available for native sequencing in other databases as they use an auto-assigned sequence column. Oracle recommends that you use table sequencing in non-Oracle databases to allow preallocation.

For table sequencing, TopLink maintains a pool of preallocated values for each sequenced class. When TopLink exhausts this pool of values, it acquires a new pool of values, as follows:

1. TopLink accesses the database, requesting that the `SEQ_COUNT` for the given class (identified by the `SEQ_NAME`) be incremented by the preallocation size and the result returned.

For example, consider the `SEQUENCE` table in [Figure 18–2](#). If you create a new purchase order and TopLink has exhausted its pool of sequence numbers, then TopLink executes a SQL statement to increment `SEQ_COUNT` for `SEQ_PURCH_ORDER` by the preallocation size (in this case, the TopLink default of 50). The database increments `SEQ_COUNT` for `SEQ_PURCH_ORDER` to 1600 and returns this number to TopLink.

2. TopLink calculates a maximum and a minimum value for the new sequence number pool, and creates the pool of values.
3. TopLink populates the object sequence attribute with the first number in the pool and writes the object to the class table.

As you add new objects to the class table, TopLink continues to assign values from the pool until it exhausts the pool. When the pool is exhausted, TopLink again requests new values from the table.

Using Oracle JDeveloper TopLink Editor and TopLink Workbench, you specify a preallocation size when you choose a sequencing type at the project or session level. That preallocation size applies to all descriptors.

Using Java, you can specify a different preallocation size for each sequence that you create.

For more information about configuring preallocation size, see [Section 20.3, "Configuring Sequencing at the Project Level"](#) or [Section 98.4, "Configuring Sequencing at the Session Level"](#).

#### 18.2.4 Sequencing with EJB 2.*n* Entity Beans with Container-Managed Persistence

To implement sequencing for entity beans with container-managed persistence, use a sequencing strategy that implements preallocation, such as table sequencing or Oracle native sequencing. Preallocation ensures that the entity bean primary key is available at the `ejbPostCreate` method. If you use non-Oracle native sequencing (for example, Sybase, Microsoft SQL Server, or Informix database native sequencing), be aware of the following:

- Non-Oracle native sequencing does not strictly conform to any EJB specification, because it does not initialize the primary key for a created object until you commit the transaction that creates the object. EJB specifications prior to 3.0 expect that the primary key is available at `ejbPostCreate` method.
- OC4J supports native sequencing; however, this type of native sequencing does not assign or return a primary key for a created object until you commit the transaction in which the object is created. Because of this, if you use native sequencing, commit a transaction immediately after calling the `ejbCreate` method to avoid problems with object identity in the TopLink cache and the container.

---

---

## Creating a Relational Project

This chapter describes the various components that you must configure in order to create a relational project.

This chapter includes the following sections:

- [Introduction to the Relational Project Creation](#)
- [Creating a Project from an Existing Object and Data Model](#)
- [Creating a Project from an Existing Object Model](#)
- [Creating a Project from an Existing Data Model](#)
- [Creating a Project from an OC4J EJB CMP EAR at Deployment Time](#)
- [Exporting Project Information](#)
- [Working with the ejb-xml.File](#)

For information on how to create more than one type of TopLink projects, see [Chapter 116, "Creating a Project"](#).

### 19.1 Introduction to the Relational Project Creation

You can create a project using Oracle JDeveloper TopLink Editor, TopLink Workbench, or Java code.

Oracle recommends using either Oracle JDeveloper or TopLink Workbench to create projects and generate deployment XML or Java source versions of the project for use at run time. For more information on how to create a project using TopLink Workbench, see [Section 116.1.2, "How to Create a Project Using TopLink Workbench"](#). For information on how to create a project using Java, see [Section 116.1.3, "How to Create a Project Using Java"](#).

You can use TopLink to create a relational project, if any of the following conditions are met:

- You have both an object and data model: see [Section 19.2, "Creating a Project from an Existing Object and Data Model"](#).
- You have an object model, but no data model yet: see [Section 19.3, "Creating a Project from an Existing Object Model"](#).
- You have a data model, but no object model yet: [Section 19.4, "Creating a Project from an Existing Data Model"](#).
- You are deploying an EJB CMP application to OC4J, you can create a project (including all mappings and data model) automatically at deployment time: see

[Section 19.5, "Creating a Project from an OC4J EJB CMP EAR at Deployment Time"](#).

For more information, see [Chapter 18, "Introduction to Relational Projects"](#).

## 19.2 Creating a Project from an Existing Object and Data Model

If you have both an existing object model (Java classes for your domain objects) and data model (such as an existing database schema), you can use either Oracle JDeveloper or TopLink Workbench to create your TopLink project.

### 19.2.1 How to Create a Project from an Existing Object and Data Model Using TopLink Workbench

1. Create the project (see [Section 116.1.2, "How to Create a Project Using TopLink Workbench"](#)).
2. Configure the project classpath (see [Section 117.3, "Configuring Project Classpath"](#)).
3. Import classes (see [Section 5.7.3, "How to Import and Update Classes"](#)).
4. Import database tables (see [Section 5.5.1.3, "Importing Tables from a Database"](#)).
5. Automatically create mappings (see [Section 120.3.1, "How to Create Mappings Automatically During Development Using TopLink Workbench"](#)).
6. Configure project options (see [Chapter 117, "Configuring a Project"](#)).

## 19.3 Creating a Project from an Existing Object Model

If you have an existing object model (Java classes for your domain objects), but you do not have a corresponding data model, you can use either Oracle JDeveloper or TopLink Workbench to create your TopLink project and automatically generate the corresponding data model.

### 19.3.1 How to Create a Project from an Existing Object Model Using TopLink Workbench

1. Create the project (see [Section 116.1.2, "How to Create a Project Using TopLink Workbench"](#)).
2. Configure the project classpath (see [Section 117.3, "Configuring Project Classpath"](#)).
3. Import classes (see [Section 5.7.3, "How to Import and Update Classes"](#)).
4. Generate database tables. For more information, see the following:
  - [Section 5.5.1.2, "Creating New Tables"](#)
  - [Section 5.5.3.4, "Generating Tables on the Database"](#)
  - [Section 20.9.1, "How to Configure Table Creator Java Source Options Using TopLink Workbench"](#)
  - [Section 19.6.2, "How to Export Table Creator Files Using TopLink Workbench"](#)
5. Configure project options (see [Chapter 117, "Configuring a Project"](#)).

## 19.4 Creating a Project from an Existing Data Model

If you have an existing data model (such as a database schema), but you do not have a corresponding data model (Java classes for domain objects), you can use either Oracle JDeveloper or TopLink Workbench to create your TopLink project and automatically generate the corresponding object model.

### 19.4.1 How to Create a Project from an Existing Data Model Using TopLink Workbench

1. Create the project (see [Section 116.1.2, "How to Create a Project Using TopLink Workbench"](#)).
2. Import database tables (see [Section 5.5.1.3, "Importing Tables from a Database"](#)).
3. Generate classes. For more information, see either of the following:
  - [Section 5.5.3.2, "Generating Classes and Descriptors from Database Tables"](#)
  - [Section 5.5.3.3, "Generating EJB Entity Beans and Descriptors from Database Tables"](#)
4. Configure project options (see [Chapter 117, "Configuring a Project"](#)).

## 19.5 Creating a Project from an OC4J EJB CMP EAR at Deployment Time

For a CMP application deployed to OC4J configured to use TopLink as the persistence manager, you can use the TopLink default mapping feature to automatically generate a TopLink project, including descriptors and mappings for all persistent objects, at deployment time.

This procedure applies only to CMP relational projects deployed to OC4J configured to use TopLink as the persistence manager.

For more information, see [Section 17.2.3.4, "Default Mapping in EJB 2.n CMP Projects Using OC4J at Run Time"](#).

## 19.6 Exporting Project Information

TopLink Workbench generates and exports the following project information:

- Project Java source (see [Section 19.6.1, "How to Export Project Java Source Using TopLink Workbench"](#))
- Table creator files (see [Section 19.6.2, "How to Export Table Creator Files Using TopLink Workbench"](#))

### 19.6.1 How to Export Project Java Source Using TopLink Workbench

For relational projects only, you can convert the project to Java source code. Generally, the generated code executes faster and deploys easier than XML files. See [Section 118.3, "Generating Java Code for Descriptors"](#) to export the model source for a *specific descriptor* in a project. To convert your relational project to Java source, use this procedure:

1. Right-click the project in the **Navigator** and choose **Export > Project Java Source** from the context menu.



You can also choose **Workbench > Export > Export Java Source** or **Selected > Export > Project Java Source** from the menu.

If you have not defined deployment and source code generation defaults (see [Chapter 117, "Configuring a Project"](#)) TopLink Workbench prompts for a project class name and directory.

---

**Note:** If your TopLink Workbench project uses the UTF-8 character set, you must use a compatible JDK when compiling the exported Java source.

If your project contains errors, the `project.xml` file may not be valid. See [Section A.3, "TopLink Workbench Error Reference"](#) for information on each reported error.

---

To generate Java source that is compatible with projects prior to this release, see [Section 20.11, "Configuring Deprecated Direct Mappings"](#).

## 19.6.2 How to Export Table Creator Files Using TopLink Workbench

For relational projects only, you can create Java source code to generate database tables defined in the project using this procedure:

1. Right-click the project in the **Navigator** and choose **Export > Table Creator Java Source** from the context menu.

You can also choose **Workbench > Export > Table Creator Java Source** or **Selected > Export > Table Creator Java Source** from the menu.

If you have not defined deployment and source code generation defaults (see [Chapter 117, "Configuring a Project"](#)) TopLink Workbench prompts for a class name and root directory.

## 19.7 Working with the ejb-xml.File

For relational projects that use EJB 2.0 CMP, use the `ejb-jar.xml` file to store persistence information for the application server. With TopLink Workbench, you can import information from an existing `ejb-jar.xml` file into your project, or you can create and update the `ejb-jar.xml` file from your project.

Each TopLink Workbench project uses a single `ejb-jar.xml` file. For each entity bean in the file, you should have an EJB descriptor in the project. All entity beans must use the same persistence type.

As you make changes in your project, you can update the `ejb-jar.xml` file to reflect your project. Additionally, if you edit the `ejb-jar.xml` file outside TopLink Workbench, you can update your project to reflect the current file.

[Table 19-1](#) describes how fields in the `ejb-jar.xml` file correspond to specific functions in TopLink Workbench.

**Table 19-1** *ejb-jar.xml Fields and TopLink Workbench*

<b>ejb-jar.xml Fields</b>	<b>TopLink Workbench</b>
<code>primkey</code>	Bean attribute mapped to the primary key in the database table (see <a href="#">Section 119.2, "Configuring Primary Keys"</a> ).
<code>ejb-name</code> , <code>prim-key-class</code> , <code>local</code> , <code>local-home</code> , <code>remote</code> , <code>home</code> , and <code>ejb-class</code>	EJB descriptor information on the EJB Info tab (see <a href="#">Section 119.18, "Configuring a Descriptor with EJB CMP and BMP Information"</a> ).



**Table 19–1 (Cont.) ejb-jar.xml Fields and TopLink Workbench**

ejb-jar.xml Fields	TopLink Workbench
abstract-schema-name	Descriptor Alias field (see <a href="#">Section 119.5, "Configuring Descriptor Alias"</a> ).
cmp-field	Direct (non-relationship) attributes on the Descriptor Info tab (see <a href="#">Section 119.1, "Configuring Common Descriptor Options"</a> ).
cmp-version	Persistence Type field on the General tab (see <a href="#">Section 117.5, "Configuring Persistence Type"</a> ). The persistence-type is set to container.
query	Queries listed in Queries tab (see <a href="#">Section 119.7, "Configuring Named Queries at the Descriptor Level"</a> ). Note: The <code>findByPrimaryKey</code> query is not in the <code>ejb-jar.xml</code> file as per the EJB 2.0 specification.
relationships	One-to-one, one-to-many, and many-to-many mappings (see <a href="#">Part XII, "Relational Mappings"</a> ).

---

**Note:** You can also use Oracle JDeveloper TopLink Editor for reading from and writing to the `ejb-jar.xml` file. For more information, see the Oracle JDeveloper online help.

---

For more information, see [Section 9.1.3, "ejb-jar.xml File"](#).

### 19.7.1 How to Write to the ejb-jar.xml File Using TopLink Workbench

To update the `ejb-jar.xml` file based on the current TopLink Workbench information, use this procedure:

---

**Note:** Use the EJB preferences to specify whether or not TopLink Workbench automatically updates the `ejb-jar.xml` file when you save the project.

---



---

**Note:** You can also write the information to a `.jar` file. TopLink Workbench automatically places the `ejb-jar.xml` file in the proper location (`META-INF/ejb-jar.xml`).

---

1. Choose **Selected > Write Project to ejb-jar.xml** from the menu.

You can also right-click the project in the **Navigator** and choose **Write Project to ejb-jar.xml** from the context menu.

- If the project does not currently contain an `ejb-jar.xml` file, the system prompts you to create a new file.
- If the system detects that changes were made to the `ejb-jar.xml` file but not yet read into TopLink Workbench (for example, you changed the file outside TopLink Workbench), then the system prompts you to read the file before writing the changes.

## 19.7.2 How to Read from the ejb-jar.xml File Using TopLink Workbench

To read the `ejb-jar.xml` information and update your TopLink Workbench project, use this procedure.

---

---

**Tip:** To automatically create EJB descriptors in TopLink Workbench for all entities, read the `ejb-jar.xml` file before adding any classes in TopLink Workbench.

---

---

1. Choose **Selected > Update Project from ejb-jar.xml** from the menu.

You can also right-click the project in the **Navigator** window and choose **Update Project from ejb-jar.xml** from the context menu.

---

---

**Note:** If you are using TopLink Workbench behind a firewall, before reading from the `ejb-jar.xml` file, you may need to configure TopLink Workbench with a proxy (see [Section 5.4.2, "How to Use Help Preferences"](#)). If TopLink Workbench fails to read the `ejb-jar.xml` file due to connection timeout or no route to host, proxy configuration is required.

---

---

---



---

## Configuring a Relational Project

This chapter describes the various components that you must configure in order to use a relational project.

This chapter contains the following sections:

- [Introduction to Relational Project Configuration](#)
- [Configuring Relational Database Platform at the Project Level](#)
- [Configuring Sequencing at the Project Level](#)
- [Configuring Login Information at the Project Level](#)
- [Configuring Development and Deployment Logins](#)
- [Configuring Named Query Parameterized SQL and Statement Caching at the Project Level](#)
- [Configuring Table Generation Options](#)
- [Configuring Table Creator Java Source Options](#)
- [Configuring Project Java Source Code Options](#)
- [Configuring Deprecated Direct Mappings](#)

This chapter also describes logging into a database during development when using TopLink Workbench. For more information, see [Section 20.6, "Logging In to the Database"](#).

For information on how to configure TopLink project options common to two or more project types, see [Chapter 117, "Configuring a Project"](#).

### 20.1 Introduction to Relational Project Configuration

In addition to the configurable options described here, you must also configure the base class options described in [Table 117-2, "Common Project Options"](#).

[Table 20-1](#) lists the configurable options for relational projects.

**Table 20-1 Configurable Options for Relational Projects**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Project save location (see <a href="#">Section 117.2, "Configuring Project Save Location"</a> )		✓	
Persistence type (see <a href="#">Section 117.5, "Configuring Persistence Type"</a> )	✓	✓	

**Table 20–1 (Cont.) Configurable Options for Relational Projects**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Project classpath (see Section 117.3, "Configuring Project Classpath")		✓	
Project comments (see Section 117.14, "Configuring Project Comments")	✓	✓	
Method or direct field access (see Section 117.4, "Configuring Method or Direct Field Access at the Project Level")	✓	✓	✓
Default descriptor advanced properties (see Section 117.6, "Configuring Default Descriptor Advanced Properties")	✓	✓	✓
Existence checking (see Section 117.7, "Configuring Existence Checking at the Project Level")	✓	✓	✓
Project deployment XML options (see Section 117.8, "Configuring Project Deployment XML Options")		✓	✓
Model Java source code options (see Section 117.9, "Configuring Model Java Source Code Options")	✓	✓	✓
Relational database (see Section 20.2, "Configuring Relational Database Platform at the Project Level")	✓	✓	✓
Sequencing (see Section 20.3, "Configuring Sequencing at the Project Level")	✓	✓	✓
Login information (see Section 20.4, "Configuring Login Information at the Project Level")	✓	✓	✓
Development and deployment logins (see Section 20.5, "Configuring Development and Deployment Logins")	✓	✓	
Cache type and size (see Section 117.10, "Configuring Cache Type and Size at the Project Level")	✓	✓	✓
Cache isolation (see Section 117.11, "Configuring Cache Isolation at the Project Level")	✓	✓	✓
Cache coordination change propagation (see Section 117.12, "Configuring Cache Coordination Change Propagation at the Project Level")	✓	✓	✓
Cache expiration (see Section 117.13, "Configuring Cache Expiration at the Project Level")	✓	✓	
Named query parameterized SQL and statement caching (see Section 20.7, "Configuring Named Query Parameterized SQL and Statement Caching at the Project Level")	✓	✓	
Table generation options (see Section 20.8, "Configuring Table Generation Options")		✓	
Table creator Java source options (see Section 20.9, "Configuring Table Creator Java Source Options")		✓	
Project Java source code options (see Section 20.10, "Configuring Project Java Source Code Options")		✓	
Deprecated direct mappings (see Section 20.11, "Configuring Deprecated Direct Mappings")	✓	✓	

For more information, see [Chapter 18, "Introduction to Relational Projects"](#).

## 20.2 Configuring Relational Database Platform at the Project Level

For each relational project, you must specify the database platform (such as Oracle Database 10g). This platform configuration is overridden by the session login, if configured.

For more information, see the following:

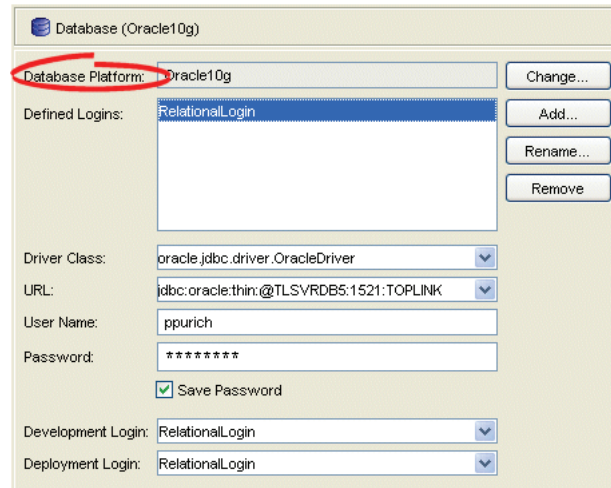
- [Section 98.2, "Configuring a Relational Database Platform at the Session Level"](#)
- [Section 96.1.3, "Data Source Platform Types"](#)

## 20.2.1 How to Configure Relational Database Platform at the Project Level Using TopLink Workbench

To specify the database platform of a relational project, use this procedure:

1. Select the database object in the **Navigator**. The Database property sheet appears.

**Figure 20–1 Database Property Sheet, Database Platform Options**



Click **Change** to select a new database platform for the project. For more information, see [Section 96.1.3, "Data Source Platform Types"](#).

## 20.3 Configuring Sequencing at the Project Level

Sequencing allows TopLink to automatically assign the primary key or ID of an object when the object is inserted.

You configure TopLink sequencing at the project or session level to tell TopLink how to obtain sequence values: that is, what type of sequences to use.

In a CMP project, you do not configure a session directly: in this case, you must configure sequences at the project level. In a POJO project, you can configure a session directly: in this case, you can use a session-level sequence configuration to override project-level sequence configuration, on a session-by-session basis, if required (see [Section 98.4, "Configuring Sequencing at the Session Level"](#)).

You can configure sequencing using Oracle JDeveloper.

Using TopLink Workbench (see [Section 20.3.2, "How to Configure Sequencing at the Project Level Using TopLink Workbench"](#)), you can configure table sequencing (see [Section 18.2.2.1, "Table Sequencing"](#)) and native sequencing ([Section 18.2.2.5, "Native Sequencing with an Oracle Database Platform"](#) and [Section 18.2.2.6, "Native Sequencing with a Non-Oracle Database Platform"](#)) and you can configure a preallocation size that applies to all sequences (see [Section 18.2.3, "Sequencing and Preallocation Size"](#)).

Using Java (see [Section 20.3.3, "How to Configure Sequencing at the Project Level Using Java"](#)), you can configure any sequence type that TopLink supports

(Section 18.2.2, "Sequencing Types"). You can create any number and combination of sequences. You can create a sequence object explicitly or use the default sequence that the platform creates. You can associate the same sequence with more than one descriptor and you can configure a separate preallocation size for each descriptor's sequence.

After configuring the sequence type at the project (or session) level, to enable sequencing, you must configure a descriptor with a sequence field and a sequence name (see Section 23.3, "Configuring Sequencing at the Descriptor Level").

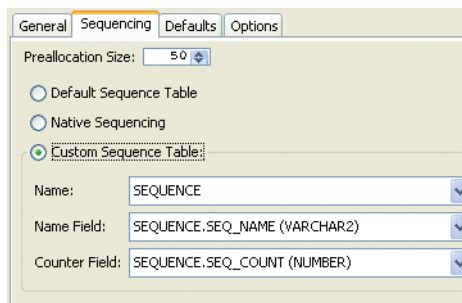
For more information about sequencing, see Section 18.2, "Sequencing in Relational Projects".

### 20.3.1 How to Configure Sequencing at the Project Level Using TopLink Workbench

To specify the sequencing information for the project, use this procedure:

1. Select the project object in the **Navigator**.
2. Select the **Sequencing** tab in the **Editor**. The Sequencing tab appears.

**Figure 20–2 Sequencing Tab**



Use this table to enter data in the following fields to configure the sequencing information:

Field	Description
<b>Preallocation Size</b>	Specify the default preallocation size (see Section 18.2.3, "Sequencing and Preallocation Size"). Default is <b>50</b> . The preallocation size you configure applies to all sequences.
<b>Default Sequence Table</b>	Select this option to use table sequencing (see Section 18.2.2.1, "Table Sequencing") with default sequence table name <code>SEQUENCE</code> , default sequence name field <code>SEQ_NAME</code> , and default sequence counter field <code>SEQ_COUNT</code> .
<b>Native Sequencing</b>	Select this option to use a sequencing object (see Section 18.2.2.5, "Native Sequencing with an Oracle Database Platform" or Section 18.2.2.6, "Native Sequencing with a Non-Oracle Database Platform") created by the database platform. This option applies only to Oracle, Sybase, Microsoft SQL, and IBM Informix database platforms.
<b>Custom Sequence Table</b>	Select this option to use table sequencing (see Section 18.2.2.1, "Table Sequencing") with a sequence table name, sequence name field, and sequence counter field name that you specify.
<b>Name</b>	Specify the name of the sequence table.
<b>Name Field</b>	Specify the name of the column used to store the sequence name.

Field	Description
Counter Field	Specify the name of the column used to store the sequence count.

### 20.3.2 How to Configure Sequencing at the Project Level Using Java

Using Java, you can configure a project to use multiple, different sequences, as [Example 20–1](#) shows.

#### **Example 20–1** Configuring Sequencing at the Project Level in Java

```
// Enable native sequencing for the project as the default. Configured the default
// preallocation size
project.getLogin().useNativeSequencing();
project.getLogin().setSequencePreallocationSize(50);

// Configure the EMP_SEQ to not use preallocation
DefaultSequence empSequence = new DefaultSequence("EMP_SEQ", 1);
project.getLogin().addSequence(empSequence);

// Configure the PROJ_SEQ to use a separate sequence table
UnarySequence projSequence = new UnarySequence("PROJ_SEQ_TAB", "COUNTER");
project.getLogin().addSequence(projSequence);
```

## 20.4 Configuring Login Information at the Project Level

This section describes how to define a login to a relational database. After you define a login, you must designate its role (see [Section 20.5, "Configuring Development and Deployment Logins"](#)).

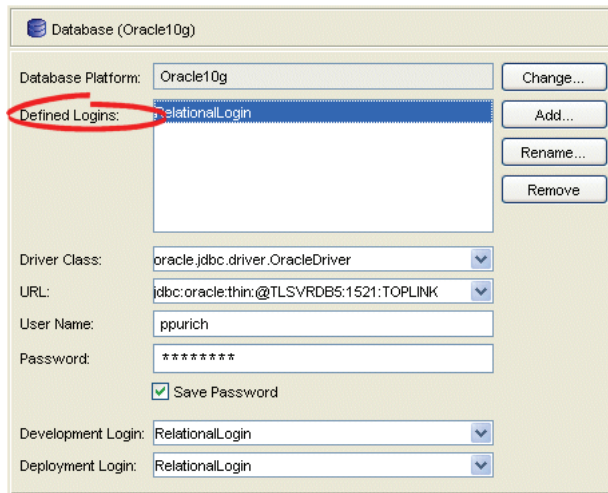
After you create a login (see [Section 20.4, "Configuring Login Information at the Project Level"](#)) and specify it as a development login (see [Section 20.5, "Configuring Development and Deployment Logins"](#)), you can log in to a database instance (see [Section 20.6, "Logging In to the Database"](#)).

### 20.4.1 How to Configure Login Information at the Project Level Using TopLink Workbench

To create or edit a database login, use this procedure:

1. Select the database object in the **Navigator**. The Database property sheet appears.

**Figure 20–3 Database Property Sheet, Database Login Fields**



2. Click **Add** to create a new Defined Login.
3. Complete the Database Login fields on the property sheet.

Use this table to enter data in the following fields on the Database property sheet to configure the database login:

Field	Description
<b>Defined Logins</b>	Login used to access the database. Click <b>Add</b> to add a new login, or <b>Remove</b> to delete an existing login.
<b>Driver Class</b>	The JDBC driver to use to connect to the database.
<b>URL</b>	The URL used to connect to the appropriate database.
<b>User Name</b>	The name required to log in to the database.
<b>Password</b>	The password required to log in to the database.
<b>Save Password</b>	Whether or not to save the <b>Password</b> for this <b>Defined Login</b> . Oracle recommends that you do not save the password with a deployment login. <b>Note:</b> If you select <b>Save Password</b> , then when you export Java source and deployment XML (see <a href="#">Section 116.3, "Exporting Project Information"</a> ), TopLink Workbench writes the database password using JCE encryption (when using JDK 1.4 or later). For information on how to specify password encryption options, see <a href="#">Section 97.3, "Configuring Password Encryption"</a> . Default: unselected.

## 20.5 Configuring Development and Deployment Logins

This section describes how to designate a defined login’s role. For information on how to define a login, see [Section 20.4, "Configuring Login Information at the Project Level"](#). TopLink recognizes the following login roles:

- [Development Role](#)
- [CMP Deployment Role](#)
- [POJO Session Role](#)



## Development Role

While using TopLink Workbench to develop a project (see [Section 15.2.4.3, "Development Role"](#)), you must define a login (see [Section 20.4, "Configuring Login Information at the Project Level"](#)) and designate it as the development login. The development login is stored in the TopLink project file. TopLink Workbench use the information in the development login whenever you perform a data source operation from within TopLink Workbench. For example, when you read or write schema information from or to a data source during application development, the development login information is never written to a `sessions.xml` or `project.xml` file and is overridden by the deployment login (or the session login) at run time.

For more information on how to use a development login to connect to a database, see [Section 20.6, "Logging In to the Database"](#).

## CMP Deployment Role

If you are creating a CMP project (see [Section 15.2.4.2, "CMP Deployment Role"](#)), you may define a run-time login (see [Section 20.4, "Configuring Login Information at the Project Level"](#)) and designate it as the deployment login. This is the login that the application will use at run time, unless overridden in the `sessions.xml` file, CMP deployment file, or through Java code.

## POJO Session Role

If you are creating a POJO project (see [Section 15.2.4.1, "POJO Session Role"](#)), Oracle recommends that you use the `sessions.xml` file to store the sessions your project uses at run time (see [Section 96.1.2, "Data Source Login Types"](#)).

## 20.5.1 How to Configure Development and Deployment Logins Using TopLink Workbench

To specify different development and deployment database logins, use this procedure:

1. Select the database object in the **Navigator**. The Database property sheet appears.

**Figure 20–4 Database Property Sheet, Development and Deployment Login Options**

Use this table to enter data in the following fields on the Database property sheet to configure the login:

Field	Description
Development Login	The <b>Defined Login</b> to be used by TopLink Workbench during development to connect with the database, and to read or write table information.  For more information on how to use a development login to connect to a database, see <a href="#">Section 20.6, "Logging In to the Database"</a> .
Deployment Login	The <b>Defined Login</b> to be used by your TopLink-enabled application during deployment.

## 20.6 Logging In to the Database

Using Oracle JDeveloper or TopLink Workbench, after you create a login (see [Section 20.4, "Configuring Login Information at the Project Level"](#)) and specify it as a development login (see [Section 20.5, "Configuring Development and Deployment Logins"](#)), you can log in to a database instance.

You must log in to the database before importing or exporting table information.

To log in to the database using TopLink Workbench, use one of the following procedures:



- Select the database object in the **Navigator** and click **Login**. TopLink Workbench logs in to the database.
- Right-click on the database object in the **Navigator** and choose **Log In to Database** from the context menu, or choose **Selected > Log In to Database** from the menu.



The database icon in the Navigator window changes to indicate you are now logged in to the database.

## 20.7 Configuring Named Query Parameterized SQL and Statement Caching at the Project Level

You can configure TopLink to use parameterized SQL (parameter binding) and prepared statement caching for all named queries and finders.

By default, TopLink uses parameterized SQL.

The use of parameterized SQL lets you create and store queries that are complete except for one or more bound parameters. The TopLink runtime binds the current parameter values when executing the query. This approach avoids the preparation of SQL execution and, thus, improves the performance of frequently executed SQL statements.

This section describes configuring parameterized SQL and statement caching options at the project level. This configuration applies to *all* named queries or finders (see [Section 108.8, "Named Queries"](#) or [Section 108.15, "EJB 2.n CMP Finders"](#)) you create on the descriptors in this project—not to all queries in general or write operations.

You can also configure parameterized SQL and statement caching options at the named query or finder-level to override this project-level configuration on a query-by-query basis (see [Section 119.7.2.9, "Configuring Named Query Options"](#)) or at the session login-level (see [Section 98.6, "Configuring JDBC Options"](#)).

For more information, see [Section 12.11.5, "How to Use Parameterized SQL \(Parameter Binding\) and Prepared Statement Caching for Optimization"](#).

**Note:** For applications using a Java EE data source or external connection pool, you must configure statement caching in the Java EE server's data source—not in TopLink.

Table 20–2 summarizes which projects support parameterized SQL and statement caching configuration.

**Table 20–2 Project Support for Default Named Query Caching and Binding**

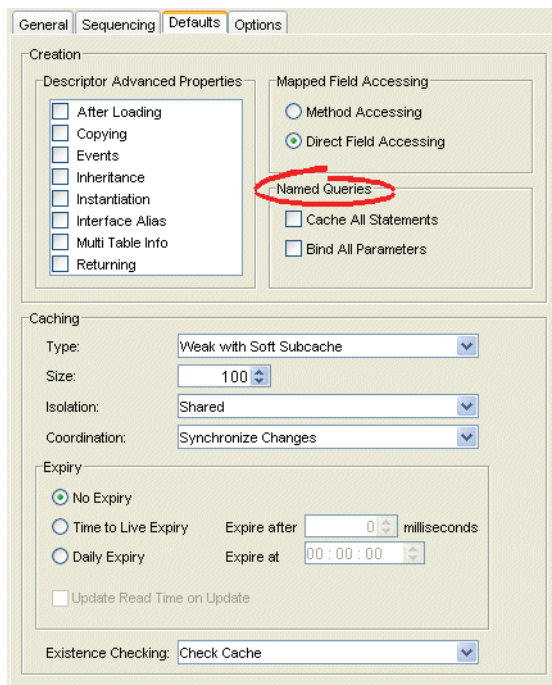
Descriptor	How to Use Oracle JDeveloper	How to Configure Named Query Parameterized SQL and Statement Caching at the Project Level Using TopLink Workbench	How to Use Java
Relational Projects	✓	✓	✓
EIS Projects			
XML Projects			

### 20.7.1 How to Configure Named Query Parameterized SQL and Statement Caching at the Project Level Using TopLink Workbench

To specify the named query options, use this procedure:

1. Select the project object in the **Navigator**.
2. Select the **Defaults** tab in the **Editor**. The Defaults tab appears.

**Figure 20–5 Defaults Tab, Named Queries Options**



Use this table to enter data in following fields on the Defaults tab to specify the named query options for newly created descriptors.:

Field	Description
Cache All Statements	Caches the query's prepared statement in the TopLink statement cache.
Bind All Parameters	By default, TopLink binds all of the query's parameters. Deselect this option to disable binding.

## 20.8 Configuring Table Generation Options

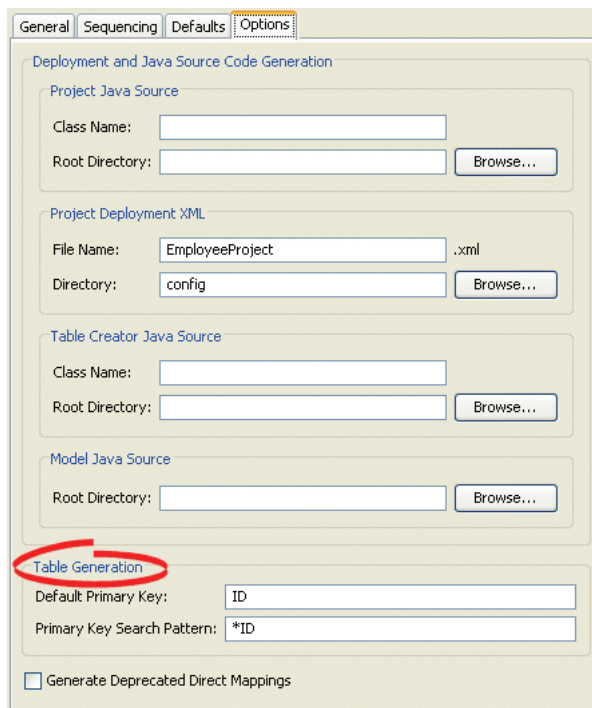
Using TopLink Workbench, you can configure options that apply when you generate database tables from the descriptors you define in your TopLink Workbench project. The resulting tables and columns will conform to the naming restrictions of the project's target database.

### 20.8.1 How to Configure Table Generation Options Using TopLink Workbench

To specify the default table generation options, use this procedure:

1. Select the project object in the **Navigator**.
2. Select the **Options** tab in the **Editor**. The Options tab appears.

**Figure 20–6 Options Tab, Table Generation Options**



Use this table to enter data in the following fields to specify the default export and generation options.

Field	Description
Default Primary Key	Enter the default name to use when generating primary keys.
Primary Key Search Pattern	Enter the default search pattern to use when generating primary keys.

## 20.9 Configuring Table Creator Java Source Options

Using TopLink Workbench, you can configure options that apply when you export Java source code that you can use to create database tables.

### 20.9.1 How to Configure Table Creator Java Source Options Using TopLink Workbench

To specify the default Java code generation options, use this procedure:

1. Select the project object in the **Navigator**.
2. Select the **Options** tab in the **Editor**. The Options tab appears.

**Figure 20–7 Options Tab, Table Creator Java Source Options**

The screenshot shows the 'Options' tab in TopLink Workbench. The 'Table Creator Java Source' section is highlighted with a red circle. The 'Table Generation' section shows the following values:

Field	Value
Default Primary Key	ID
Primary Key Search Pattern	*ID

Use this table to enter data in the following fields to specify the default table creator options.

Field	Description
Class Name	Base class name to use when generating table's Java source code from the project.
Root Directory	Directory for storing the generated source code.

## 20.10 Configuring Project Java Source Code Options

Using TopLink Workbench, you can export a project as Java source. You can configure the class name and root directory that TopLink Workbench uses when exporting the project to Java source code.

For more information on exporting a project as Java source, see [Section 19.7.1, "How to Export Project Java Source Using TopLink Workbench"](#).

### 20.10.1 How to Configure Project Java Source Code Options Using TopLink Workbench

To specify the default Java code generation options, use this procedure:

1. Select the project object in the **Navigator**.
2. Select the **Options** tab in the **Editor**. The Options tab appears.

**Figure 20–8 Options Tab, Project Java Source Options**

The screenshot shows the 'Options' tab in the TopLink Workbench interface. The 'Project Java Source' section is highlighted with a red circle. It contains fields for 'Class Name' and 'Root Directory' with a 'Browse...' button. Below it is the 'Project Deployment XML' section with 'File Name' (EmployeeProject.xml) and 'Directory' (config) fields, also with a 'Browse...' button. The 'Table Creator Java Source' section has 'Class Name' and 'Root Directory' fields with a 'Browse...' button. The 'Model Java Source' section has a 'Root Directory' field with a 'Browse...' button. At the bottom, the 'Table Generation' section has 'Default Primary Key' (ID) and 'Primary Key Search Pattern' (\*ID) fields. A checkbox for 'Generate Deprecated Direct Mappings' is at the bottom left.

Use this table to enter data in the following fields to specify the default export and generation options:

Field	Description
Class Name	Base class name to use when generating Java source code from the project.
Root Directory	Directory for storing the generated source code.

## 20.11 Configuring Deprecated Direct Mappings

You can configure deprecated direct mapping options when using Oracle JDeveloper or TopLink Workbench.

Starting with the 10.1.3.1 release, TopLink no longer uses the following direct mapping types:

- Type conversion
- Object type
- Serialized object

Instead, TopLink uses a direct-to-field mapping with a specialized converter.

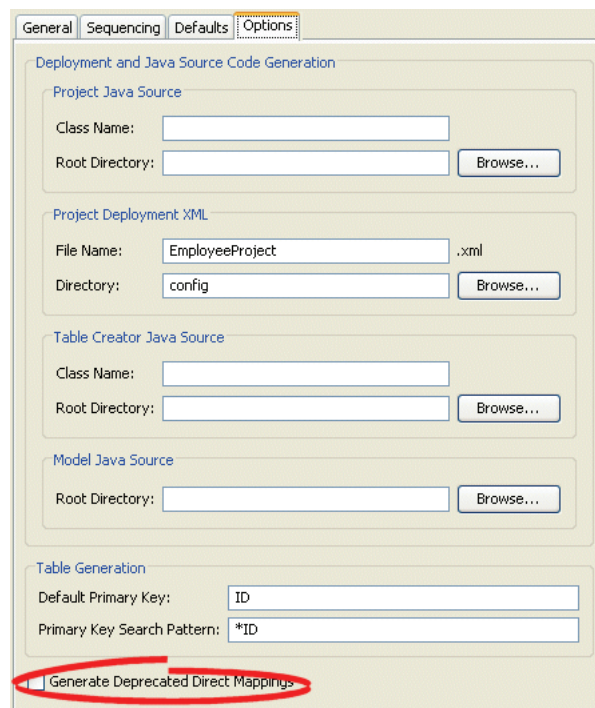
To generate backward-compatible deployment XML and Java source code files, use the **Generate Deprecated Direct Mappings** option.

### 20.11.1 How to Configure Deprecated Direct Mappings Using TopLink Workbench

To specify if TopLink Workbench should generate the deprecated direct mappings (instead of using the converter) when exporting projects, use this procedure:

1. Select the project object in the **Navigator**.
2. Select the **Options** tab in the **Editor**. The Options tab appears.

**Figure 20–9 Options Tab, Generate Deprecated Direct Mappings Option**



Select the **Generate Deprecated Direct Mappings** option on the tab to specify that TopLink Workbench should generate backward-compatible code (using the deprecated direct mappings, instead of the converter).





# Part X

---

## Relational Descriptors

This part contains general information about relational descriptors, as well as detailed information on how to create and configure these descriptors.

This part includes the following chapters:

- [Chapter 21, "Introduction to Relational Descriptors"](#)

This chapter introduces concepts of relational descriptors.

- [Chapter 22, "Creating a Relational Descriptor"](#)

This chapter explains how to create relational descriptors.

- [Chapter 23, "Configuring a Relational Descriptor"](#)

This chapter explains how to configure descriptor options specific to a relational descriptor.



---

## Introduction to Relational Descriptors

This chapter provides an overview of relational descriptors, as well as explains the role of inheritance and various types of descriptors in relational projects.

This chapter includes the following sections:

- [Relational Descriptors](#)
- [Aggregate and Composite Descriptors in Relational Projects](#)
- [Descriptors and Inheritance in Relational Projects](#)

For information on descriptor concepts and features common to more than one type of TopLink descriptors, see [Chapter 16, "Introduction to Descriptors"](#).

### 21.1 Relational Descriptors

Relational descriptors describe Java objects that you map to tables in a relational database. You use them in relational projects (see [Chapter 18, "Introduction to Relational Projects"](#)).

Using relational descriptors in a relational project, you can configure relational mappings (see [Section 27.1, "Relational Mapping Types"](#)).

For more information, see the following:

- [Section 22.2, "Creating a Relational Descriptor"](#)
- [Chapter 23, "Configuring a Relational Descriptor"](#)

### 21.2 Aggregate and Composite Descriptors in Relational Projects

In a relational project, you can designate the descriptor as an aggregate (see [Section 22.2.1.2, "Creating Relational Aggregate Descriptors"](#)).

This lets you configure an aggregate mapping (see [Chapter 37, "Configuring a Relational Aggregate Object Mapping"](#)) to associate data members in the target object with fields in the source object's underlying database tables.

When you designate a relational descriptor as an aggregate, TopLink lets you specify a mapping type for each field in the target class, but defers associating the field with a database table until you configure the aggregate object mapping in the source descriptor. In other words, the target class descriptor defines *how* each target class field is mapped, but the source class descriptor defines *where* each target class field is mapped. This lets you share an aggregate object among many parent descriptors mapped to different tables.

When you designate a relational descriptor as an aggregate, you tell TopLink that the class will be a target of an aggregate object mapping, and this ensures that the TopLink runtime handles the target class as follows:

- It inserts, updates, and deletes the target class in parallel with its source class.
- It does not cache the target class on its own; instead, it caches the target class as part of its source class.
- It does not allow the target class to be read, written, deleted, or registered in a unit of work.

When working with aggregate relational descriptors, consider the following:

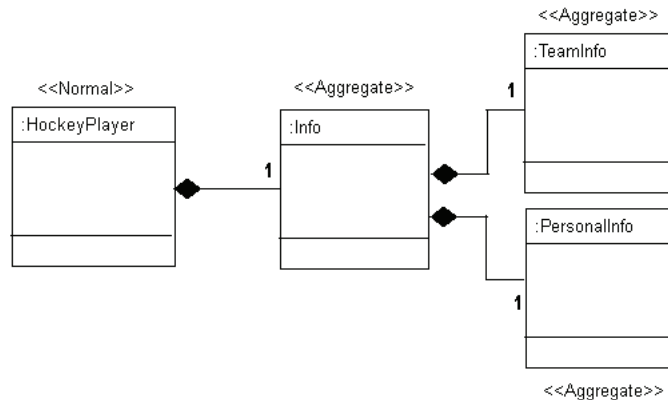
- [Relational Aggregates and Nesting](#)
- [Relational Aggregates and Inheritance](#)
- [Relational Aggregates and EJB 2.n Entity Beans](#)

For more information, see [Section 23.6, "Configuring a Relational Descriptor as a Class or Aggregate Type"](#).

### 21.2.1 Relational Aggregates and Nesting

TopLink supports nested aggregates. In [Figure 21–1](#) source class `HockeyPlayer` is a normal nonaggregate class descriptor. It owns target class `Info` which is designated as an aggregate. The `Info` class itself owns target classes `PersonalInfo` and `TeamInfo` which are each designated as aggregates.

**Figure 21–1 Nested Aggregates**



In EJB 3.0, an aggregate is known as an embeddable. In the EJB 3.0 specification, an embeddable may not contain another embeddable (that is, the EJB 3.0 specification does not support nested aggregates).

However, if you deploy a TopLink-enabled EJB 3.0 application with persistence to Oracle WebLogic Server, you can take advantage of an EclipseLink extension of the EJB 3.0 specification to configure nested embeddables. Note that if you do so, your application will not be strictly EJB 3.0-compliant. [Example 21–1](#) shows the classes from [Figure 21–1](#) using EJB 3.0 annotations to take advantage of the EclipseLink extension of the EJB 3.0 specification to allow `Info` (an embeddable) to own embeddables `TeamInfo` and `PersonalInfo`.

**Example 21–1 Nested Embeddables**

```
public class HockeyPlayer implements Serializable {
```

```

private int playerId;
private Info info;
private String lastName;
private String firstName;
...
@Embedded
public Info getInfo() {
    return info;
}
}

@Embeddable
public class Info implements Serializable {
    TeamInfo teamInfo; // EclipseLink extension of EJB 3.0 allows Embeddable with Embeddable
    PersonalInfo personalInfo;

    public Info() {}

    @Embedded
    public PersonalInfo getPersonalInfo() {
        return personalInfo;
    }

    public void setPersonalInfo(PersonalInfo personalInfo) {
        this.personalInfo = personalInfo;
    }

    @Embedded
    public TeamInfo getTeamInfo() {
        return teamInfo;
    }

    public void setTeamInfo(TeamInfo teamInfo) {
        this.teamInfo = teamInfo;
    }
}

@Embeddable
public class PersonalInfo implements Serializable {
    private int age;
    private double weight;
    private double height;
    ...
}

@Embeddable
public class TeamInfo implements Serializable {
    private String position;
    private int jerseyNumber;
    private HockeyTeam hockeyTeam;
    ...
}

```

## 21.2.2 Relational Aggregates and Inheritance

You can configure inheritance for a relational descriptor designated as an aggregate (see [Section 16.2.2, "Descriptors and Inheritance"](#)), however, in this case, *all* the descriptors in the inheritance tree must be aggregates. Aggregate and class descriptors cannot exist in the same inheritance tree.

### 21.2.3 Relational Aggregates and EJB 2.n Entity Beans

You can use relational aggregate descriptors in an EJB project, but you cannot configure EJB information for a relational descriptor designated as an aggregate (see [Section 16.2.3, "Descriptors and CMP and BMP"](#)).

For information on using relational aggregates and EJB 3.0, see [Section 21.2.1, "Relational Aggregates and Nesting"](#).

## 21.3 Descriptors and Inheritance in Relational Projects

Inheritance describes how a derived class inherits the characteristics of its superclass. You can use descriptors to describe the inheritance relationships between classes in your relational projects.

This section includes information on the following topics:

- [Inheritance and Primary Keys in Relational Projects](#)
- [Single- and Multi-Table Inheritance in Relational Projects](#)

For more information, see [Section 16.3, "Descriptors and Inheritance"](#).

### 21.3.1 Inheritance and Primary Keys in Relational Projects

For relational projects, TopLink assumes that all of the classes in an inheritance hierarchy have the same primary key, as set in the root descriptor. Child descriptors associated with data source representations that have different primary keys must define the mapping between the root primary key and the local one.

For more information, see [Section 119.2, "Configuring Primary Keys"](#).

### 21.3.2 Single- and Multi-Table Inheritance in Relational Projects

In a relational project, you can map your inheritance hierarchy to a single table (see [Section 21.3.2.1, "Single-Table Inheritance"](#)) or to multiple tables (see [Section 21.3.2.2, "Multi-Table Inheritance"](#)). Use these options to achieve the balance between storage efficiency and access efficiency that is appropriate for your application.

#### 21.3.2.1 Single-Table Inheritance

In this example, you store classes with multiple levels of inheritance in a single table to optimize database access speeds.

The entire inheritance hierarchy shown in [Figure 21–1, "Nested Aggregates"](#) can share the same table, as in [Figure 21–2](#). The `FueledVehicle` and `NonFueledVehicle` subclasses can share the same table even though `FueledVehicle` has some attributes that `NonFueledVehicle` does not. The `NonFueledVehicle` instances waste database resources because the database must still allocate space for the unused portion of its row. However, this approach saves on accessing time because there is no need to join to another table to get the additional `FueledVehicle` information.

As [Figure 21–2](#) shows, this approach uses a class indicator field. For more information, see [Section 16.3.1, "How to Specify a Class Indicator"](#).

**Figure 21–2 Inheritance Using a Superclass Table with Optional Fields**

VEHICLE table

ID	PASS_CAP	VHCL_TYPE	FUEL_CAP	FUEL_TYPE	CAR_DESCR	BICYCLE_DESCR
1	1	B				Mountain Bike
2	3	V				
3	8	F	20	Diesel		
4	5	C	15	Unleaded	Toyota Camry	

Class Indicator Field:  
**V** = Vehicle  
**F** = Fueled Vehicle  
**N** = Non-Fueled Vehicle  
**C** = Car  
**B** = Bicycle

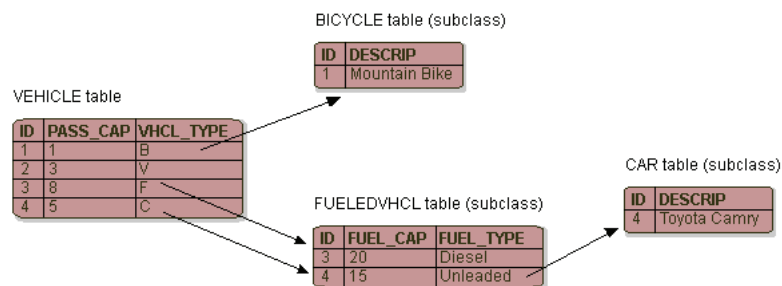
### 21.3.2.2 Multi-Table Inheritance

In this example, you store classes with multiple levels of inheritance in multiple tables to optimize database storage space.

In the inheritance hierarchy shown in Figure 21–1, "Nested Aggregates", for subclasses that require additional attributes, you use multiple tables instead of a single superclass table. This optimizes storage space because there are no unused fields in the database. However, this may affect performance because TopLink must read from more than one table before it can instantiate the object. TopLink first looks at the class indicator field (see Section 16.3.1, "How to Specify a Class Indicator") to determine the class of object to create, then uses the descriptor for that class to read from the subclass tables.

Figure 21–3 illustrates the TopLink implementation of the FUELEDVHCL, CAR, and BICYCLE tables. All objects are stored in the VEHICLE table. FueledVehicle, Car, and Bicycle information are also stored in secondary tables. Note that because the NonFueledVehicle class does not hold any attributes or relationships, it does not need a secondary table.

**Figure 21–3 Inheritance Using Separate Tables for Each Subclass**



**Note:** In general, using multitable inheritance is inefficient because it can require excessive joins and multiple table fetching.

**21.3.2.2.1 Inheritance Outer-Joins** If a root or branch inheritance descriptor has subclasses that span multiple tables, you can configure a database view to optimize the performance of queries against the parent descriptor by outer-joining all of the subclass tables. This allows TopLink to fetch all of the subclass instances in one query, instead of multiple queries. It also allows queries for the parent class that use cursors or ordering.

By default, TopLink executes multiple queries to read in a multiple table inheritance hierarchy, which, in some cases, is the most efficient way to query. In addition,

TopLink supports querying the inheritance hierarchy using a single outer-join query ([Section 119.19, "Configuring Reading Subclasses on Queries"](#)).

You can also set a database view on the descriptor that outer-joins or unions all of the tables. For more information, see [Section 23.7, "Configuring Multitable Information"](#).



---

---

## Creating a Relational Descriptor

This chapter describes how to create relational descriptors.

This chapter includes the following sections:

- [Introduction to Relational Descriptor Creation](#)
- [Creating a Relational Descriptor](#)

For information on how to create more than one type of descriptors, see [Chapter 118](#), "Creating a Descriptor".

### 22.1 Introduction to Relational Descriptor Creation

After you create a descriptor, you must configure its various options (see [Chapter 119](#), "Configuring a Descriptor") and use it to define mappings.

For complete information on the various types of mapping that TopLink supports, see [Chapter 17](#), "Introduction to Mappings" and [Chapter 120](#), "Creating a Mapping".

For complete information on the various types of descriptor that TopLink supports, see [Section 16.1](#), "Descriptor Types".

For more information, see [Chapter 21](#), "Introduction to Relational Descriptors".

### 22.2 Creating a Relational Descriptor

You can create a relational descriptor using Oracle JDeveloper, TopLink Workbench (see [Section 22.2.1](#), "How to Create a Relational Descriptor Using TopLink Workbench"), or Java code (see [Section 22.2.2](#), "How to Create a Relational Descriptor Using Java").

#### 22.2.1 How to Create a Relational Descriptor Using TopLink Workbench

Using TopLink Workbench, you can create the following types of descriptor in a relational project:

- Relational class descriptors (see [Section 22.2.1.1](#), "Creating Relational Class Descriptors");
- Relational aggregate descriptors (see [Section 22.2.1.2](#), "Creating Relational Aggregate Descriptors");
- Relational interface descriptors (see [Section 22.2.1.3](#), "Creating Relational Interface Descriptors").

### 22.2.1.1 Creating Relational Class Descriptors



By default, when you add a Java class to a relational project (see [Section 117.3, "Configuring Project Classpath"](#)), TopLink Workbench creates a relational class descriptor for it. A class descriptor is applicable to any persistent object except an object that is owned by another in an aggregate relationship. In this case, you must describe the owned object with an aggregate descriptor (see [Section 22.2.1.2, "Creating Relational Aggregate Descriptors"](#)). Using a class descriptor, you can configure any relational mapping except aggregate collection and aggregate object mappings.

### 22.2.1.2 Creating Relational Aggregate Descriptors



An aggregate object is an object that is strictly dependent on its owning object. Aggregate descriptors do not define a table, primary key, or many of the standard descriptor options as they obtain these from their owning descriptor. If you want to configure an aggregate mapping to associate data members in a target object with fields in a source object's underlying database tables (see [Chapter 35, "Configuring a Relational Aggregate Collection Mapping"](#) and [Chapter 37, "Configuring a Relational Aggregate Object Mapping"](#)), you must designate the target object's descriptor as an aggregate (see [Section 23.6, "Configuring a Relational Descriptor as a Class or Aggregate Type"](#)).

### 22.2.1.3 Creating Relational Interface Descriptors



If you add an interface to a relational project (see [Section 117.3, "Configuring Project Classpath"](#)), TopLink Workbench creates an interface descriptor for it.

An interface is a collection of abstract behavior that other classes can use. It is a purely Java concept and has no representation on the relational database. Therefore, a descriptor defined for the interfaces does not map any relational entities on the database.

The interface descriptor includes the following elements:

- The Java interface it describes.
- The parent interface(s) it implements.
- A list of abstract query keys.

An interface descriptor does not define any mappings, because there is no concrete data or table associated with it. A list of abstract query keys is defined so that you can issue queries on the interfaces (see [Section 119.11, "Configuring Interface Query Keys"](#)). A read query on the interface results in reading one or more of its implementors.

## 22.2.2 How to Create a Relational Descriptor Using Java

[Example 22-1](#) shows how to create a relational descriptor using Java code.

### **Example 22-1** Creating a Relational Descriptor in Java

```
RelationalDescriptor descriptor = new RelationalDescriptor();
descriptor.setJavaClass(YourClass.class);
```

To designate a relational descriptor as an aggregate, use `ClassDescriptor` method `descriptorIsAggregate`. For a `RelationalDescriptor` configured as an aggregate, you do not define a primary key, but when using Java, you must configure the associated table (see [Section 23.2, "Configuring Associated Tables"](#)) and field mappings (see [Chapter 17, "Introduction to Mappings"](#)).

To allow a relational descriptor to participate in an aggregate collection mapping (see [Section 27.9, "Aggregate Collection Mapping"](#)), use `ClassDescriptor` method

`descriptorIsAggregateCollection`. For a `RelationalDescriptor` configured for use with an aggregate collection mapping, you do define primary keys (see [Section 119.2, "Configuring Primary Keys"](#)) and an associated table (see [Section 23.2, "Configuring Associated Tables"](#)), but you do not have to map the primary keys if they are shared from their parent.

To configure a relational descriptor for an interface, use `ClassDescriptor` method `setJavaInterface`, passing in the `java.lang.Class` of the interface. You should only use an interface descriptor for an interface that has multiple implementors. If an interface has only a single implementor, then rather than creating an interface descriptor, just set the implementor descriptor's interface alias (see [Section 23.5, "Configuring Interface Alias"](#)).



## Configuring a Relational Descriptor

This chapter describes how to configure a relational descriptor.

This chapter contains the following sections:

- [Introduction to Relational Descriptor Configuration](#)
- [Configuring Associated Tables](#)
- [Configuring Sequencing at the Descriptor Level](#)
- [Configuring Custom SQL Queries for Basic Persistence Operations](#)
- [Configuring Interface Alias](#)
- [Configuring a Relational Descriptor as a Class or Aggregate Type](#)
- [Configuring Multitable Information](#)

For information on how to configure descriptor options common to two or more descriptor types, see [Chapter 119, "Configuring a Descriptor"](#).

### 23.1 Introduction to Relational Descriptor Configuration

[Table 23–1](#) lists the default configurable options for a relational descriptor.

**Table 23–1** Configurable Options for Relational Descriptor

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Associated tables (see <a href="#">Section 23.2, "Configuring Associated Tables"</a> )	✓	✓	✓
Primary keys (see <a href="#">Section 119.2, "Configuring Primary Keys"</a> )	✓	✓	✓
Sequencing (see <a href="#">Section 23.3, "Configuring Sequencing at the Descriptor Level"</a> )	✓	✓	✓
Read-only descriptors (see <a href="#">Section 119.3, "Configuring Read-Only Descriptors"</a> )	✓	✓	✓
Unit of work conforming (see <a href="#">Section 119.4, "Configuring Unit of Work Conforming at the Descriptor Level"</a> )	✓	✓	✓
Descriptor alias (see <a href="#">Section 119.5, "Configuring Descriptor Alias"</a> )	✓	✓	
Descriptor comments (see <a href="#">Section 119.6, "Configuring Descriptor Comments"</a> )	✓	✓	
Classes (see <a href="#">Section 5.7.2, "How to Configure Classes"</a> )		✓	
Named queries (see <a href="#">Section 119.7, "Configuring Named Queries at the Descriptor Level"</a> )	✓	✓	✓

**Table 23–1 (Cont.) Configurable Options for Relational Descriptor**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Custom SQL queries for basic persistence operations (see Section 23.4, "Configuring Custom SQL Queries for Basic Persistence Operations")	✓	✓	✓
Query timeout (see Section 119.8, "Configuring Query Timeout at the Descriptor Level")	✓	✓	✓
Cache refreshing (see Section 119.9, "Configuring Cache Refreshing")	✓	✓	✓
Query keys (see Section 119.10, "Configuring Query Keys")	✓	✓	✓
Interface query keys (see Section 119.11, "Configuring Interface Query Keys")	✓	✓	✓
Interface alias (see Section 23.5, "Configuring Interface Alias")	✓	✓	✓
Cache type and size (see Section 119.12, "Configuring Cache Type and Size at the Descriptor Level")	✓	✓	✓
Cache isolation (see Section 119.13, "Configuring Cache Isolation at the Descriptor Level")	✓	✓	✓
Cache coordination change propagation (see Section 119.15, "Configuring Cache Coordination Change Propagation at the Descriptor Level")	✓	✓	✓
Cache expiration (see Section 119.16, "Configuring Cache Expiration at the Descriptor Level")	✓	✓	✓
Cache existence Checking (see Section 119.17, "Configuring Cache Existence Checking at the Descriptor Level")	✓	✓	✓
EJB information (see Section 119.18, "Configuring a Descriptor with EJB CMP and BMP Information")	✓	✓	✓
Relational descriptor as a class or aggregate type (see Section 23.6, "Configuring a Relational Descriptor as a Class or Aggregate Type")	✓	✓	✓
Reading subclasses on queries (see Section 119.19, "Configuring Reading Subclasses on Queries")	✓	✓	✓
Inheritance for a child class descriptor (see Section 119.20, "Configuring Inheritance for a Child (Branch or Leaf) Class Descriptor")	✓	✓	✓
Inheritance for a parent class descriptor (see Section 119.21, "Configuring Inheritance for a Parent (Root) Descriptor")	✓	✓	✓
Inheritance expressions for a parent class descriptor (see Section 119.22, "Configuring Inheritance Expressions for a Parent (Root) Class Descriptor")			✓
Inherited attribute mapping in a subclass (see Section 119.23, "Configuring Inherited Attribute Mapping in a Subclass")	✓	✓	✓
Multitable information (see Section 23.7, "Configuring Multitable Information")	✓	✓	✓
Domain object method as an event handler (see Section 119.24, "Configuring a Domain Object Method as an Event Handler")	✓	✓	✓
Descriptor event listener as an event handler (see Section 119.25, "Configuring a Descriptor Event Listener as an Event Handler")	✓	✓	✓
Locking policy (see Section 119.26, "Configuring Locking Policy")	✓	✓	✓
Returning policy (see Section 119.27, "Configuring Returning Policy")	✓	✓	✓
Instantiation policy (see Section 119.28, "Configuring Instantiation Policy")	✓	✓	✓
Copy policy (see Section 119.29, "Configuring Copy Policy")	✓	✓	✓
Change policy (see Section 119.30, "Configuring Change Policy")			✓

**Table 23–1 (Cont.) Configurable Options for Relational Descriptor**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
History policy (see Section 119.31, "Configuring a History Policy")			✓
Wrapper policy (see Section 119.32, "Configuring Wrapper Policy")			✓
Fetch groups (see Section 119.33, "Configuring Fetch Groups")			✓
Amendment methods (see Section 119.35, "Configuring Amendment Methods")	✓	✓	
Mapping (see Section 121, "Configuring a Mapping")	✓	✓	✓

For more information, see Chapter 21, "Introduction to Relational Descriptors".

## 23.2 Configuring Associated Tables

Each relational class descriptor (see Section 22.2.1.1, "Creating Relational Class Descriptors") must be associated with a database table for storing instances of that class. This does not apply to relational aggregate descriptors (see Section 22.2.1.2, "Creating Relational Aggregate Descriptors").

### 23.2.1 How to Configure Associated Tables Using TopLink Workbench

To associate a descriptor with a database table, use this procedure:

1. Select a descriptor in the **Navigator**. Its properties appear in the Editor.
2. Click the **Descriptor Info** tab. The Descriptor Info tab appears.

**Figure 23–1 Descriptor Info Tab, Associated Table Options**

Use the **Associated Table** list to select a database table for the descriptor. You must associate a descriptor with a database table *before* specifying primary keys.

### 23.2.2 How to Configure Associated Tables Using Java

To configure a descriptor's associated table(s) using Java, use `RelationalDescriptor` methods `setTableName` or `addTableName`.

## 23.3 Configuring Sequencing at the Descriptor Level

Sequencing allows TopLink to automatically assign the primary key or ID of an object when the object is inserted.

You configure TopLink sequencing at the project level (Section 20.3, "Configuring Sequencing at the Project Level") or session level (see Section 98.4, "Configuring Sequencing at the Session Level") to tell TopLink how to obtain sequence values: that is, what type of sequences to use.

To enable sequencing, you must then configure TopLink sequencing at the descriptor level to tell TopLink into which table and column to write the sequence value when an instance of a descriptor's reference class is created.

Only descriptors that have been configured with a sequence field and a sequence name will be assigned sequence numbers.

The sequence field is the database field that the sequence number will be assigned to: this is almost always the primary key field (see Section 119.2, "Configuring Primary Keys"). The sequence name is the name of the sequence to be used for this descriptor. The purpose of the sequence name depends on the type of sequencing you are using:

When using table sequencing, the sequence name refers to the row's `SEQ_NAME` value used to store this sequence.

When using Oracle native sequencing, the sequence name refers to the Oracle sequence object that has been created in the database. When using native sequencing on other databases, the sequence name does not have any direct meaning, but should still be set for compatibility.

The sequence name can also refer to a custom sequence defined in the project.

For more information, see Section 18.2, "Sequencing in Relational Projects".

### 23.3.1 How to Configure Sequencing at the Descriptor Level Using TopLink Workbench

To configure sequencing for a descriptor, use this procedure:

1. Select a descriptor in the **Navigator**. Its properties appear in the Editor.
2. Click the **Descriptor Info** tab. The Descriptor Info tab appears.



Figure 23–2 Descriptor Info Tab, Sequencing Options

The screenshot shows the 'Descriptor Info' tab with the following fields and options:

- Associated Table: ADDRESS
- Primary Keys: ADDRESS.ADDRESS\_ID (with Add... and Remove buttons)
- Use Sequencing: (highlighted with a red circle)
- Name: [text input]
- Table: <none selected>
- Field: <none selected>
- Read-Only
- Conform Results in Unit of Work
- Descriptor Alias: Address
- Comment: [text input]

Use the following information to specify sequencing options:

Field	Description
Use Sequencing	Specify if this descriptor uses sequencing. If selected, specify the Name, Table, and Field for sequencing.
Name	Enter the name of the sequence. <ul style="list-style-type: none"> <li>■ <b>For table sequencing:</b> Enter the name of the value in the sequence name column (for default table sequencing, the column named SEQ_NAME) of the sequence table (for default table sequencing, the table named SEQUENCE) that TopLink uses to look up the corresponding sequence count value (for default table sequencing, the corresponding value in the SEQ_COUNT column) for this descriptor's reference class. For more information, see <a href="#">Section 18.2.2.1, "Table Sequencing"</a>.</li> <li>■ <b>For native sequencing (Oracle platform):</b> Enter the name of the sequence object that Oracle Database creates to manage sequencing for this descriptor's reference class. For more information, see <a href="#">Section 18.2.2.5, "Native Sequencing with an Oracle Database Platform"</a></li> <li>■ <b>For native sequencing (non-Oracle platform):</b> For database compatibility, enter a generic name for the sequence, such as SEQ. For more information, see <a href="#">Section 18.2.2.6, "Native Sequencing with a Non-Oracle Database Platform"</a>.</li> </ul>
Table	Specify the name of the database table that contains the field (see <b>Field</b> ) into which TopLink is to write the sequence value when a new instance of this descriptor's reference class is created. This is almost always this descriptor's primary table.
Field	Specify the name of the field in the specified table (see <b>Table</b> ) into which TopLink is to write the sequence value when a new instance of this descriptor's reference class is created. This field is almost always the class's primary key (see <a href="#">Section 119.2, "Configuring Primary Keys"</a> ). <ul style="list-style-type: none"> <li>■ <b>For native sequencing (non-Oracle platform):</b> Ensure that your database schema specifies the correct type for this field (see <a href="#">Section 18.2.2.6, "Native Sequencing with a Non-Oracle Database Platform"</a>).</li> </ul>

## 23.3.2 How to Configure Sequencing at the Descriptor Level Using Java

Using Java, you can configure sequencing to use multiple different types of sequence for different descriptors. You configure the sequence objects on the session's login and reference them from the descriptor by their name. The descriptor's sequence name refers to the sequence object's name you register in the session's login.

The following examples assume the session sequence configuration shown in [Example 23–1](#).

### **Example 23–1 Example Sequences**

```
dbLogin.addSequence(new TableSequence("EMP_SEQ", 25));
dbLogin.addSequence(new DefaultSequence("PHONE_SEQ", 30));
dbLogin.addSequence(new UnaryTableSequence("ADD_SEQ", 55));
dbLogin.addSequence(new NativeSequence("NAT_SEQ", 10));
```

Using Java code, you can perform the following sequence configurations:

- [Configuring a Sequence by Name](#)
- [Configuring the Same Sequence for Multiple Descriptors](#)
- [Configuring the Platform Default Sequence](#)

### **23.3.2.1 Configuring a Sequence by Name**

As [Example 23–2](#) shows, you associate a sequence with a descriptor by sequence name. The sequence EMP\_SEQ was added to the login for this project in [Example 23–1](#). When a new instance of the Employee class is created, the TopLink runtime will use the sequence named EMP\_SEQ (in this example, a TableSequence) to obtain a value for the EMP\_ID field.

### **Example 23–2 Associating a Sequence with a Descriptor**

```
empDescriptor.setSequenceNumberFieldName("EMP_ID"); // primary key field
empDescriptor.setSequenceNumberName("EMP_SEQ");
```

### **23.3.2.2 Configuring the Same Sequence for Multiple Descriptors**

As [Example 23–3](#) shows, you can associate the same sequence with more than one descriptor. In this example, both the Employee descriptor and Phone descriptor use the same NativeSequence. Having descriptors share the same sequence can improve pre-allocation performance. For more information on pre-allocation, see [Section 18.2.3, "Sequencing and Preallocation Size"](#).

### **Example 23–3 Configuring a Sequence for Multiple Descriptors**

```
empDescriptor.setSequenceNumberFieldName("EMP_ID"); // primary key field
empDescriptor.setSequenceNumberName("NAT_SEQ");
phoneDescriptor.setSequenceNumberFieldName("PHONE_ID"); // primary key field
phoneDescriptor.setSequenceNumberName("NAT_SEQ");
```

### **23.3.2.3 Configuring the Platform Default Sequence**

In [Example 23–4](#), you associate a nonexistent sequence (NEW\_SEQ) with a descriptor. Because you did not add a sequence named NEW\_SEQ to the login for this project in [Example 23–1](#), the TopLink runtime will create a DefaultSequence named NEW\_SEQ for this descriptor. For more information about DefaultSequence, see [Section 18.2.2.4, "Default Sequencing"](#).

**Example 23–4 Configuring a Default Sequence**

```
descriptor.setSequenceNumberFieldName("EMP_ID"); // primary key field
descriptor.setSequenceNumberName("NEW_SEQ");
```

## 23.4 Configuring Custom SQL Queries for Basic Persistence Operations

You can use TopLink to define an SQL query for each basic persistence operation (insert, update, delete, read-object, read-all, or does-exist) so that when you query and modify your relational-mapped objects, the TopLink runtime will use the appropriate SQL query instead of the default SQL query.

SQL strings can include any fields that the descriptor maps, as well as arguments. You specify arguments in the SQL string using #<arg-name>, such as:

```
select * from EMP where EMP_ID = #EMP_ID
```

The insert and update SQL strings can take any field that the descriptor maps as an argument.

The read-object, delete and does-exist SQL strings can only take the primary key fields as arguments.

The read-all SQL string must return all instances of the class and thus can take no arguments.

You can define a custom SQL string for insert, update, delete, read-object, and read-all using Oracle JDeveloper TopLink Editor or TopLink Workbench (see [Section 23.4.1, "How to Configure Custom SQL Queries for Basic Persistence Operations Using TopLink Workbench"](#)).

You can define a custom SQL string or Call object for insert, update, delete, read-object, read-all, and does-exist using Java (see [Section 23.4.2, "How to Configure Custom SQL Queries for Basic Persistence Operations Using Java"](#)). Using a Call, you can define more complex SQL strings and invoke custom stored procedures.

For CMP projects, the `ejb-jar.xml` file stores query lists. You can define the queries in the file and then read them into Oracle JDeveloper or TopLink Workbench, or define them on the **Queries** tab of TopLink Workbench and write them to the file (see [Section 19.7, "Working with the ejb-xml.File"](#)).

---

**Note:** When you customize the update persistence operation for an application that uses optimistic locking (see [Section 119.26, "Configuring Locking Policy"](#)), the custom update string must not write the object if the row version field has changed since the initial object was read. In addition, it must increment the version field if it writes the object successfully.

For example:

```
update Employee set F_NAME = #F_NAME, VERSION = VERSION + 1
where (EMP_ID = #EMP_ID) AND (VERSION = #VERSION)
```

The update string must also maintain the row count of the database.

---



---

**Note:** TopLink does not validate the SQL code that you enter. Enter the SQL code appropriate for your database platform (see [Section 96.1.3, "Data Source Platform Types"](#)).

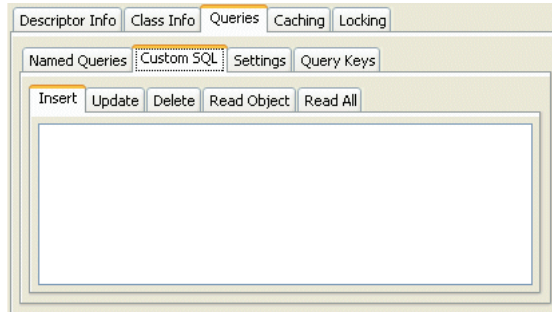
---

## 23.4.1 How to Configure Custom SQL Queries for Basic Persistence Operations Using TopLink Workbench

To configure custom SQL queries for basic persistence operations:

1. In the **Navigator**, select a descriptor in a relational database project.
2. Click the **Queries** tab in the **Editor**.
3. Click the **Custom SQL** tab.

**Figure 23–3 Queries, Custom SQL Tab**



Click the appropriate SQL function tab and type your own SQL string to control these actions for a descriptor. Use the following information to complete the tab:

Tab	Description
<b>Insert</b>	Defines the insert SQL that TopLink uses to insert a new object's data into the database.
<b>Update</b>	<p>Defines the update SQL that TopLink uses to update any changed existing object's data in the database.</p> <p>When you define a descriptor's <b>update</b> query, you must conform to the following:</p> <ul style="list-style-type: none"> <li>■ If the application uses optimistic locking, you must ensure that the row is not written if the version field has changed since the object was read.</li> <li>■ The update query must increment the version field if the row is written.</li> <li>■ The update string must maintain the row count of the database.</li> </ul>
<b>Delete</b>	Defines the delete SQL that TopLink uses to delete an object.
<b>Read Object</b>	<p>Defines the read SQL that TopLink uses in any <code>ReadObjectQuery</code>, whose selection criteria is based on the object's primary key.</p> <p>When you define a descriptor's <b>read-object</b> query, your implementation overrides any <code>ReadObjectQuery</code>, whose selection criteria is based on the object's primary key. TopLink generates dynamic SQL for all other <code>Session readObject</code> method signatures.</p> <p>To customize other <code>Session readObject</code> method signatures, define additional named queries and use them in your application instead of the <code>Session</code> methods.</p>

Tab	Description
Read All	<p>Defines the read-all SQL that TopLink uses when you call <code>Session</code> method <code>readAllObjects(java.lang.Class)</code> passing in the <code>java.lang.Class</code> that this descriptor represents.</p> <p>When you define a descriptor's <b>read-all</b> query, your implementation overrides only the <code>Session</code> method <code>readAll(java.lang.Class)</code>, not the version that takes a <code>Class</code> and <code>Expression</code>. As a result, this query reads every single instance. TopLink generates dynamic SQL for all other <code>Session readAll</code> method signatures.</p> <p>To customize other <code>Session readAll</code> method signatures, define additional named queries and use them in your application instead of the <code>Session</code> methods.</p>

## 23.4.2 How to Configure Custom SQL Queries for Basic Persistence Operations Using Java

The `DescriptorQueryManager` generates default SQL for the following persistence operations:

- Insert
- Update
- Delete
- Read-object
- Read-all
- Does-exist

Using Java code, you can use the descriptor query manager to provide custom SQL strings to perform these functions on a class-by-class basis.

Use `ClassDescriptor` method `getQueryManager` to acquire the `DescriptorQueryManager`, and then use the `DescriptorQueryManager` methods that [Table 23–2](#) lists.

**Table 23–2** *Descriptor Query Manager Methods for Configuring Custom SQL*

To Change the Default SQL for...	Use Descriptor Query Manager Method...
Insert	<code>setInsertQuery (InsertObjectQuery query)</code>
	<code>setInsertSQLString (String sqlString)</code>
	<code>setInsertCall(Call call)</code>
Update	<code>setUpdateQuery (UpdateObjectQuery query)</code>
	<code>setUpdateSQLString (String sqlString)</code>
	<code>setUpdateCall(Call call)</code>
Delete	<code>setDeleteQuery (DeleteObjectQuery query)</code>
	<code>setDeleteSQLString (String sqlString)</code>
	<code>setDeleteCall(Call call)</code>
Read	<code>setReadObjectQuery (ReadObjectQuery query)</code>

**Table 23–2 (Cont.) Descriptor Query Manager Methods for Configuring Custom SQL**

To Change the Default SQL for...	Use Descriptor Query Manager Method...
	<code>setReadObjectSQLString (String sqlString)</code>
	<code>setReadObjectCall(Call call)</code>
Read all	<code>setReadAllQuery (ReadAllQuery query)</code>
	<code>setReadAllSQLString (String sqlString)</code>
	<code>setReadAllCall(Call call)</code>
Does exist	<code>setDoesExistQuery(DoesExistQuery query)</code>
	<code>setDoesExistSQLString(String sqlString)</code>
	<code>setDoesExistCall(Call call)</code>

[Example 23–5](#) shows how to implement an amendment method to configure a descriptor query manager to use custom SQL strings. Alternatively, using an `SQLCall`, you can specify more complex SQL strings using features such as `in`, `out`, and `in-out` parameters and parameter types (see [Section 109.4, "Using a SQLCall"](#)).

**Example 23–5 Configuring a Descriptor Query Manager with Custom SQL Strings**

```
public static void addToDescriptor(ClassDescriptor descriptor) {

    // Read-object by primary key procedure
    descriptor.getQueryManager().setReadObjectSQLString(
        "select * from EMP where EMP_ID = #EMP_ID");

    // Read-all instances procedure
    descriptor.getQueryManager().setReadAllSQLString(
        "select * from EMP");

    // Insert procedure
    descriptor.getQueryManager().setInsertSQLString(
        "insert into EMP (EMP_ID, F_NAME, L_NAME, MGR_ID) values
        (#EMP_ID, #F_NAME, #L_NAME, #MGR_ID)");

    // Update procedure
    descriptor.getQueryManager().setUpdateSQLString(
        "update EMP set (F_NAME, L_NAME, MGR_ID) values
        (#F_NAME, #L_NAME, #MGR_ID) where EMP_ID = #EMP_ID");
}
```

[Example 23–6](#) shows how to implement an amendment method to configure a descriptor query manager to use Oracle stored procedures using a `StoredProcedureCall` (see [Section 109.5, "Using a StoredProcedureCall"](#)). This example uses output cursors to return the result set (see [Section 111.11, "Handling Cursor and Stream Query Results"](#)).

**Example 23–6 Configuring a Descriptor Query Manager with Custom Stored Procedure Calls**

```
public static void addToDescriptor(ClassDescriptor descriptor) {

    // Read-object by primary key procedure
    StoredProcedureCall readCall = new StoredProcedureCall();
```

```

readCall.setProcedureName("READ_EMP");
readCall.addNamedArgument("P_EMP_ID", "EMP_ID");
readCall.useNamedCursorOutputAsResultSet("RESULT_CURSOR");
descriptor.getQueryManager().setReadObjectCall(readCall);

// Read-all instances procedure
StoredProcedureCall readAllCall = new StoredProcedureCall();
readAllCall.setProcedureName("READ_ALL_EMP");
readAllCall.useNamedCursorOutputAsResultSet("RESULT_CURSOR");
descriptor.getQueryManager().setReadAllCall(readAllCall);

// Insert procedure
StoredProcedureCall insertCall = new StoredProcedureCall();
insertCall.setProcedureName("INSERT_EMP");
insertCall.addNamedArgument("P_EMP_ID", "EMP_ID");
insertCall.addNamedArgument("P_F_NAME", "F_NAME");
insertCall.addNamedArgument("P_L_NAME", "L_NAME");
insertCall.addNamedArgument("P_MGR_ID", "MGR_ID");
descriptor.getQueryManager().setInsertCall(insertCall);

// Update procedure
StoredProcedureCall updateCall = new StoredProcedureCall();
updateCall.setProcedureName("UPDATE_EMP");
updateCall.addNamedArgument("P_EMP_ID", "EMP_ID");
updateCall.addNamedArgument("P_F_NAME", "F_NAME");
updateCall.addNamedArgument("P_L_NAME", "L_NAME");
updateCall.addNamedArgument("P_MGR_ID", "MGR_ID");
descriptor.getQueryManager().setUpdateCall(updateCall);
}

```

## 23.5 Configuring Interface Alias

An interface alias allows an interface to be used to refer to a descriptor instead of the implementation class. This can be useful for classes that have public interface and the applications desire to refer to the class using the public interface. Specifying the interface alias allows any queries executed on a TopLink session to use the interface as the reference class instead of the implementation class.

Each descriptor can have one interface alias. Use the interface in queries and relationship mappings.

---

**Note:** If you use an interface alias, do not associate an interface descriptor with the interface.

---

This section includes information on configuring an interface alias. Interfaces cannot be *created* in Oracle JDeveloper or TopLink Workbench; you must add the Java package or class to your Oracle JDeveloper or TopLink Workbench project before configuring it.

### 23.5.1 How to Configure Interface Alias Using TopLink Workbench

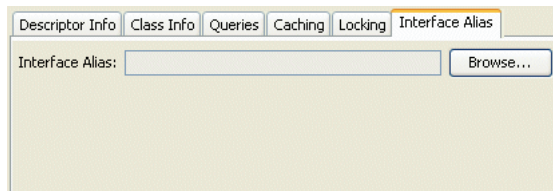
To specify an interface alias, use this procedure:

1. In the **Navigator**, select a descriptor.

If the **Interface Alias** advanced property is not visible for the descriptor, right-click the descriptor and choose **Select Advanced Properties > Interface Alias** from context menu or from the Selected menu.

2. Click the **Interface Alias** tab.

**Figure 23–4 Interface Alias Tab**



In the **Interface Alias** field, click **Browse** and select an interface.

## 23.5.2 How to Configure Interface Alias Using Java

To configure a descriptor with an interface alias using Java, create an amendment method (see [Section 119.35, "Configuring Amendment Methods"](#)) and use `InterfacePolicy` method `addParentInterface` as [Example 23–7](#) shows.

**Example 23–7 Configuring an Interface Alias**

```
public static void addToDescriptor(Descriptor descriptor) {
    descriptor.getInterfacePolicy().addParentInterface(MyInterface.class);
}
```

## 23.6 Configuring a Relational Descriptor as a Class or Aggregate Type

By default, when you add a Java class to a relational project (see [Section 117.3, "Configuring Project Classpath"](#)), Oracle JDeveloper or TopLink Workbench create a relational class descriptor for it. A class descriptor is applicable to any persistent object except an object that is owned by another in an aggregate relationship. In this case, you must describe the owned object with an aggregate descriptor. Using a class descriptor, you can configure any relational mapping except aggregate collection and aggregate object mappings.

An aggregate object is an object that is strictly dependent on its owning object. Aggregate descriptors do not define a table, primary key, or many of the standard descriptor options as they obtain these from their owning descriptor. If you want to configure an aggregate mapping to associate data members in a target object with fields in a source object's underlying database tables (see [Chapter 35, "Configuring a Relational Aggregate Collection Mapping"](#) and [Chapter 37, "Configuring a Relational Aggregate Object Mapping"](#)), you must designate the target object's descriptor as an aggregate.

Alternatively, you can remove the aggregate designation from a relational descriptor and return it to its default type.

You can configure inheritance for a descriptor designated as an aggregate (see [Section 119.20, "Configuring Inheritance for a Child \(Branch or Leaf\) Class Descriptor"](#)), however, in this case, *all* the descriptors in the inheritance tree must be aggregates. Aggregate and class descriptors cannot exist in the same inheritance tree. For more information, see [Section 16.3.4, "Aggregate and Composite Descriptors and Inheritance"](#).

If you configure a descriptor as an aggregate, you cannot configure the descriptor with EJB information (see [Section 119.18, "Configuring a Descriptor with EJB CMP and BMP Information"](#)).



For more information, see [Section 50.1.1, "XML Descriptors and Aggregation"](#).

### 23.6.1 How to Configure a Relational Descriptor as a Class or Aggregate Type Using TopLink Workbench

To configure a relational descriptor as class or aggregate, use this procedure.

1. In the **Navigator**, select a relational descriptor.
2. Click the **Class** or **Aggregate** descriptor button on the mapping toolbar.

You can also select the descriptor and choose **Selected > Descriptor Type > Class** or **Aggregate** from the menu or by right-clicking on the descriptor in the **Navigator** window and selecting **Descriptor Type > Class** or **Aggregate** from the context menu.



3. If you select **Aggregate**, specify each of the aggregate descriptor's attributes as a direct to field mapping. See [Chapter 29, "Configuring a Relational Direct-to-Field Mapping"](#) for more information.



Specify each of the aggregate descriptor's attributes as a direct to field mapping. See [Chapter 29, "Configuring a Relational Direct-to-Field Mapping"](#) for more information.

Although the attributes of a target class are not mapped directly to a data source until you configure an aggregate object mapping, you must still specify their mapping type in the target class's descriptor. This tells TopLink what type of mapping to use when you do configure the aggregate mapping in the source object's descriptor. For more information, see [Section 21.2, "Aggregate and Composite Descriptors in Relational Projects"](#).

### 23.6.2 How to Configure a Relational Descriptor as a Class or Aggregate Type Using Java

Using Java, to configure a relational descriptor as an aggregate, use `ClassDescriptor` method `descriptorIsAggregate`.

To configure a relational descriptor for use in an aggregate collection mapping, use `ClassDescriptor` method `descriptorIsAggregateCollection`.

To configure a relational descriptor as a nonaggregate, use `ClassDescriptor` method `descriptorIsNormal`.

## 23.7 Configuring Multitable Information

Descriptors can use multiple tables in mappings. Use multiple tables when either of the following occurs:

- A subclass is involved in inheritance, and its superclass is mapped to one table, while the subclass has additional attributes that are mapped to a second table.
- A class is not involved in inheritance and its data is spread out across multiple tables.

When a descriptor has multiple tables, you must be able to join a row from the primary table to all the additional tables. By default, TopLink assumes that the primary key of the first, or primary, table is included in the additional tables, thereby joining the tables. TopLink also supports custom methods for joining tables. If the primary key field names of the multiple tables do not match, a foreign key can be used to join the tables. The foreign key can either be from the primary table to the secondary table, or from the secondary table to the primary table, or between two of the

secondary tables (see [Section 23.7.1, "How to Configure Multitable Information Using TopLink Workbench"](#)).

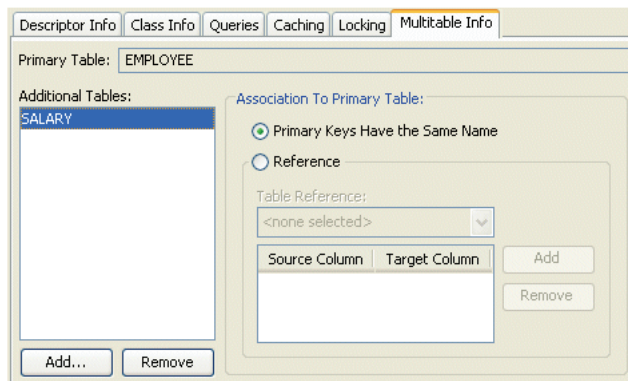
For complex multitable situations, a more complex join expression may be required. These include requiring the join to also check a type code, or using an outer-join. TopLink provides support for a multiple-table-join-expression for these cases (see [Section 23.7.2, "How to Configure Multitable Information Using Java"](#)).

### 23.7.1 How to Configure Multitable Information Using TopLink Workbench

To associate multiple tables with a descriptor, use this procedure.

1. In the **Navigator**, select a descriptor.  
 If the **Multitable Info** advanced property is not visible for the descriptor, right-click the descriptor and choose **Select Advanced Properties > Multitable Info** from the context menu or from the **Selected** menu.
2. Click the **Multitable Info** tab.

**Figure 23–5 Multitable Info Tab**

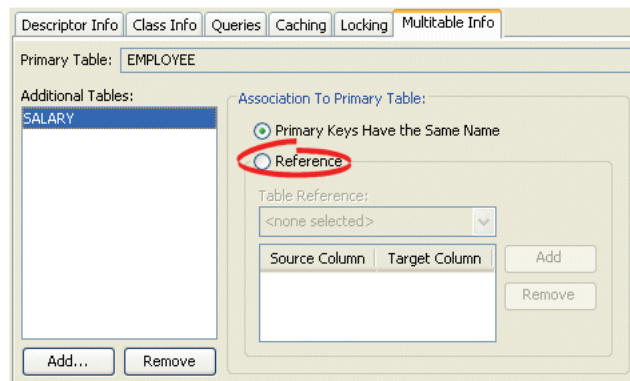


Use the following information to enter data in each field of the tab:

Field	Description
<b>Primary Table</b>	The primary table for this descriptor. This field is for display only.
<b>Additional Tables</b>	Use <b>Add</b> and <b>Remove</b> to add or remove additional tables.
<b>Association to Primary Table</b>	Specify how each <b>Additional Table</b> is associated to the <b>Primary Table</b> : <ul style="list-style-type: none"> <li>▪ <b>Primary Keys Have Same Names</b>—when associating tables by identically named primary keys, TopLink requires no additional configuration.</li> <li>▪ <b>Reference</b>—when associating an additional table to the primary table with a <b>Reference</b> (that is, a foreign key), you can specify the <b>Table Reference</b>, as well as the <b>Source</b> and <b>Target</b> fields. Continue with <a href="#">Section , "Associating Tables with References"</a>.</li> </ul>

#### Associating Tables with References

When associating a table using **Reference**, additional options appear. You must choose a reference that relates the correct fields in the primary table to the primary keys in the selected table.

**Figure 23–6 Multitable Info Tab, Associated by Reference**

Choose a **Table Reference** that defines how the primary keys of the primary table relate to the primary keys of the selected table. Click **Add** to add a primary key association.

### 23.7.2 How to Configure Multitable Information Using Java

Using Java, configure a descriptor with multitable information using the following `oracle.toplink.descriptors.ClassDescriptor` methods:

- `addTableName(java.lang.String tableName)`
- `addForeignKeyFieldNameForMultipleTable(java.lang.String sourceForeignKeyFieldName, java.lang.String targetPrimaryKeyFieldName)`

To specify a complex multiple-table-join-expression, create a descriptor amendment method (see [Section 119.35, "Configuring Amendment Methods"](#)) and add the join expression using `oracle.toplink.descriptors.DescriptorQueryManager` method `setMultipleTableJoinExpression`. For more information, see [Section 111.7, "Appending Additional Join Expressions"](#).



# Part XI

---

## Object-Relational Data Type Descriptors

This part contains general information about object-relational data type descriptors, as well as detailed information on how to create and configure these descriptors.

This part includes the following chapters:

- [Chapter 24, "Introduction to Object-Relational Data Type Descriptors"](#)  
This chapter introduces concepts of object-relational data type descriptors.
- [Chapter 25, "Creating an Object-Relational Data Type Descriptor"](#)  
This chapter explains how to create object-relational data type descriptors.
- [Chapter 26, "Configuring an Object-Relational Data Type Descriptor"](#)  
This chapter explains how to configure descriptor options specific to an object-relational data type descriptor.



---

---

## Introduction to Object-Relational Data Type Descriptors

This chapter provides an overview of object-relational data type descriptors.

This chapter includes the following section:

- [Object-Relational Data Type Descriptors](#)

For information on descriptor concepts and features common to more than one type of TopLink descriptors, see [Chapter 16, "Introduction to Descriptors"](#).

### 24.1 Object-Relational Data Type Descriptors

The object-relational data type paradigm extends traditional relational databases to include object-oriented functions. Oracle, IBM DB2, Informix, and other DBMS databases allow users to store, access, and use complex data in more sophisticated ways.

The object-relational data type standard is an evolving standard concerned mainly with extending the database data structures and SQL (SQL 3).

Object-relational data type descriptors describe Java objects that you map to special relational database types that correspond more closely to object types. Using these special object-relational data type database types can simplify mapping objects to relational database tables. Not all relational databases support these special object-relational data type database types.

Using object-relational data type descriptors in a relational project, you can configure object-relational data type mappings to these special object-relational data type database data types (see [Section 40.1, "Object-Relational Data Type Mapping Types"](#)).

For more information, see the following:

- [Section 25.2, "Creating an Object-Relational Data Type Descriptor"](#)
- [Chapter 26, "Configuring an Object-Relational Data Type Descriptor"](#)





---

# Creating an Object-Relational Data Type Descriptor

This chapter describes how to create relational and object-relational data type descriptors.

This chapter includes the following sections:

- [Introduction to Object-Relational Data Type Descriptor Creation](#)
- [Creating an Object-Relational Data Type Descriptor](#)

For information on how to create more than one type of descriptors, see [Chapter 118](#), "Creating a Descriptor".

## 25.1 Introduction to Object-Relational Data Type Descriptor Creation

After you create a descriptor, you must configure its various options (see [Chapter 119](#), "Configuring a Descriptor") and use it to define mappings.

For complete information on the various types of mapping that TopLink supports, see [Chapter 17](#), "Introduction to Mappings" and [Chapter 120](#), "Creating a Mapping".

For complete information on the various types of descriptor that TopLink supports, see [Section 16.1](#), "Descriptor Types".

For more information, see [Chapter 24](#), "Introduction to Object-Relational Data Type Descriptors".

## 25.2 Creating an Object-Relational Data Type Descriptor

You cannot create an object-relational data type descriptor using Oracle JDeveloper or TopLink Workbench: you must use Java code. For more information on creating descriptors in Java code, see the *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.

For more information, see [Section 24.1](#), "Object-Relational Data Type Descriptors".

### 25.2.1 How to Create an Object-Relational Data Type Descriptor Using Java

Use the `ObjectRelationalDescriptor` class to define an object-relational data type descriptor. This class extends `RelationalDescriptor` to add the following methods:

- `setStructureName`: call this method to set the name of the object-relational data type structure that represents the object class in the data source.

- `addFieldOrdering`: call this method repeatedly to define the order in which object attributes are persisted to the data source. This defines a field index that `TopLink` uses if your object-relational data type data source driver uses JDBC indexed arrays.

[Example 25–1](#) shows an `Employee` object that is mapped to an Oracle Database using its object-relational data type features.

**Example 25–1 Employee Class**

```
public class Employee {
    Long id;
    String firstName;
    String lastName;

    ...
}
```

[Example 25–2](#) shows the object-relational data type database type (`Employee_t`) created to model the `Employee` object within the database. Such an object-relational data type database type is also known as a structure. This example also shows how to create and populate a database table (called `department`) that stores instances of the `Employee_t` audio tape.

**Example 25–2 Employee Object-Relational Data Type Data Model**

```
CREATE TYPE EMPLOYEE_T AS OBJECT(ID NUMBER(10),
                                F_NAME VARCHAR2(100),
                                L_NAME VARCHAR2(100),) NOT FINAL;
CREATE TABLE EMPLOYEES OF EMPLOYEE_T;
```

[Example 25–3](#) shows how to code an object-relational data type descriptor in Java to describe the object-relational data type database type `Employee_t`.

**Example 25–3 Creating an Object-Relational Data Type Descriptor in Java**

```
import oracle.toplink.objectrelational.*;
...
ObjectRelationalDescriptor descriptor = new ObjectRelationalDescriptor();
descriptor.setJavaClass(Employee.class);
descriptor.setTableName("EMPLOYEES");
descriptor.setStructureName("EMPLOYEE_T");
descriptor.setPrimaryKeyFieldName("ID");
descriptor.addFieldOrdering("ID");
descriptor.addFieldOrdering("F_NAME");
descriptor.addFieldOrdering("L_NAME");
descriptor.addDirectMapping("id", "OBJECT_ID");
descriptor.addDirectMapping("firstName", "F_NAME");
descriptor.addDirectMapping("lastName", "L_NAME");
```

For more information on configuring object-relational data type descriptors, see [Chapter 26, "Configuring an Object-Relational Data Type Descriptor"](#).

For more information on the object-relational data type mappings that `TopLink` supports, see [Chapter 40, "Introduction to Object-Relational Data Type Mappings"](#).

## Configuring an Object-Relational Data Type Descriptor

This chapter describes the various components that you must configure to be able to use an object-relational data type descriptor.

This chapter includes the following sections:

- [Introduction to Object-Relational Data Type Descriptor Configuration](#)
- [Configuring Field Ordering](#)

For information on how to configure TopLink descriptor options common to two or more descriptor types, see [Chapter 119, "Configuring a Descriptor"](#).

### 26.1 Introduction to Object-Relational Data Type Descriptor Configuration

[Table 26–1](#) lists the configurable options for an object-relational data type descriptor.

**Table 26–1** Configurable Options for Object-Relational Data Type Descriptor

Option to Configure	JDeveloper	TopLink Workbench	Java
Field ordering (see <a href="#">Section 26.2, "Configuring Field Ordering"</a> )			✓
Primary keys (see <a href="#">Section 119.2, "Configuring Primary Keys"</a> )			✓
Read-only descriptors (see <a href="#">Section 119.3, "Configuring Read-Only Descriptors"</a> )			✓
Unit of work conforming (see <a href="#">Section 119.4, "Configuring Unit of Work Conforming at the Descriptor Level"</a> )			✓
Query keys (see <a href="#">Section 119.10, "Configuring Query Keys"</a> )			✓
Cache expiration (see <a href="#">Section 119.16, "Configuring Cache Expiration at the Descriptor Level"</a> )			✓
Amendment methods (see <a href="#">Section 119.35, "Configuring Amendment Methods"</a> )			✓
Reading subclasses on queries (see <a href="#">Section 119.19, "Configuring Reading Subclasses on Queries"</a> )			✓
Inheritance for a child class descriptor (see <a href="#">Section 119.20, "Configuring Inheritance for a Child (Branch or Leaf) Class Descriptor"</a> )			✓
Inheritance for a parent class descriptor (see <a href="#">Section 119.21, "Configuring Inheritance for a Parent (Root) Descriptor"</a> )			✓
Inheritance expressions for a parent class descriptor (see <a href="#">Section 119.22, "Configuring Inheritance Expressions for a Parent (Root) Class Descriptor"</a> )			✓

**Table 26–1 (Cont.) Configurable Options for Object-Relational Data Type Descriptor**

Option to Configure	JDeveloper	TopLink Workbench	Java
Inherited attribute mapping in a subclass (see Section 119.23, "Configuring Inherited Attribute Mapping in a Subclass")			✓
Cache type and size (see Section 119.12, "Configuring Cache Type and Size at the Descriptor Level")			✓
Domain object method as an event handler (see Section 119.24, "Configuring a Domain Object Method as an Event Handler")			✓
Descriptor event listener as an event handler (see Section 119.25, "Configuring a Descriptor Event Listener as an Event Handler")			✓
Locking policy (see Section 119.26, "Configuring Locking Policy")			✓
Copy policy (see Section 119.29, "Configuring Copy Policy")			✓
Instantiation policy (see Section 119.28, "Configuring Instantiation Policy")			✓
Wrapper policy (see Section 119.32, "Configuring Wrapper Policy")			✓
History policy (see Section 119.31, "Configuring a History Policy")			✓
Returning policy (see Section 119.27, "Configuring Returning Policy")			✓

For more information, see [Chapter 21, "Introduction to Relational Descriptors"](#).

## 26.2 Configuring Field Ordering

If your object-relational data type data source driver uses JDBC indexed arrays, you can specify the order in which TopLink persists object attributes to define the field index.

### 26.2.1 How to Configure Field Ordering Using Java

Use `ObjectRelationalDescriptor` method `addFieldOrdering` to specify the field ordering. [Example 26–1](#) shows how to specify the order of the object-relational data type database fields `OBJECT_ID`, `F_NAME`, and `L_NAME` for the `Employee` descriptor.

**Example 26–1 Field Ordering**

```
descriptor.addFieldOrdering("ID");
descriptor.addFieldOrdering("F_NAME");
descriptor.addFieldOrdering("L_NAME");
```

# Part XII

---

## Relational Mappings

A relational mapping transforms any object data member type to a corresponding relational database (SQL) data source representation in any supported relational database. Relational mappings let you map an object model into a relational data model.

This part contains the following chapters:

- [Chapter 27, "Introduction to Relational Mappings"](#)  
This chapter describes each of the different TopLink relational mapping types and important relational mapping concepts.
- [Chapter 28, "Configuring a Relational Mapping"](#)  
This chapter explains how to configure TopLink relational mapping options common to two or more relational mapping types.
- [Chapter 29, "Configuring a Relational Direct-to-Field Mapping"](#)  
This chapter explains how to configure a direct to field relational database mapping.
- [Chapter 30, "Configuring a Relational Direct-to-XMLType Mapping"](#)  
This chapter explains how to configure a direct mapping to an Oracle XDB XML type field.
- [Chapter 31, "Configuring a Relational One-to-One Mapping"](#)  
This chapter explains how to configure a one-to-one relational database mapping.
- [Chapter 32, "Configuring a Relational Variable One-to-One Mapping"](#)  
This chapter explains how to configure a variable one-to-one relational database mapping.
- [Chapter 33, "Configuring a Relational One-to-Many Mapping"](#)  
This chapter explains how to configure a one-to-many relational database mapping.
- [Chapter 34, "Configuring a Relational Many-to-Many Mapping"](#)  
This chapter explains how to configure a many-to-many relational database mapping.
- [Chapter 35, "Configuring a Relational Aggregate Collection Mapping"](#)  
This chapter explains how to configure an aggregate collection relational database mapping.
- [Chapter 36, "Configuring a Relational Direct Collection Mapping"](#)

This chapter explains how to configure a direct collection relational database mapping.

- [Chapter 37, "Configuring a Relational Aggregate Object Mapping"](#)

This chapter explains how to configure an aggregate object relational database mapping.

- [Chapter 38, "Configuring a Relational Direct Map Mapping"](#)

This chapter explains how to configure a direct map relational database mapping.

- [Chapter 39, "Configuring a Relational Transformation Mapping"](#)

This chapter explains how to configure a transformation relational database mapping.

---

---

## Introduction to Relational Mappings

A relational mapping transforms any object data member type to a corresponding relational database (SQL) data source representation in any supported relational database. Relational mappings let you map an object model into a relational data model.

Relational mappings transform object data members to relational database fields. Use them to map simple data types including primitives (such as `int`), JDK classes (such as `String`), and large object (LOB) values. You can also use them to transform object data members that reference other domain objects by way of association where data source representations require object identity maintenance (such as sequencing and back references) and possess various types of multiplicity and navigability. The appropriate mapping class is chosen primarily by the cardinality of the relationship.

Do not confuse relational mappings with object-relational data type mappings (see [Chapter 40, "Introduction to Object-Relational Data Type Mappings"](#)). An object-relational data type mapping transforms certain object data member types to structured data source representations optimized for storage in specialized object-relational data type databases, such as Oracle Database. Object-relational data type mappings let you map an object model into an object-relational data type data model. In general, you can use relational mappings with any supported relational database. You can only use object-relational data type mappings with specialized object-relational data type databases optimized to support object-relational data type data source representations.

This chapter includes the following sections:

- [Relational Mapping Types](#)
- [Relational Mapping Concepts](#)
- [Direct-to-Field Mapping](#)
- [Direct-to-XMLType Mapping](#)
- [One-to-One Mapping](#)
- [Variable One-to-One Mapping](#)
- [One-to-Many Mapping](#)
- [Many-to-Many Mapping](#)
- [Aggregate Collection Mapping](#)
- [Direct Collection Mapping](#)
- [Direct Map Mapping](#)
- [Aggregate Object Mapping](#)

- [Transformation Mapping](#)

For information on mapping concepts and features common to more than one type of TopLink mappings, see [Chapter 17, "Introduction to Mappings"](#).

## 27.1 Relational Mapping Types

TopLink supports the relational mappings listed in [Table 27-1](#).

**Table 27-1 TopLink Relational Mapping Types**

Type of Mapping	Description	Oracle JDeveloper	TopLink Workbench	Java
Direct-to-field (see <a href="#">Section 27.3, "Direct-to-Field Mapping"</a> )	Map a Java attribute directly to a database field.	✓	✓	✓
Direct-to-XMLType (see <a href="#">Section 27.4, "Direct-to-XMLType Mapping"</a> )	Map Java attributes to an XMLType column in an Oracle Database (introduced in version 9.2.0.1).	✓	✓	✓
One-to-one (see <a href="#">Section 27.5, "One-to-One Mapping"</a> )	Map a reference to another persistent Java object to the database.	✓	✓	✓
Variable one-to-one (see <a href="#">Section 27.6, "Variable One-to-One Mapping"</a> )	Map a reference to an interface to the database.	✓	✓	✓
One-to-many (see <a href="#">Section 27.7, "One-to-Many Mapping"</a> )	Map Java collections of persistent objects to the database.	✓	✓	✓
Many-to-many (see <a href="#">Section 27.8, "Many-to-Many Mapping"</a> )	Use an association table to map Java collections of persistent objects to the database.	✓	✓	✓
Aggregate collection (see <a href="#">Section 27.9, "Aggregate Collection Mapping"</a> )	Map Java collections of persistent objects to the database.			✓
Direct collection (see <a href="#">Section 27.10, "Direct Collection Mapping"</a> )	Map Java collections of objects that do not have descriptors.	✓	✓	✓
Direct map (see <a href="#">Section 27.11, "Direct Map Mapping"</a> )	Direct map mappings store instances that implement <code>java.util.Map</code> .	✓	✓	✓
Aggregate object (see <a href="#">Section 27.12, "Aggregate Object Mapping"</a> )	Create strict one-to-one mappings that require both objects to exist in the same database row.	✓	✓	
Transformation (see <a href="#">Section 27.13, "Transformation Mapping"</a> )	Create custom mappings where one or more fields can be used to create the object to be stored in the attribute.	✓	✓	✓

## 27.2 Relational Mapping Concepts

This section introduces direct mapping concepts unique to TopLink, including the following:

- [Directionality](#)
- [Converters and Transformers](#)
- [Relational Mappings and EJB 2.n CMP](#)

### 27.2.1 Directionality

The direction of a relationship may be either unidirectional or bidirectional. In a unidirectional relationship, only one entity bean has a relationship field that refers to the other. All TopLink relational mappings are unidirectional, from the class being described (the *source* class) to the class with which it is associated (the *target* class). The



target class does not have a reference to the source class in a unidirectional relationship.

In a bidirectional relationship, each entity bean has a relationship field that refers to the other bean. Through the relationship field, an entity bean's code can access its related object. To implement a bidirectional relationship (classes that reference each other), use two unidirectional mappings with the sources and targets reversed.

---



---

**Note:** Maintenance of bidirectional relationships presents a number of technical challenges. For more information, see the following:

- [Section 2.14.3.4, "Maintaining Bidirectional Relationships"](#)
  - [Section 121.18, "Configuring Bidirectional Relationship"](#)
  - [Section 17.2.4, "Indirection \(Lazy Loading\)"](#)
- 
- 

## 27.2.2 Converters and Transformers

You can store object attributes directly in a database table as follows:

- Direct mapping (see [Section 27.2.2.1, "Using a Direct Mapping"](#))
- Converter Mapping (see [Section 27.2.2.2, "Using a Converter Mapping"](#))
- Transformation mapping (see [Section 27.2.2.3, "Using a Transformation Mapping"](#))

### 27.2.2.1 Using a Direct Mapping

If the attribute type is comparable to a database type, the information can be stored directly simply by using a direct-to-field mapping (see [Section 27.3, "Direct-to-Field Mapping"](#)).

### 27.2.2.2 Using a Converter Mapping

If the attribute type is comparable to a database type but requires conversion, the information can be stored directly by using a direct-to-field mapping (see [Section 27.3, "Direct-to-Field Mapping"](#)) and an appropriate Converter instance.

In the previous release, TopLink provided subclasses of `DirectToFieldMapping` for object type direct mappings, serialized object direct mappings, and type conversion direct mappings. In this release, these subclasses are deprecated. In their place, Oracle recommends that you use the `DirectToFieldMapping` method `setConverter` and the corresponding Converter instance. [Table 27–2](#) summarizes these changes.

**Table 27–2 Using a Converter for Direct-to-Field Mappings**

Deprecated DirectToFieldMapping subclass...	Replaced by Converter instance...
<code>ObjectTypeMapping</code>	<code>ObjectTypeConverter</code> (see <a href="#">Section 17.2.6.3, "Object Type Converter"</a> )
<code>SerializedObjectMapping</code>	<code>SerializedObjectConverter</code> (see <a href="#">Section 17.2.6.1, "Serialized Object Converter"</a> )
<code>TypeConversionMapping</code>	<code>TypeConversionConverter</code> (see <a href="#">Section 17.2.6.2, "Type Conversion Converter"</a> )

If the application's objects contain attributes that cannot be represented as direct-to-field with an existing converter, use a direct-to-field mapping with a custom converter.

### 27.2.2.3 Using a Transformation Mapping

If there is no database primitive type that is logically comparable to the attribute's type, or, if an attribute requires data from multiple fields, it must be transformed on its way to and from the database.

In this case, use a transformation mapping (see [Section 27.13, "Transformation Mapping"](#)).

## 27.2.3 Relational Mappings and EJB 2.n CMP

Use direct mappings to map the (non-CMR) CMF attributes of a bean.

In EJB CMP projects, the bean class does not define real variables in the class – only abstract getter and setter methods. To map the bean's attributes, you must import `ejb-jar.xml` file into TopLink Workbench (see [Section 117.5, "Configuring Persistence Type"](#)).

You can map entity bean attributes using direct mappings without any special considerations.

---



---

**Note:** When you work with EJB, do not map the entity context attribute (type `javax.ejb.EntityContext`).

---



---

There are some special considerations when using one-to-one mappings (see [Section 27.5.1, "One-to-One Mappings and EJB 2.n CMP"](#)), one-to-many mappings (see [Section 27.7.1, "One-to-Many Mappings and EJB 2.n CMP"](#)), and many-to-many mappings (see [Section 27.8.1, "Many-to-Many Mappings and EJB 2.n CMP"](#)).

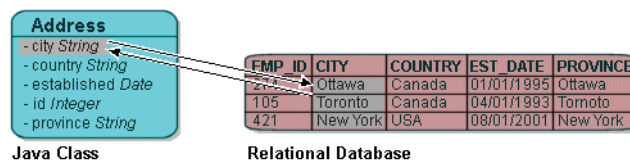
## 27.3 Direct-to-Field Mapping

Use direct-to-field mappings to map primitive object attributes, or non persistent regular objects, such as the JDK classes. For example, use a direct-to-field mapping to store a `String` attribute in a `VARCHAR` field.

### Example 27–1 Direct-to-Field Mapping Example

[Figure 27–1](#) illustrates a direct-to-field mapping between the Java attribute `city` and the relational database column `CITY`. Similarly, direct-to-field mappings could be defined from `country` to `COUNTRY`, `id` to `ADDRESS_ID`, `established` to `EST_DATE`, and `province` to `PROVINCE`.

**Figure 27–1** Direct-to-Field Mapping



You can use a direct-to-field mapping with any of the following `Converter` instances:

- Object type converter (see [Section 17.2.6.3, "Object Type Converter"](#))
- Serialized object converter (see [Section 17.2.6.1, "Serialized Object Converter"](#))
- Type conversion converter (see [Section 17.2.6.2, "Type Conversion Converter"](#))

You can use a direct-to-field mapping with a change policy (see [Section 119.30](#), "Configuring Change Policy").

See [Chapter 29](#), "Configuring a Relational Direct-to-Field Mapping" for more information.

## 27.4 Direct-to-XMLType Mapping

Using a direct-to-XMLType mapping, you can map XML data in the form of a `String` or an `org.w3c.dom.Document` object to an `XMLType` column in an Oracle Database (introduced in version 9.2.0.1).

If you plan to use direct-to-XMLType mappings in TopLink Workbench and the TopLink runtime, you must include the Oracle Database `xdb.jar` file in the TopLink Workbench classpath (see [Section 5.2](#), "Configuring the TopLink Workbench Environment").

The TopLink query framework provides a number of expression operators you can use to create queries based on the content of that XML data (see [Section 110.2.4](#), "XMLType Functions").

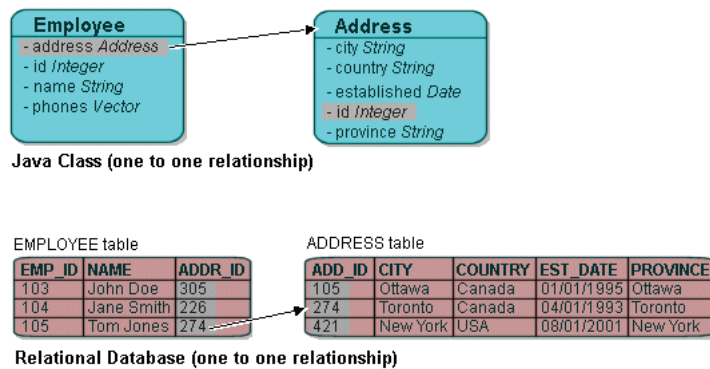
See [Chapter 30](#), "Configuring a Relational Direct-to-XMLType Mapping" for more information.

## 27.5 One-to-One Mapping

One-to-one mappings represent simple pointer references between two Java objects. In Java, a single pointer stored in an attribute represents the mapping between the source and target objects. Relational database tables implement these mappings using foreign keys.

[Figure 27–2](#) illustrates a one-to-one relationship from the `address` attribute of an `Employee` object to an `Address` object. To store this relationship in the database, create a one-to-one mapping between the `address` attribute and the `Address` class. This mapping stores the `id` of the `Address` instance in the `EMPLOYEE` table when the `Employee` instance is written. It also links the `Employee` instance to the `Address` instance when the `Employee` is read from the database. Because an `Address` does not have any references to the `Employee`, it does not have to provide a mapping to `Employee`.

For one-to-one mappings, the source table normally contains a foreign key reference to a record in the target table. In [Figure 27–2](#), the `ADDR_ID` field of the `EMPLOYEE` table is a foreign key.

**Figure 27–2 One-to-One Mappings**

You can also implement a one-to-one mapping where the target table contains a foreign key reference to the source table. In Figure 27–2, the database design would change such that the ADDRESS row would contain the EMP\_ID to identify the Employee to which it belonged. In this case, the target must also have a relationship mapping to the source.

The update, insert and delete operations, which are normally done for the target before the source for privately owned one-to-one relationships, are performed in the opposite order when the target owns the foreign key. Target foreign keys normally occur in bidirectional one-to-one mappings (see Section 27.2.1, "Directionality"), because one side has a foreign key and the other shares the same foreign key in the other's table.

Target foreign keys can also occur when large cascaded composite primary keys exist (that is, one object's primary key is composed of the primary key of many other objects). In this case it is possible to have a one-to-one mapping that contains both foreign keys and target foreign keys.

In a foreign key, TopLink automatically updates the foreign key value in the object's row. In a target foreign key, it does not. In TopLink, use the **Target Foreign Key** option when a target foreign key relationship is defined.

When mapping a relationship, you must understand these differences between a foreign key and a target foreign key, to ensure that the relationship is defined correctly.

In a bidirectional relationship where the two classes in the relationship reference each other, only one of the mappings should have a foreign key. The other mapping should have a target foreign key. If one of the mappings in a bidirectional relationship is a one-to-many mapping, see Chapter 32, "Configuring a Relational Variable One-to-One Mapping" for details.

You can use a one-to-one mapping with a change policy (see Section 119.30, "Configuring Change Policy").

See Section 31, "Configuring a Relational One-to-One Mapping" for more information.

### 27.5.1 One-to-One Mappings and EJB 2.n CMP

To maintain EJB compliance, the object attribute that points to the target of the relationship must be the local interface type—not the bean class.

TopLink provides variations on one-to-one mappings that lets you define complex relationships when the target of the relationship is a dependent Java object. For example, *variable one-to-one mappings* enable you to specify variable target objects in the

relationship. These variations are not available for entity beans, but are valid for dependent Java objects.

For more information, see the [Chapter 32, "Configuring a Relational Variable One-to-One Mapping"](#).

## 27.6 Variable One-to-One Mapping

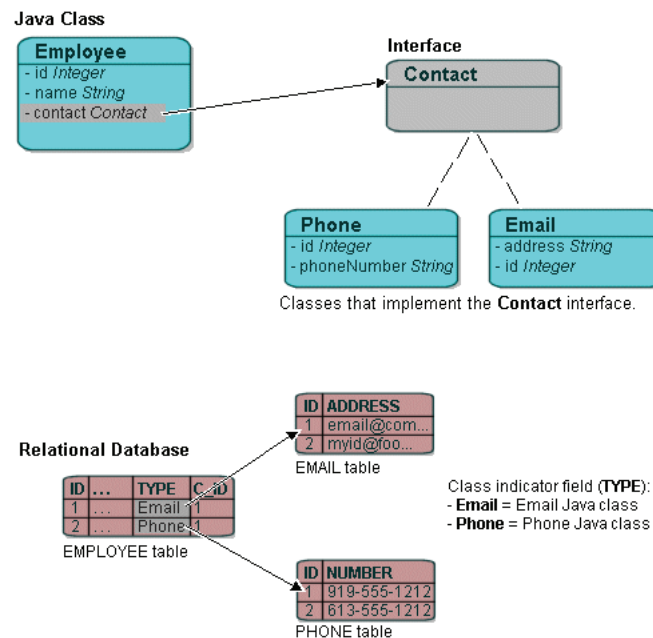
Variable class relationships are similar to polymorphic relationships, except that in this case the target classes are not related through inheritance (and thus not good candidates for an abstract table), but through an interface.

To define variable class relationships in TopLink Workbench, use the variable one-to-one mapping selection, but choose the interface as the reference class. This makes the mapping a variable one-to-one. When defining mappings in Java code, use the `VariableOneToOneMapping` class.

TopLink supports variable relationships only in one-to-one mappings. It handles this relationship in two ways:

- Through the class indicator field (see [Section 32.2, "Configuring Class Indicator"](#)).
- Through unique primary key values among target classes implementing the interface (see [Section 32.3, "Configuring Unique Primary Key"](#)).

**Figure 27–3 Variable One-to-One Mappings with Class Indicator**



See [Chapter 32, "Configuring a Relational Variable One-to-One Mapping"](#) for more information.

## 27.7 One-to-Many Mapping

One-to-many mappings are used to represent the relationship between a single source object and a collection of target objects. They are a good example of something that is

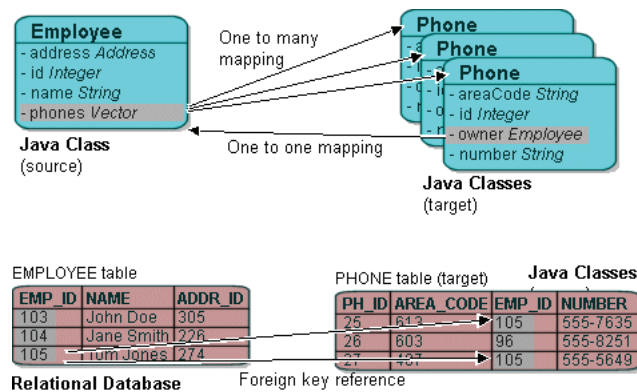
simple to implement in Java using a `Collection` (or other collection types) of target objects, but difficult to implement using relational databases.

In a Java `Collection`, the owner references its parts. In a relational database, the parts reference their owner. Relational databases use this implementation to make querying more efficient.

The purpose of creating this one-to-one mapping in the target is so that the foreign key information can be written when the target object is saved. Alternatives to the one-to-one mapping back reference include the following:

- Use a direct-to-field mapping to map the foreign key and maintain its value in the application. Here the object model does not require a back reference, but the data model still requires a foreign key in the target table.
- Use a many-to-many mapping to implement a logical one-to-many. This has the advantage of not requiring a back reference in the object model and not requiring a foreign key in the data model. In this model the many-to-many relation table stores the collection. It is possible to put a constraint on the join table to enforce that the relation is a logical one-to-many relationship.

**Figure 27-4 One-to-Many Relationships**



**Note:** The phone attribute shown in Figure 27-4 is of type `Vector`. You can use a `Collection` interface (or any class that implements the `Collection` interface) for declaring the collection attribute. See Section 121.14, "Configuring Container Policy" for details.

You can use a many-to-many mapping with a change policy (see Section 119.30, "Configuring Change Policy").

See Chapter 33, "Configuring a Relational One-to-Many Mapping" for more information.

### 27.7.1 One-to-Many Mappings and EJB 2.n CMP

Use one-to-many mappings for relationships between entity beans or between an entity bean and a collection of privately owned regular Java objects. When you create one-to-many mappings, also create a one-to-one mapping from the target objects back to the source. The object attribute that contains a pointer to the bean must be the local interface type—not the bean class.

TopLink automatically maintains back-pointers when you create or update bidirectional relationships between beans.

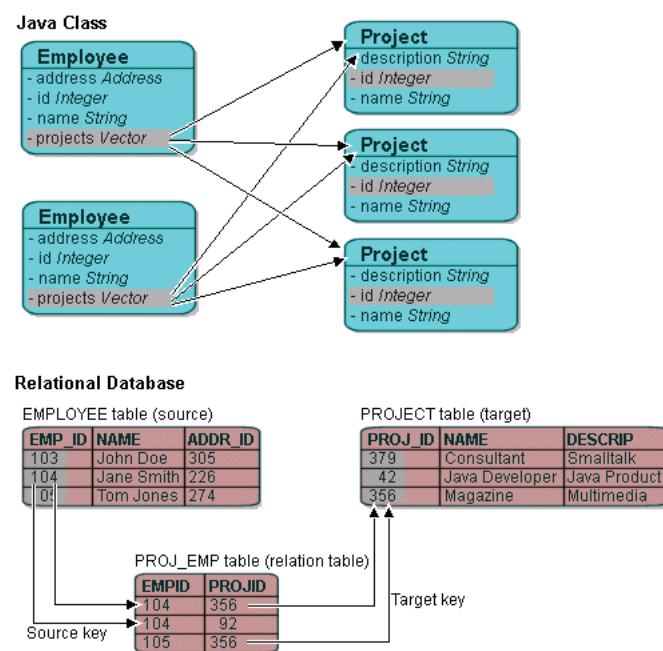
For more information, see [Section 121.18, "Configuring Bidirectional Relationship"](#).

## 27.8 Many-to-Many Mapping

Many-to-many mappings represent the relationships between a collection of source objects and a collection of target objects. They require the creation of an intermediate table for managing the associations between the source and target records.

[Figure 27-5](#) illustrates a many-to-many mapping in Java and in relational database tables.

**Figure 27-5 Many-to-many Relationships**



**Note:** The projects attribute shown in [Figure 27-5](#) is of type Vector. You can use a Collection interface (or any class that implements the Collection interface) for declaring the collection attribute. See [Section 121.14, "Configuring Container Policy"](#) for details.

Many-to-many mappings are implemented using a relation table. This table contains columns for the primary keys of the source and target tables. Composite primary keys require a column for each field of the composite key. The intermediate table must be created in the database before using the many-to-many mapping.

The target class does not have to implement any behavior for the many-to-many mappings. If the target class also creates a many-to-many mapping back to its source, then it can use the same relation table, but one of the mappings must be set to read-only. If both mappings write to the table, they can cause collisions.

Indirection (lazy loading) is enabled by default in a many-to-many mapping, which requires that the attribute have the `ValueHolderInterface` type or transparent collections. For more information on indirection, see [Section 17.2.4, "Indirection \(Lazy Loading\)"](#).

You can use a many-to-many mapping with a change policy (see [Section 119.30, "Configuring Change Policy"](#)).

See [Chapter 34, "Configuring a Relational Many-to-Many Mapping"](#) for more information.

### 27.8.1 Many-to-Many Mappings and EJB 2.n CMP

When you use CMP, many-to-many mappings are valid only between entity beans, and cannot be privately owned. The only exception is when a many-to-many mapping is used to implement a logical one-to-many mapping with a relation table.

`TopLink` automatically maintains back-pointers when you create or update bidirectional relationships.

For more information, see [Section 121.18, "Configuring Bidirectional Relationship"](#).

## 27.9 Aggregate Collection Mapping

Aggregate collection mappings are used to represent the aggregate relationship between a single-source object and a collection of target objects. Unlike the `TopLink` one-to-many mappings, in which there should be a one-to-one back reference mapping from the target objects to the source object, there is no back reference required for the aggregate collection mappings, because the foreign key relationship is resolved by the aggregation.

---

---

**Note:** To use aggregate collections with `TopLink Workbench`, you must use an amendment method (see [Section 119.35, "Configuring Amendment Methods"](#)), or manually edit the project source to add the mapping.

---

---

Although aggregate collection mappings are similar to one-to-many mappings, they are not replacements for one-to-many mappings. Use aggregate collections only in situations where the target collections are of a reasonable size and if having a one-to-one back mapping is difficult.

Because one-to-many relationships offer better performance and are more robust and scalable, consider using a one-to-many relationship rather than an aggregate collection. In addition, aggregate collections are privately owned by the source of the relationship and must not be shared or referenced by other objects.

This section describes the following:

- [Aggregate Collection Mappings and Inheritance](#)
- [Aggregate Collection Mappings and EJB](#)
- [How to Implement Aggregate Collection Mappings](#)

See [Chapter 35, "Configuring a Relational Aggregate Collection Mapping"](#) for more information.



### 27.9.1 Aggregate Collection Mappings and Inheritance

Aggregate collection descriptors can use inheritance. You must also declare subclasses as aggregate collection. The subclasses can have their own mapped tables, or share the table with their parent class. See [Section 16.2.2, "Descriptors and Inheritance"](#) for more information on inheritance.

In a Java `Collection`, the owner references its parts. In a relational database, the parts reference their owners. Relational databases use this implementation to make querying more efficient.

Aggregate collection mappings require a target table for the target objects.

To implement an aggregate collection mapping, the following must take place:

- The descriptor of the target class must declare itself as an aggregate collection object. Unlike the aggregate object mapping, in which the target descriptor does not have a specific table to associate with, there must be a target table for the target object.
- The descriptor of the source class must add an aggregate collection mapping that specifies the target class.

### 27.9.2 Aggregate Collection Mappings and EJB

You can use aggregate collection mappings with entity beans if the source of the relationship is an entity bean or Java object, and the mapping targets are regular Java objects. Entity beans cannot be the target of an aggregate object mapping.

### 27.9.3 How to Implement Aggregate Collection Mappings

To implement an aggregate collection mapping, the following must take place:

- The descriptor of the target class must declare itself to be an aggregate collection object. Unlike the aggregate object mapping, in which the target descriptor does not have a specific table to associate with, there must be a target table for the target object.
- The descriptor of the source class must add an aggregate collection mapping that specifies the target class.

## 27.10 Direct Collection Mapping

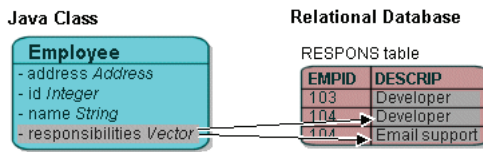
Direct collection mappings store collections of Java objects that are not TopLink-enabled. The object type stored in the direct collection is typically a Java type, such as `String`.

It is also possible to use direct collection mappings to map a collection of non-`String` objects. For example, it is possible to have an attribute that contains a collection of `Integer` or `Date` instances. The instances stored in the collection can be any type supported by the database and has a corresponding wrapper class in Java.

Support for primitive data types such as `int` is not provided, because Java `Collection` holds only objects.

[Figure 27–6](#) illustrates how a direct collection is stored in a separate table with two fields. The first field is the reference key field, which contains a reference to the primary key of the instance owning the collection. The second field contains an object in the collection and is called the direct field. There is one record in the table for each object in the collection.

**Figure 27–6 Direct Collection Mappings**



**Note:** The `responsibilities` attribute shown in Figure 27–6 is of type `Vector`. You can use a `Collection` interface (or any class that implements the `Collection` interface) for declaring the collection attribute. See Section 121.14, "Configuring Container Policy" for details.

Maps are not supported for direct collection because there is no key value.

You can use a direct collection mapping with any of the following `Converter` instances:

- Section 17.2.6.3, "Object Type Converter"
- Section 17.2.6.1, "Serialized Object Converter"
- Section 17.2.6.2, "Type Conversion Converter"

You can use a direct collection mapping with a change policy (see Section 119.30, "Configuring Change Policy").

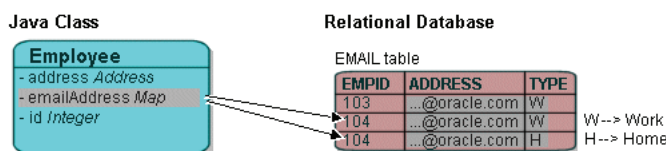
See Chapter 36, "Configuring a Relational Direct Collection Mapping" for more information.

## 27.11 Direct Map Mapping

Direct map mappings store instances that implement `java.util.Map`. Unlike one-to-many or many-to-many mappings, the keys and values of the map in this type of mapping are Java objects that do not have descriptors. The object type stored in the key and the value of direct map are Java primitive wrapper types such as `String` objects.

Figure 27–7 illustrates how a direct map is stored in a separate table with three fields. The first field (`EMPID`) is the reference key field, which contains a reference to the primary key of the instance owning the collection. The second field (`ADDRESS`) contains an object in the collection and is called the direct value field. The third field (`TYPE`) contains the direct key field. In this example, the direct map uses a object type converter for the direct key field, converting the single character `W` in the database to the full string `Work` in the object (and `H` to `Home`).

**Figure 27–7 Direct Map Mappings**



You can use a direct collection mapping with any of the following `Converter` instances:

- [Section 17.2.6.3, "Object Type Converter"](#)
- [Section 17.2.6.1, "Serialized Object Converter"](#)
- [Section 17.2.6.2, "Type Conversion Converter"](#)

You can use a direct map mapping with a change policy (see [Section 119.30, "Configuring Change Policy"](#)).

See [Chapter 38, "Configuring a Relational Direct Map Mapping"](#) for more information.

## 27.12 Aggregate Object Mapping

Two objects—a source (parent or owning) object and a target (child or owned) object—are related by aggregation if there is a strict one-to-one relationship between them and all the attributes of the target object can be retrieved from the same table(s) as the source object. This means that if the source object exists, then the target object must also exist and if the source object is destroyed, then the target object is also destroyed.

An aggregate mapping allows you to associate data members in the target object with fields in the source object's underlying database tables.

You configure the aggregate mapping in the source object's descriptor. However, before doing so, you must designate the target object's descriptor as an aggregate (see [Section 23.6, "Configuring a Relational Descriptor as a Class or Aggregate Type"](#)).

Aggregate objects are privately owned and should not be shared or referenced by other objects.

You cannot configure one-to-one, one-to-many, or many-to-many mappings from a nonaggregate object to an aggregate target object.

You can configure such mappings from an aggregate target object to another nonaggregate object. If you configure a one-to-many mapping from an aggregate target object to another nonaggregate object, you must configure a one-to-one mapping from the other object back to the source object that owns the aggregate (instead of to the aggregate target object itself). This is because the source object contains the table and primary key information of the aggregate target.

You can configure inheritance for a descriptor designated as an aggregate (see [Section 16.2.2, "Descriptors and Inheritance"](#)), however, in this case, *all* the descriptors in the inheritance tree must be aggregates. Aggregate and class descriptors cannot exist in the same inheritance tree.

This section describes the following:

- [Aggregate Object Mappings with a Single Source Object](#)
- [Aggregate Object Mappings with Multiple Source Objects](#)
- [How to Implement an Aggregate Object Relationship Mapping](#)

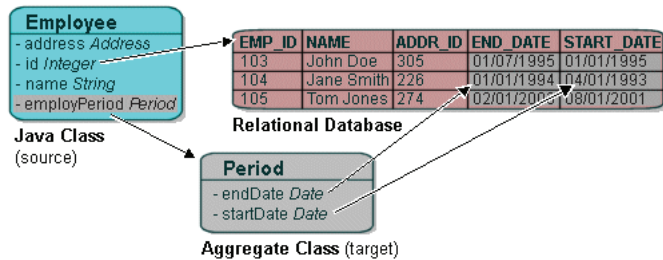
You can use an aggregate object mapping with a change policy (see [Section 119.30, "Configuring Change Policy"](#)).

For more information on configuring an aggregate object relationship mapping, see [Chapter 37, "Configuring a Relational Aggregate Object Mapping"](#).

### 27.12.1 Aggregate Object Mappings with a Single Source Object

Figure 27–8 shows an example aggregate object mapping between source object Employee and target object Period. In this example, the target object is not shared by other types of source object.

Figure 27–8 Aggregate Object Mapping with a Single Source Object

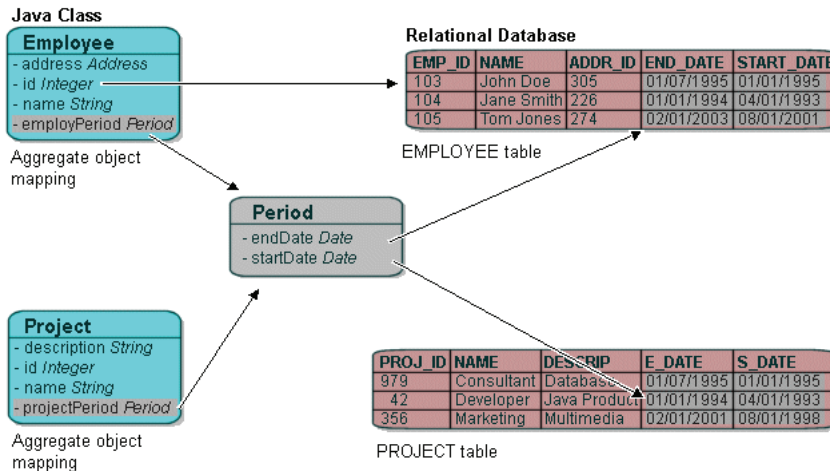


Aggregate target classes not shared among multiple source classes can have any type of mapping, including other aggregate object mappings.

### 27.12.2 Aggregate Object Mappings with Multiple Source Objects

Figure 27–9 shows an example aggregate object mapping in which different source objects—Employee and Project—map instances of the same type of target object, Period.

Figure 27–9 Aggregate Object Mapping with Multiple Source Objects



When you configure the aggregate object mapping in the source object, you choose the source object table for that particular mapping. This allows different source types to store the same target information within their tables. Each source object’s table may use different field names. TopLink automatically manages the case where multiple source object tables use different field names.

For example, in Figure 27–9, The `Employee` attribute `employPeriod` is mapped by an aggregate object mapping to target object `Period`. This mapping associates `Period` attribute `startDate` with `EMPLOYEE` table field `START_DATE`. The `Project` attribute `projectPeriod` is also mapped by an aggregate object mapping to target

object `Period`. This mapping associates `Period` attribute `startDate` with `PROJECT` table field `S_DATE`.

Aggregate target classes shared with multiple source classes cannot have one-to-many or many-to-many mappings.

### 27.12.3 How to Implement an Aggregate Object Relationship Mapping

You must ensure that the following takes place:

- The descriptor of the target class declares itself to be an aggregate object. Because all its information comes from its parent's table(s), the target descriptor does not have a specific table associated with it. You must, however, choose one or more candidate table(s) from which you can use fields in mapping the target.

In the example above, you could choose the `EMPLOYEE` table so that the `START_DATE` and `END_DATE` fields are available during mapping.

- The descriptor of the source class adds an aggregate object mapping that specifies the target class.

In the example above, the `Employee` class has an attribute called `employPeriod` that would be mapped as an aggregate object mapping with `Period` as the reference class.

The source class must ensure that its table has fields that correspond to the field names registered with the target class.

- If a source object has a `null` target reference, `TopLink` writes `NULLs` to the aggregate database fields (see [Section 37.3, "Configuring Allowing Null Values"](#)). When the source is read from the database, it can handle this `null` target in one of two ways:
  - Create an instance of the object with all its attributes equal to `null`.
  - Put a `null` reference in the source object without instantiating a target. (This is the default method of handling `null` targets.)

## 27.13 Transformation Mapping

Use transformation mappings for specialized translations for how a value is represented in Java and how it is represented in the database.

---



---

**Tip:** Use transformation mappings only when mapping multiple fields into a single attribute. Because of the complexity of transformation mappings, it is often easier to perform the transformation with a converter or getter and setter methods of a direct-to-field mapping. See [Chapter 29, "Configuring a Relational Direct-to-Field Mapping"](#) for more information.

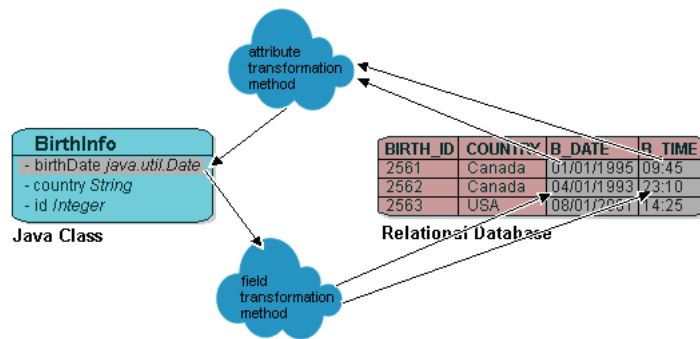
---



---

[Figure 27-10](#) illustrates a transformation mapping. The values from the `B_DATE` and `B_TIME` fields are used to create a `java.util.Date` to be stored in the `birthdate` attribute.

Figure 27-10 Transformation Mappings



Often, a transformation mapping is appropriate when values from multiple fields are used to create an object. This type of mapping requires that you provide an *attribute transformation* that is invoked when reading the object from the database. This must have at least one parameter that is an instance of `Record`. In your attribute transformation, you can use `Record` method `get` to retrieve the value in a specific column. Your attribute transformation can specify a second parameter, when it is an instance of `Session`. The `Session` performs queries on the database to get additional values needed in the transformation. The transformation should *return* the value to be stored in the attribute.

Transformation mappings also require a *field transformation* for each field, to be written to the database when the object is saved. The transformation returns the value to be stored in that field.

See [Chapter 39, "Configuring a Relational Transformation Mapping"](#) for more information.

---



---

## Configuring a Relational Mapping

This chapter describes how to configure a relational mapping.

This chapter includes the following sections:

- [Introduction to Relational Mapping Configuration](#)
- [Configuring Common Relational Mapping Options](#)
- [Configuring a Database Field](#)
- [Configuring Reference Descriptor](#)
- [Configuring Batch Reading](#)
- [Configuring Query Key Order](#)
- [Configuring Table and Field References \(Foreign and Target Foreign Keys\)](#)
- [Configuring Joining at the Mapping Level](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 28.1 Introduction to Relational Mapping Configuration

[Table 28–1](#) lists the types of relational mappings that you can configure and provides a cross-reference to the type-specific chapter that lists the configurable options supported by that type.

**Table 28–1** *Configuring Relational Mappings*

<b>If you are creating...</b>	<b>See...</b>
Direct-to-Field Mapping	<a href="#">Chapter 29, "Configuring a Relational Direct-to-Field Mapping"</a>
Transformation Mapping	<a href="#">Chapter 39, "Configuring a Relational Transformation Mapping"</a>
Direct-to-XMLType Mapping	<a href="#">Chapter 30, "Configuring a Relational Direct-to-XMLType Mapping"</a>
One-to-One Mapping	<a href="#">Chapter 31, "Configuring a Relational One-to-One Mapping"</a>
Variable One-to-One Mapping	<a href="#">Chapter 32, "Configuring a Relational Variable One-to-One Mapping"</a>
One-to-Many Mapping	<a href="#">Chapter 33, "Configuring a Relational One-to-Many Mapping"</a>
Many-to-Many Mapping	<a href="#">Chapter 34, "Configuring a Relational Many-to-Many Mapping"</a>
Aggregate Collection Mapping	<a href="#">Chapter 35, "Configuring a Relational Aggregate Collection Mapping"</a>
Direct Collection Mapping	<a href="#">Chapter 36, "Configuring a Relational Direct Collection Mapping"</a>
Direct Map Mapping	<a href="#">Chapter 38, "Configuring a Relational Direct Map Mapping"</a>

**Table 28–1 (Cont.) Configuring Relational Mappings**

If you are creating...	See...
Aggregate Object Mapping	Chapter 37, "Configuring a Relational Aggregate Object Mapping"

For more information, see the following:

- [Chapter 17, "Introduction to Mappings"](#)
- [Chapter 27, "Introduction to Relational Mappings"](#)

## 28.2 Configuring Common Relational Mapping Options

Table 28–2 lists the configurable options shared by two or more relational mapping types.

**Table 28–2 Common Relational Mapping Options**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Database field (see Section 28.3, "Configuring a Database Field")	✓	✓	✓
Reference descriptor (see Section 28.4, "Configuring Reference Descriptor")	✓	✓	✓
Container policy (see Section 121.14, "Configuring Container Policy")	✓	✓	✓
Method or direct field access (see Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level")	✓	✓	✓
Default null value (see Section 121.5, "Configuring a Default Null Value at the Mapping Level")	✓	✓	✓
Read-only mapping (see Section 121.2, "Configuring Read-Only Mappings")	✓	✓	✓
Indirection (lazy loading) (see Section 121.3, "Configuring Indirection (Lazy Loading)")	✓	✓	✓
Private or Independent relationships (see Section 121.7, "Configuring Private or Independent Relationships")	✓	✓	✓
Mapping comments (see Section 121.8, "Configuring Mapping Comments")	✓	✓	
Serialized object converter (see Section 121.9, "Configuring a Serialized Object Converter")	✓	✓	✓
Type conversion converter (see Section 121.10, "Configuring a Type Conversion Converter")	✓	✓	✓
Object type converter (see Section 121.11, "Configuring an Object Type Converter")	✓	✓	✓
Bidirectional relationship (see Section 121.18, "Configuring Bidirectional Relationship")	✓	✓	✓
Batch reading (see Section 28.5, "Configuring Batch Reading")	✓	✓	✓
Query key order (see Section 28.6, "Configuring Query Key Order")	✓	✓	✓
Table and field references (see Section 28.7, "Configuring Table and Field References (Foreign and Target Foreign Keys)")	✓	✓	✓
Joining (see Section 28.8, "Configuring Joining at the Mapping Level")	✓	✓	✓



## 28.3 Configuring a Database Field

You can associate an object attribute with a database field.

Table 28–3 summarizes which relational mappings support this option.

**Table 28–3 Relational Mapping Support for Database Field**

Mapping	How to Use Oracle JDeveloper	How to Configure a Database Field Using TopLink Workbench	How to Use Java
Direct-to-Field Mapping	✓	✓	✓
Direct-to-XMLType Mapping	✓	✓	✓

When choosing the database field, you must consider Java and database field type compatibility.

TopLink supports the following Java types:

- `java.lang: Boolean, Float, Integer, String, Double, Long, Short, Byte, Byte[ ], Character, Character[ ]`; all the primitives associated with these classes
- `java.math: BigInteger, BigDecimal`
- `java.sql: Date, Time, Timestamp`
- `java.util: Date, Calendar`

While executing reads, the mappings in Table 28–6 perform the simple one-way data conversions that Table 28–4 describes. For two-way or more complex conversions, you must use converters (see Section 27.2.2, "Converters and Transformers").

The mappings in Table 28–3 also allow you to specify a null value. This may be required if primitive types are used in the object, and the database field allows null values. For more information, see Section 121.5, "Configuring a Default Null Value at the Mapping Level".

**Table 28–4 Type Conversions Provided by Direct-to-Field Mappings**

Java type	Database type
Integer, Float, Double, Byte, Short, BigDecimal, BigInteger, int, float, double, byte, short	NUMBER, NUMERIC, DECIMAL, FLOAT, DOUBLE, INT, SMALLINT, BIT, BOOLEAN
Boolean, boolean	BOOLEAN, BIT, SMALLINT, NUMBER, NUMERIC, DECIMAL, FLOAT, DOUBLE, INT
String	VARCHAR, CHAR, VARCHAR2, CLOB, TEXT, LONG, LONG VARCHAR, MEMO  The following types apply only to Oracle9: NVARCHAR2, NCLOB, NCHAR
byte[ ]	BLOB, LONG RAW, IMAGE, RAW, VARBINARY, BINARY, LONG VARBINARY
Time	TIME
<code>java.sql.Date</code>	DATE
Timestamp, <code>java.util.Date</code> , Calendar	TIMESTAMP

### Support for oracle.sql.TimeStamp

TopLink provides additional support for mapping Java date and time data types to Oracle Database DATE, TIMESTAMP, and TIMESTAMPTZ data types when you use the Oracle JDBC driver with Oracle9i Database or later and the Oracle9Platform in TopLink.

In a direct-to-field mapping, you are not required to specify the database type of the field value; TopLink determines the appropriate data type conversion.

Table 28–5 lists the supported direct-to-field mapping combinations.

**Table 28–5 Supported Oracle Database Date and Time Direct-to-Field Mappings**

Java Type	Database Type	Description
java.sql.Time	TIMESTAMP	Full bidirectional support.
	TIMESTAMPTZ	Full bidirectional support.
	DATE	Full bidirectional support.
java.sql.Date	TIMESTAMP	Full bidirectional support.
	TIMESTAMPTZ	Full bidirectional support.
	DATE	Full bidirectional support.
java.sql.Timestamp	TIMESTAMP	Full bidirectional support.
	TIMESTAMPTZ	Full bidirectional support.
	DATE	Nanoseconds are not stored in the database.
java.util.Date	TIMESTAMP	Full bidirectional support.
	TIMESTAMPTZ	Full bidirectional support.
	DATE	Milliseconds are not stored in the database.
java.util.Calendar	TIMESTAMP	Native SQL or binding gives Calendar timezone. Note: The TIMESTAMP database value has no timezone – the Calendar object provides the local timezone by default. If the database is not in this timezone, you must obtain the database timezone by some other means and update the Calendar object accordingly. For this reason, TIMESTAMPTZ may be a better choice.
	TIMESTAMPTZ	Native SQL or binding stores timezone; standard SQL is based on the local timezone.
	DATE	Neither timezone nor milliseconds are stored in the database.

Note that some of these mappings result in a loss of precision: avoid these combinations if you require this level of precision. For example, if you create a direct-to-field mapping between a java.sql.Date attribute and a TIMESTAMPTZ database field, there is no loss of precision. However, if you create a direct-to-field mapping between a java.sql.Timestamp attribute and a DATE database field, the nanoseconds or milliseconds of the attribute are not stored in the database.

### 28.3.1 How to Configure a Database Field Using TopLink Workbench

Use this procedure to select a specific database field for a direct mapping.

1. Select the direct mapping attribute in the **Navigator**. Its properties appear in the Editor.

2. Click the **General** tab. The General tab appears.

**Figure 28–1** Direct Mapping General Tab, Database Field Option

The screenshot shows a configuration window with two tabs: 'General' and 'Converter'. The 'General' tab is active. At the top, there is a 'Database Field' dropdown menu with a red circle around it, showing 'ADDRESS.CITY (VARCHAR2)'. Below this are three sections, each with a checkbox and a dropdown menu:

- Method Accessing:**  Method Accessing. Get Method: <none selected>. Set Method: <none selected>.
- Default Null Value:**  Default Null Value. Type: <none selected>. Value: (empty text box).
- Read-Only:**  Read-Only.

At the bottom, there is a 'Comment' text area.

Use the **Database Field** field to select a field for this direct mapping. You must have previously associated the descriptor with a database table as described in [Section 23.2, "Configuring Associated Tables"](#).

---

**Note:** For direct-to-field mappings of an aggregate descriptor (see [Section 23.6, "Configuring a Relational Descriptor as a Class or Aggregate Type"](#)), this field is for display only and cannot be changed.

---

## 28.4 Configuring Reference Descriptor

In relational mappings that extend `oracle.toplink.mappings.ForeignReferenceMapping`, attributes reference other TopLink descriptors—not the data source. You can select any descriptor in the project.

[Table 28–6](#) summarizes which relational mappings support this option.

**Table 28–6** Relational Mapping Support for Reference Descriptor

Mapping	How to Use Oracle JDeveloper	How to Configure a Reference Descriptor Using TopLink Workbench	How to Use Java
One-to-one (see <a href="#">Section 27.5, "One-to-One Mapping"</a> )	✓	✓	✓
Variable one-to-one (see <a href="#">Section 27.6, "Variable One-to-One Mapping"</a> )	✓	✓	✓
One-to-many (see <a href="#">Section 27.7, "One-to-Many Mapping"</a> )	✓	✓	✓
Many-to-many (see <a href="#">Section 27.8, "Many-to-Many Mapping"</a> )	✓	✓	✓

**Table 28–6 (Cont.) Relational Mapping Support for Reference Descriptor**

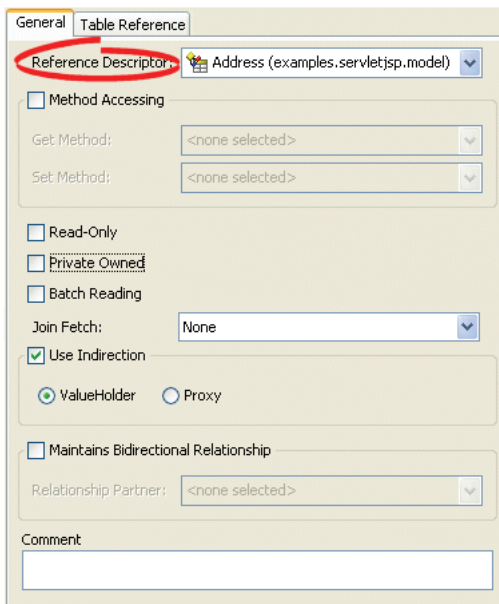
Mapping	How to Use Oracle JDeveloper	How to Configure a Reference Descriptor Using TopLink Workbench	How to Use Java
Aggregate collection (see Section 27.9, "Aggregate Collection Mapping")			✓
Aggregate object (see Section 27.12, "Aggregate Object Mapping")	✓	✓	✓

### 28.4.1 How to Configure a Reference Descriptor Using TopLink Workbench

To specify a reference descriptor for a relational mapping, use this procedure.

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

**Figure 28–2 General Tab, Reference Descriptor Field**



Use the **Reference Descriptor** field to select the descriptor referenced by this relationship mapping.

---

**Note:** For aggregate mappings the **Reference Descriptor** must be an *aggregate*. See Section 23.6, "Configuring a Relational Descriptor as a Class or Aggregate Type" for more information.

For variable one-to-one mappings, the Reference Descriptor must be an *interface*. See Chapter 32, "Configuring a Relational Variable One-to-One Mapping" for more information.

---

You can specify a reference descriptor that is not in the current TopLink Workbench project. For example, to create a mapping to an `Employee` class that does not exist in the current project, do the following:

1. Add the `Employee` class to your current project. See [Section 116.2, "Working with Projects"](#).
2. Create the relationship mapping to the `Employee` descriptor.
3. Deactivate the `Employee` descriptor. See [Active and Inactive Descriptors](#).

When you generate the deployment XML for your project, the *mapping* to the `Employee` class will be included, but not the `Employee` class.

## 28.5 Configuring Batch Reading

Batch reading can be used in most of the relational mappings. This feature should be used only if it is known that the related objects are always required with the source object.

[Table 28–7](#) summarizes which relational mappings support this option.

**Table 28–7 Relational Mapping Support for Batch Reading**

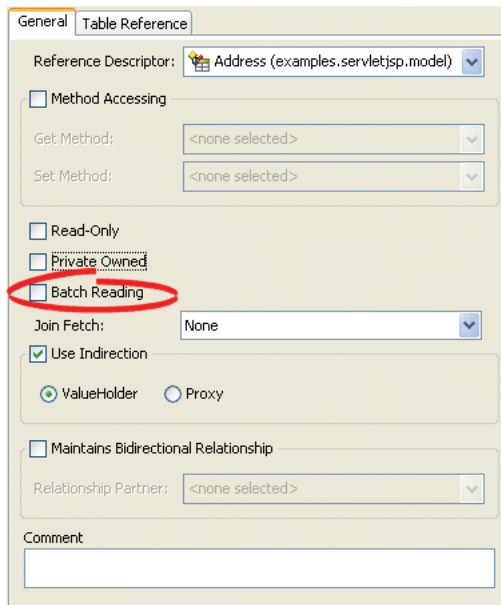
Mapping	How to Use Oracle JDeveloper	How to Configure Batch Reading Using TopLink Workbench	How to Configure Batch Reading Using Java
One-to-one (see <a href="#">Section 27.5, "One-to-One Mapping"</a> )	✓	✓	✓
One-to-many (see <a href="#">Section 27.7, "One-to-Many Mapping"</a> )	✓	✓	✓
Many-to-many (see <a href="#">Section 27.8, "Many-to-Many Mapping"</a> )	✓	✓	✓
Direct collection (see <a href="#">Section 27.10, "Direct Collection Mapping"</a> )	✓	✓	✓
Direct map (see <a href="#">Section 27.11, "Direct Map Mapping"</a> )	✓	✓	✓
Aggregate object (see <a href="#">Section 27.12, "Aggregate Object Mapping"</a> )			✓

### 28.5.1 How to Configure Batch Reading Using TopLink Workbench

To use batch reading in a relationship mapping, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

**Figure 28–3 General Tab, Batch Reading Option**



To specify that this mapping using batch reading, select the **Batch Reading** option.

### 28.5.2 How to Configure Batch Reading Using Java

[Example 28–1](#) shows how to use a `DescriptorCustomizer` class to add batch reading to a mapping.

**Example 28–1 Query Optimization Using Batching**

```
public void customize(ClassDescriptor descriptor) {
    OneToManyMapping phoneNumbersMapping = new OneToManyMapping();
    phoneNumbersMapping =
        (OneToManyMapping) descriptor.getMappingForAttributeName("phones");
    phoneNumbersMapping.useBatchReading();

    // add mapping to descriptor
    descriptor.addMapping(phoneNumbersMapping);
}
```

## 28.6 Configuring Query Key Order

You can configure `TopLink` to maintain collections in order by query key.

[Table 28–8](#) summarizes which relational mappings support this option.

**Table 28–8 Relational Mapping Support for Query Key Order**

Mapping	How to Use Oracle JDeveloper	How to Configure Query Key Order Using TopLink Workbench	How to Configure Query Key Order Using Java
Many-to-many (see <a href="#">Section 27.8, "Many-to-Many Mapping"</a> )	✓	✓	✓

**Table 28–8 (Cont.) Relational Mapping Support for Query Key Order**

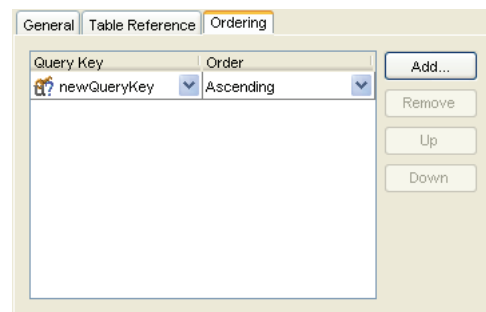
Mapping	How to Use Oracle JDeveloper	How to Configure Query Key Order Using TopLink Workbench	How to Configure Query Key Order Using Java
One-to-many (see <a href="#">Section 27.7</a> , "One-to-Many Mapping")	✓	✓	✓
Aggregate collection (see <a href="#">Section 27.9</a> , "Aggregate Collection Mapping")			✓

### 28.6.1 How to Configure Query Key Order Using TopLink Workbench

To specify the order of a mapping’s query keys, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **Ordering** tab. The Ordering tab appears.

**Figure 28–4 Ordering Tab**



Field	Description
Query Key	Specify the query key to order by. Click <b>Add</b> to add query keys to, or <b>Remove</b> to remove query keys from the ordering operation. Click <b>Up</b> or <b>Down</b> to change the sort order of selected query keys.
Order	Specify if TopLink orders the selected query key in <b>Ascending</b> or <b>Descending</b> (alphabetical) order.

### 28.6.2 How to Configure Query Key Order Using Java

[Example 28–2](#) shows how to use the `DescriptorCustomizer` class to add complex ordering to a mapping.

**Example 28–2 Configuring Query Key Order**

```
public void customize(ClassDescriptor descriptor) {

    OneToManyMapping phoneNumbersMapping = new OneToManyMapping();

    phoneNumbersMapping =
        (OneToManyMapping) descriptor.getMappingForAttributeName("phones");
}
```

```
phoneNumbersMapping.addAscendingOrdering("areaCode");

ExpressionBuilder phone =
    phoneNumbersMapping.getSelectionQuery().getExpressionBuilder();

phoneNumbersMapping.getSelectionQuery().addOrdering(
    phone.get("type").toUpperCase().ascending());

// add mapping to descriptor
descriptor.addMapping(phoneNumbersMapping);
}
```

---

**Note:** You can provide the same functionality by using a descriptor amendment method (see [Section 13.6, "Using the Descriptor Amendment Methods"](#)).

---

## 28.7 Configuring Table and Field References (Foreign and Target Foreign Keys)

A foreign key is a combination of one or more database columns that reference a unique key, usually the primary key, in another table. Foreign keys can be any number of fields (similar to a primary key), all of which are treated as a unit. A foreign key and the parent key it references must have the same number and type of fields.

Mappings that extend `oracle.toplink.mappings.ForeignReferenceMapping` use foreign keys to find information in the database so that the target object(s) can be instantiated. For example, if every `Employee` has an attribute `address` that contains an instance of `Address` (which has its own descriptor and table) then, the one-to-one mapping for the `address` attribute would specify foreign key information to find an `Address` for a particular `Employee`.

`TopLink` classifies foreign keys into two categories in mappings—**foreign keys** and **target foreign keys**:

- In a *foreign key*, the key is found in the table associated with the mapping's own descriptor. For example, an `Employee` foreign key to `ADDRESS` would be in the `EMPLOYEE` table.
- In a *target foreign key*, the reference is from the target object's table back to the key from the mapping's descriptor's table. For example, the `ADDRESS` table would have a foreign key to `EMPLOYEE`.

---

**Caution:** Make sure you fully understand the distinction between *foreign key* and *target foreign key* before defining a mapping.

---

The table reference is the database table that contains the foreign key references.

[Table 28–9](#) summarizes which relational mappings support this option.



**Table 28–9 Relational Mapping Support for Table Reference**

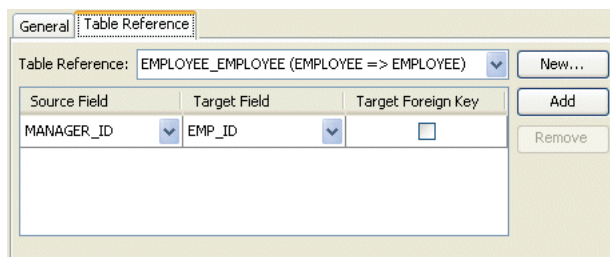
Mapping	How to Use Oracle JDeveloper	How to Configure Table and Field References (Foreign and Target Foreign Keys) Using TopLink Workbench	How to Configure Table and Field References (Foreign and Target Foreign Keys) Using Java
One-to-one (see Section 27.5, "One-to-One Mapping")	✓	✓	✓
One-to-many (see Section 27.7, "One-to-Many Mapping")	✓	✓	✓
Many-to-many (see Section 27.8, "Many-to-Many Mapping")	✓	✓	✓
Aggregate collection (see Section 27.9, "Aggregate Collection Mapping")			✓
Direct collection (see Section 27.10, "Direct Collection Mapping")	✓	✓	✓
Direct map (see Section 27.11, "Direct Map Mapping")	✓	✓	✓

Using TopLink Workbench, you can either import this table from your database or create it. If you import tables from the database (see Section 5.5.1.3, "Importing Tables from a Database"), TopLink creates references that correspond to existing database constraints (if supported by the driver). You can also define references in TopLink without creating similar constraints on the database.

### 28.7.1 How to Configure Table and Field References (Foreign and Target Foreign Keys) Using TopLink Workbench

To specify a table for a mapping reference, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **Table Reference** tab. The Reference tab appears.

**Figure 28–5 Table Reference Tab, Table Reference Field**


Use the following information to select the field references on the tab:

Field	Description
<b>Table Reference</b>	Select an existing table, or click <b>New</b> to create a new table reference.
<b>Source and Target Field</b>	Click <b>Add</b> to create new foreign key reference. To delete an existing key pair reference, select the <b>Source</b> and <b>Target</b> fields and click <b>Remove</b> .
<b>Source Field</b>	Select the database field from the <i>source</i> table for this foreign key reference.
<b>Target Field</b>	Select the database field from the <i>target</i> table for this foreign key reference.
<b>Target Foreign Key</b>	Specify whether or not the reference is from the target object's table back to the key from the mapping's descriptor's table.

## 28.7.2 How to Configure Table and Field References (Foreign and Target Foreign Keys) Using Java

Use the `addTargetForeignKeyFieldName` method (and pass the name of the field name of the target foreign key and the source of the primary key in the source table) to specify foreign key information.

For composite source primary keys, use the `addTargetForeignKeyFieldName` method for each of the fields that comprise the primary key.

[Example 28–3](#) shows how to use the `DescriptorCustomizer` class to add complex join to a mapping.

### **Example 28–3 Adding Complex Join to a Mapping**

```
public void customize(ClassDescriptor descriptor) {

    OneToManyMapping phoneNumbersMapping = new OneToManyMapping();
    phoneNumbersMapping =
        (OneToManyMapping) descriptor.getMappingForAttributeName("cellPhones");

    ExpressionBuilder phone =
        phoneNumbersMapping.getSelectionQuery().getExpressionBuilder();

    phoneNumbersMapping.addTargetForeignKeyFieldName("PHONE.EMP_ID", "EMP.ID");

    phoneNumbersMapping.getSelectionQuery(
        phone.getField("PHONE.EMP_ID").equal(phone.getParameter("EMP.ID").
            and(phone.getField("PHONE.TYPE").equal("CELL")));

    // add mapping to descriptor
    descriptor.addMapping(phoneNumbersMapping);
}
```

---

**Note:** You can provide the same functionality by using a descriptor amendment method (see [Section 13.6, "Using the Descriptor Amendment Methods"](#)).

---

## 28.8 Configuring Joining at the Mapping Level

TopLink supports configuring an inner or outer join at the mapping level for a `ForeignReferenceMapping`. When a class that owns the mapping is read, the TopLink runtime will always get the class and the target of the reference mapping with one database hit.

Use this feature only if the target object is *always* required with the source object, or when indirection (lazy loading) is *not* used. For more information, see [Section 17.2.4, "Indirection \(Lazy Loading\)"](#).

You can also configure join reading at the query level. For more information, see [Section 108.7.1.5, "Join Reading and Object-Level Read Queries"](#).

You can use Oracle JDeveloper, TopLink Workbench, or Java to configure joining at the mapping level.

For more information about joins, see [Section 110.2.7, "Expressions for Joining and Complex Relationships"](#).

### 28.8.1 How to Configure Joining at the Mapping Level Using TopLink Workbench

To use joining in a relationship mapping, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

**Figure 28–6** General Tab, Use Joining Option

The screenshot shows the 'General' tab of the 'Table Reference' configuration in TopLink Workbench. The 'Reference Descriptor' is set to 'Address (examples.servletjsp.model)'. The 'Join Fetch' dropdown is highlighted with a red circle and is currently set to 'None'. Other options include 'Method Accessing', 'Read-Only', 'Private Owned', 'Batch Reading', 'Use Indirection' (checked), 'ValueHolder' (selected), 'Proxy', and 'Maintains Bidirectional Relationship'.

To use joining with this relationship, select the appropriate join-fetch method:

- **Inner**
- **Outer**
- **None** (default)

## 28.8.2 How to Configure Joining at the Mapping Level Using Java

[Example 28–4](#) shows how to use the `DescriptorCustomizer` class to add complex join at the mapping level.

### **Example 28–4 Adding Join at the Mapping Level**

```
public void customize(ClassDescriptor descriptor) {  
  
    OneToManyMapping addressMapping = new OneToManyMapping();  
    addressMapping =  
        (OneToManyMapping)descriptor.getMappingForAttributeName("address");  
    addressMapping.useJoining();  
    ...  
    // add mapping to descriptor  
    descriptor.addMapping(addressMapping);  
}
```

---

---

**Note:** You can provide the same functionality by using a descriptor amendment method (see [Section 13.6, "Using the Descriptor Amendment Methods"](#)).

---

---

## Configuring a Relational Direct-to-Field Mapping

This chapter describes the various components that you must configure in order to use a relational direct-to-field mapping.

This chapter includes the following section:

- [Introduction to Relational Direct-to-Field Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 29.1 Introduction to Relational Direct-to-Field Mapping Configuration

[Table 29–1](#) lists the configurable options for a relational direct-to-field mapping.

**Table 29–1** Configurable Options for Relational Direct-to-Field Mapping

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Database field (see <a href="#">Section 28.3, "Configuring a Database Field"</a> )	✓	✓	✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Default null value (see <a href="#">Section 121.5, "Configuring a Default Null Value at the Mapping Level"</a> )	✓	✓	✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Mapping comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	
Serialized object converter (see <a href="#">Section 121.9, "Configuring a Serialized Object Converter"</a> )	✓	✓	✓
Type conversion converter (see <a href="#">Section 121.10, "Configuring a Type Conversion Converter"</a> )	✓	✓	✓
Object type converter (see <a href="#">Section 121.11, "Configuring an Object Type Converter"</a> )	✓	✓	✓

[Example 29–1](#) shows how to create a direct-to-field mapping and add it to a descriptor using Java code.

**Example 29–1 Direct-to-Field Mapping**

```
public void customize(ClassDescriptor descriptor) {
    DirectToFieldMapping mapping = new DirectToFieldMapping();

    // configure mapping
    ...

    // add mapping to descriptor
    descriptor.addMapping(mapping);
}
```

For more information, see the following:

- [Section 27.3, "Direct-to-Field Mapping"](#)
- [Chapter 28, "Configuring a Relational Mapping"](#)
- [Part XXVIII, "Creation and Configuration of Mappings"](#)

## Configuring a Relational Direct-to-XMLType Mapping

This chapter describes the various components that you must configure in order to use a relational direct-to-XMLType mapping.

This chapter includes the following sections:

- [Introduction to Relational Direct-to-XMLType Mapping](#)
- [Configuring Read Whole Document](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 30.1 Introduction to Relational Direct-to-XMLType Mapping

[Table 30–1](#) lists the configurable options for a relational direct-to-XMLType mapping.

**Table 30–1** Configurable Options for Relational Direct-To-XMLType Mapping

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Database field (see <a href="#">Section 28.3, "Configuring a Database Field"</a> )	✓	✓	✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Read whole document (see <a href="#">Section 30.2, "Configuring Read Whole Document"</a> )	✓	✓	✓
Mapping comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	

[Example 30–1](#) shows how to create a direct-to-XMLType mapping and add it to a descriptor using Java code.

**Example 30–1** Direct-to-XMLType Mapping

```
public void customize(ClassDescriptor descriptor) {
    DirectToXMLTypeMapping mapping = new DirectToXMLTypeMapping();

    // configure mapping
    ...
}
```

```
// add mapping to descriptor
descriptor.addMapping(mapping);
}
```

For more information, see the following:

- [Section 27.4, "Direct-to-XMLType Mapping"](#)
- [Chapter 28, "Configuring a Relational Mapping"](#)
- [Part XXVIII, "Creation and Configuration of Mappings"](#)

## 30.2 Configuring Read Whole Document

When mapping an XML Type to a Document Object Model (DOM), by default TopLink uses the database representation of the DOM. This allows for lazy loading of the XML data from the database.

However, if you require the entire DOM, (or if you require the DOM to be available in a disconnected fashion from the database connection) use the **Read Whole** option to retrieve the entire DOM from the database.

### 30.2.1 How to Configure Read Whole Document Using TopLink Workbench

To specify that this mapping reads the whole XML document, use this procedure:

1. Select the mapping in the **Navigator**. Its properties appear in the Editor.
2. Click **General**. The General tab appears.

**Figure 30–1** *Direct to XML Mapping Property Sheet, Read Whole Document Option*

The screenshot shows a property sheet for a Direct to XML Mapping. The 'Database Field' is 'ADDRESS.CITY (VARCHAR2)'. Under 'Method Accessing', 'Get Method' and 'Set Method' are both '<none selected>'. The 'Read-Only' checkbox is unchecked. The 'Read Whole Document' checkbox is checked and circled in red. There is a 'Comment' field at the bottom.

Choose the **Read Whole Document** option to read the whole XML document. If you do not select this option, the connection must remain open for TopLink to read the database values.

### 30.2.2 How to Configure Read Whole Document Using Java

Use the following `DirectToXMLTypeMapping` methods:

- `setShouldReadWholeDocument`
- `shouldReadWholeDocument`

For more information about the available methods for `DirectToXMLTypeMapping`, see the *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.



## Configuring a Relational One-to-One Mapping

This chapter describes the various components that you must configure in order to use a relational one-to-one mapping.

This chapter includes the following sections:

- [Introduction to Relational One-to-One Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 31.1 Introduction to Relational One-to-One Mapping Configuration

[Table 31–1](#) lists the configurable options for a relational one-to-one mapping.

**Table 31–1** Configurable Options for Relational One-to-One Mapping

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Reference descriptor (see <a href="#">Section 28.4, "Configuring Reference Descriptor"</a> )	✓	✓	✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Read-only mapping (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Private or Independent relationships (see <a href="#">Section 121.7, "Configuring Private or Independent Relationships"</a> )	✓	✓	✓
Batch reading (see <a href="#">Section 28.5, "Configuring Batch Reading"</a> )	✓	✓	✓
Joining (see <a href="#">Section 28.8, "Configuring Joining at the Mapping Level"</a> )	✓	✓	✓
Indirection (lazy loading) (see <a href="#">Section 121.3, "Configuring Indirection (Lazy Loading)"</a> )	✓	✓	✓
Bidirectional relationship (see <a href="#">Section 121.18, "Configuring Bidirectional Relationship"</a> )	✓	✓	✓
Mapping comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	
Table and field references (see <a href="#">Section 28.7, "Configuring Table and Field References (Foreign and Target Foreign Keys)"</a> )	✓	✓	✓

[Example 31–1](#) shows how to create a one-to-one mapping and add it to a descriptor using Java code.

**Example 31–1 One-to-One Mapping**

```
public void customize(ClassDescriptor descriptor) {
    OneToOneMapping mapping = new OneToOneMapping();

    // configure mapping
    ...

    // add mapping to descriptor
    descriptor.addMapping(mapping);
}
```

For more information, see the following:

- [Section 27.5, "One-to-One Mapping"](#)
- [Chapter 28, "Configuring a Relational Mapping"](#)
- [Part XXVIII, "Creation and Configuration of Mappings"](#)

For information on using JPA to configure one-to-one mappings, see "[@OneToOne](#)" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Introduction\\_to\\_EclipseLink\\_JPA\\_%28ELUG%29#.40OneToOne](http://wiki.eclipse.org/Introduction_to_EclipseLink_JPA_%28ELUG%29#.40OneToOne).

## Configuring a Relational Variable One-to-One Mapping

This chapter describes the various components that you must configure in order to use a relational variable one-to-one mapping.

This chapter includes the following sections:

- Introduction to Relational Variable One-to-One Mapping Configuration
- Configuring Class Indicator
- Configuring Unique Primary Key
- Configuring Query Key Association

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 32.1 Introduction to Relational Variable One-to-One Mapping Configuration

[Table 32-1](#) lists the configurable options for a relational variable one-to-one mapping.

**Table 32-1** Configurable Options for Relational Variable One-to-One Mapping

Option	Oracle JDeveloper	TopLink Workbench	Java
<a href="#">Section 28.4, "Configuring Reference Descriptor"</a>	✓	✓	✓
<a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a>	✓	✓	✓
<a href="#">Section 121.2, "Configuring Read-Only Mappings"</a>	✓	✓	✓
<a href="#">Section 121.7, "Configuring Private or Independent Relationships"</a>	✓	✓	✓
<a href="#">Section 121.3, "Configuring Indirection (Lazy Loading)"</a>	✓	✓	✓
<a href="#">Section 121.8, "Configuring Mapping Comments"</a>	✓	✓	
<a href="#">Section 32.4, "Configuring Query Key Association"</a>	✓	✓	✓
<a href="#">Section 32.2, "Configuring Class Indicator"</a>	✓	✓	✓
<a href="#">Section 32.3, "Configuring Unique Primary Key"</a>	✓	✓	✓

Example 32–1 shows how to create a variable one-to-one mapping and add it to a descriptor using Java code.

**Example 32–1 Variable One-to-One Mapping**

```
public void customize(ClassDescriptor descriptor) {
    VariableOneToOneMapping mapping = new VariableOneToOneMapping();

    // configure mapping
    ...

    // add mapping to descriptor
    descriptor.addMapping(mapping);
}

```

For more information, see the following:

- Section 27.6, "Variable One-to-One Mapping"
- Chapter 28, "Configuring a Relational Mapping"
- Part XXVIII, "Creation and Configuration of Mappings"

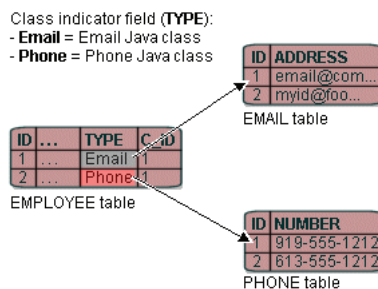
## 32.2 Configuring Class Indicator

In variable one-to-one mappings, you can use an indicator column in the source table to specify the correct target table, as illustrated in Figure 32–1. This section includes information on implementing class indicators.

A source table has an indicator column that specifies the target table through the class indicator field.

Figure 32–1 illustrates the EMPLOYEE table that has a TYPE column that indicates the target for the row (either PHONE or EMAIL).

**Figure 32–1 Variable One-to-One Mapping using Class indicator Field**



The principles of defining such a variable class relationship are similar to defining a normal one-to-one relationship, except:

- The reference class is a Java *interface*, not a Java *class*. However, the method to set the interface is identical.
- You must specify a type indicator field.
- You specify the class indicator values on the mapping so that mapping can determine the class of object to create.
- You must specify the foreign key names and the respective abstract query keys from the target interface descriptor.

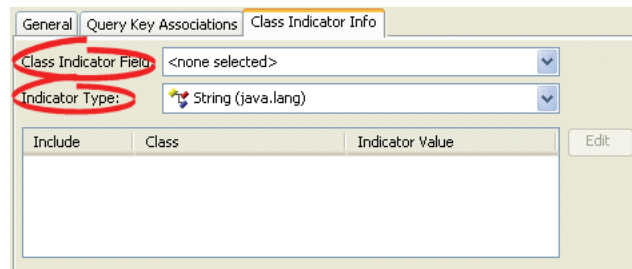
Alternatively, you can use unique primary keys to specify the correct target. See [Section 32.3, "Configuring Unique Primary Key"](#) for more information.

### 32.2.1 How to Configure a Class Indicator Using TopLink Workbench

To specify the class indicators for a variable one-to-one mapping, use this procedure:

1. Select the variable one-to-one mapping in the **Navigator**. Its attributes appear in the Editor.
2. Click the **Class Indicator Info** tab. The Class Indicator Info tab appears.

**Figure 32–2 Class Indicator Info Tab**



Use the **Class Indicator Field** to select the field on the database table (associated with the mapping's descriptor) to use as the indicator for the variable mapping.

Use the **Indicator Type** to specify the data type of the class indicator field by selecting the data type from the list.

To specify the specific class indicator field values for each (nonabstract) child class, click **Edit** and enter the appropriate value for each child class.

### 32.2.2 How to Configure a Class Indicator Using Java

[Example 32–2](#) illustrates how to define a variable one-to-one mapping using a class (type) indicator in Java code.

**Example 32–2 Defining Variable One-to-One Mapping Using a Class Indicator**

```
public void customize(ClassDescriptor descriptor) {
    VariableOneToOneMapping variableOneToOneMapping =
        new VariableOneToOneMapping();
    variableOneToOneMapping.setAttributeName("contact");
    variableOneToOneMapping.setReferenceClass(Contact.class);
    variableOneToOneMapping.setForeignQueryKeyName("C_ID", "id");
    variableOneToOneMapping.setTypeFieldName("TYPE");

    // configure class indicators
    variableOneToOneMapping.addClassIndicator(Email.class, "Email");
    variableOneToOneMapping.addClassIndicator(Phone.class, "Phone");

    variableOneToOneMapping.dontUseIndirection();
    variableOneToOneMapping.privateOwnedRelationship();

    // add mapping to descriptor
    descriptor.addMapping(variableOneToOneMapping);
}
```

For more information about the available methods for `VariableOneToOneMapping`, see the *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.

## 32.3 Configuring Unique Primary Key

In variable one-to-one mappings, you can use a unique primary key in the source table to specify the correct target table, as illustrated in [Figure 32-4](#). This section includes information on implementing class indicators.

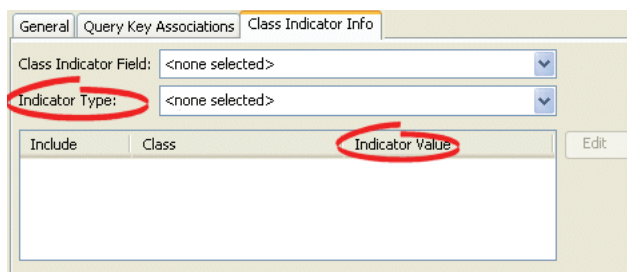
Alternatively, you can use a class indicator to specify the correct target. See [Section 32.2, "Configuring Class Indicator"](#) for more information.

### 32.3.1 How to Configure a Unique Primary Key Using TopLink Workbench

To specify the variable one-to-one mapping with a primary key, use this procedure:

1. Select the variable one-to-one mapping in the **Navigator**. Its attributes appear in the Editor.
2. Click the **Class Indicator Info** tab. The Class Indicator Info tab appears.

**Figure 32-3** Class Indicator Info Tab, Configuring Primary Key



Use the **Class Indicator Field** to select none.

Use the **Indicator Type** to select none.

Use the **Indicator Value** column to specify none.

After choosing the reference descriptor for the mapping, deselect the **Include** check boxes.

---

**Note:** You cannot deselect the value in the Class Indicator Field, unless the foreign key values in the source table are unique.

---

### 32.3.2 How to Configure a Unique Primary Key Using Java

[Example 32-3](#) illustrates how to define a variable one-to-one mapping using a unique primary key in Java code.

**Example 32-3** Defining Variable One-to-One Mapping Using a Unique Primary Key

```
public void customize(ClassDescriptor descriptor) {
    VariableOneToOneMapping variableOneToOneMapping =
        new VariableOneToOneMapping();
    variableOneToOneMapping.setAttributeName("contact");
}
```

```

variableOneToOneMapping.setReferenceClass(Contact.class);
variableOneToOneMapping.setForeignQueryKeyName("C_ID", "id");
variableOneToOneMapping.dontUseIndirection();
variableOneToOneMapping.privateOwnedRelationship();

// add mapping to descriptor
descriptor.addMapping(variableOneToOneMapping);
}

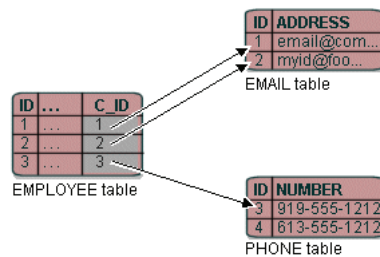
```

For more information about the available methods for `VariableOneToOneMapping`, see the *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.

### 32.3.3 What You May Need to Know About Unique Primary Keys

As [Figure 32–4](#) illustrates, the value of the foreign key in the source table (`C_ID`) mapped to the primary key of the target table is unique. No primary key values among the target tables are the same, so primary key values are not unique just in the table, but also among the tables.

**Figure 32–4 Variable One-to-One Relationship with Unique Primary Key**



If there is no indicator stored in the source table, `TopLink` cannot determine to which target table the foreign key value is mapped. Therefore, `TopLink` reads through all the target tables until it finds an entry in one of the target tables. This is an inefficient way of setting up a relation model. The class indicator is much more efficient as it reduces the number of reads performed on the tables to get the data. In the class indicator method, `TopLink` knows exactly which target table to look into for the data.

The principles of defining such a variable class relationship are similar to defining class indicator variable one-to-one relationships, except the following:

- A type indicator field is not specified.
- The class indicator values are not specified.

The type indicator field and its values are not needed, because `TopLink` goes through all the target tables until data is finally found.

## 32.4 Configuring Query Key Association

This section includes information on configuring query key associations using development tools, as well as Java.

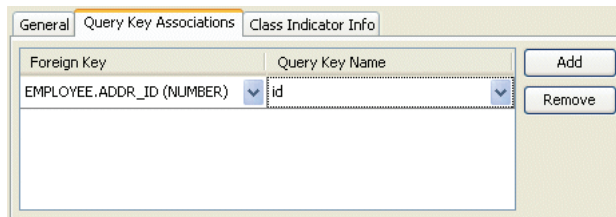
### 32.4.1 How to Configure a Query Key Association Using TopLink Workbench

To specify the query keys used for a variable one-to-one mapping, use this procedure:

1. Select the variable one-to-one mapping in the **Navigator**. Its attributes appear in the Editor.

2. Click the **Query Key Associations** tab. The Query Key Associations tab appears

**Figure 32–5 Query Key Associations Tab**



Use the following information to enter data in each field on the tab:

Use the **Indicator Type** to specify the data type of the class indicator field by selecting the data type from the list.

Field	Description
Foreign Key	The field from the database table to use as the foreign key in this relationship.
Query Key Name	The name of the query key (from the referenced descriptor) for this association. See <a href="#">Section 119.10, "Configuring Query Keys"</a> for more information.

### 32.4.2 How to Configure a Query Key Association Using Java

The API to configure query key associations is `oracle.toplink.mappings.VariableOneToOneMapping` method `addForeignQueryKeyName(String, String)`.

For more information about the available methods for `VariableOneToOneMapping`, see the *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.



## Configuring a Relational One-to-Many Mapping

This chapter describes the various components that you must configure in order to use a relational one-to-many mapping.

This chapter includes the following section:

- [Introduction to Relational One-to-Many Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 33.1 Introduction to Relational One-to-Many Mapping Configuration

[Table 33–1](#) lists the configurable options for a relational one-to-many mapping.

**Table 33–1** Configurable Options for Relational One-to-Many Mapping

Option	Oracle JDeveloper	TopLink Workbench	Java
<a href="#">Reference descriptor (see Section 28.4, "Configuring Reference Descriptor")</a>	✓	✓	✓
<a href="#">Method or direct field access (see Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level")</a>	✓	✓	✓
<a href="#">Read-only mapping (see Section 121.2, "Configuring Read-Only Mappings")</a>	✓	✓	✓
<a href="#">Private or Independent relationships (see Section 121.7, "Configuring Private or Independent Relationships")</a>	✓	✓	✓
<a href="#">Batch reading (see Section 28.5, "Configuring Batch Reading")</a>	✓	✓	✓
<a href="#">Indirection (lazy loading) (see Section 121.3, "Configuring Indirection (Lazy Loading)")</a>	✓	✓	✓
<a href="#">Bidirectional relationship (see Section 121.18, "Configuring Bidirectional Relationship")</a>	✓	✓	✓
<a href="#">Container policy (see Section 121.14, "Configuring Container Policy")</a>	✓	✓	✓
<a href="#">Mapping comments (see Section 121.8, "Configuring Mapping Comments")</a>	✓	✓	
<a href="#">Table and field references (see Section 28.7, "Configuring Table and Field References (Foreign and Target Foreign Keys)")</a>	✓	✓	✓

**Table 33–1 (Cont.) Configurable Options for Relational One-to-Many Mapping**

Option	Oracle JDeveloper	TopLink Workbench	Java
Query key order (see <a href="#">Section 28.6, "Configuring Query Key Order"</a> )	✓	✓	✓

[Example 33–1](#) shows how to create a one-to-many mapping and add it to a descriptor using Java code.

**Example 33–1 One-to-Many Mapping**

```
public void customize(ClassDescriptor descriptor) {
    OneToManyMapping mapping = new OneToManyMapping();

    // configure mapping
    ...

    // add mapping to descriptor
    descriptor.addMapping(mapping);
}
```

For more information, see the following:

- [Section 27.7, "One-to-Many Mapping"](#)
- [Chapter 28, "Configuring a Relational Mapping"](#)
- [Part XXVIII, "Creation and Configuration of Mappings"](#)

For information on using JPA to configure one-to-many mappings, see "[@OneToMany](#)" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Introduction\\_to\\_EclipseLink\\_JPA\\_%28ELUG%29#.40OneToMany](http://wiki.eclipse.org/Introduction_to_EclipseLink_JPA_%28ELUG%29#.40OneToMany).

## Configuring a Relational Many-to-Many Mapping

This chapter describes the various components that you must configure in order to use a relational many-to-many mapping.

This chapter includes the following sections:

- [Introduction to Relational Many-to-Many Mapping Configuration](#)
- [Configuring a Relation Table](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 34.1 Introduction to Relational Many-to-Many Mapping Configuration

[Table 34–1](#) lists the configurable options for a relational many-to-many mapping.

**Table 34–1** Configurable Options for Relational Many-to-Many Mapping

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
<a href="#">Reference descriptor (see Section 28.4, "Configuring Reference Descriptor")</a>	✓	✓	✓
<a href="#">Method or direct field access (see Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level")</a>	✓	✓	✓
<a href="#">Read-only mapping (see Section 121.2, "Configuring Read-Only Mappings")</a>	✓	✓	✓
<a href="#">Private or Independent relationships (see Section 121.7, "Configuring Private or Independent Relationships")</a>	✓	✓	✓
<a href="#">Batch reading (see Section 28.5, "Configuring Batch Reading")</a>	✓	✓	✓
<a href="#">Indirection (lazy loading) (see Section 121.3, "Configuring Indirection (Lazy Loading)")</a>	✓	✓	✓
<a href="#">Bidirectional relationship (see Section 121.18, "Configuring Bidirectional Relationship")</a>	✓	✓	✓
<a href="#">Container policy (see Section 121.14, "Configuring Container Policy")</a>	✓	✓	✓
<a href="#">Mapping comments (see Section 121.8, "Configuring Mapping Comments")</a>	✓	✓	
<a href="#">Relational table (see Section 34.2, "Configuring a Relation Table")</a>	✓	✓	✓

**Table 34–1 (Cont.) Configurable Options for Relational Many-to-Many Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Table and field references (see Section 28.7, "Configuring Table and Field References (Foreign and Target Foreign Keys)") (Source)	✓	✓	✓
Table and field references (see Section 28.7, "Configuring Table and Field References (Foreign and Target Foreign Keys)") (Target)	✓	✓	✓
Query key order (see Section 28.6, "Configuring Query Key Order")	✓	✓	✓

Example 34–1 shows how to create a many-to-many mapping and add it to a descriptor using Java code.

**Example 34–1 Many-to-Many Mapping**

```
public void customize(ClassDescriptor descriptor) {
    ManyToManyMapping mapping = new ManyToManyMapping();

    // configure mapping
    ...

    // add mapping to descriptor
    descriptor.addMapping(mapping);
}

```

For more information, see the following:

- Section 27.8, "Many-to-Many Mapping"
- Chapter 28, "Configuring a Relational Mapping"
- Part XXVIII, "Creation and Configuration of Mappings"

For information on using JPA to configure many-to-many mappings, see "@ManyToMany" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Introduction\\_to\\_EclipseLink\\_JPA\\_%28ELUG%29#.40ManyToMany](http://wiki.eclipse.org/Introduction_to_EclipseLink_JPA_%28ELUG%29#.40ManyToMany).

## 34.2 Configuring a Relation Table

The relation table contains the columns for the primary keys of the source table and target table involved in the many-to-many mapping. You must create this table in the database before completing the mapping. See Section 5.5, "Using Databases" for information on creating database tables.

In Figure 27–5, the PROJ\_EMP table serves as the relation table between the PROJECT and EMPLOYEE tables.

### 34.2.1 How to Configure a Relation Table Using TopLink Workbench

To select a relation table for a mapping, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

**Figure 34–1 Table Reference Tab, Relation Table Option**

The screenshot shows a configuration window with three tabs: 'General', 'Table Reference', and 'Ordering'. The 'Table Reference' tab is active. At the top, there is a 'Relation Table:' dropdown menu with 'PROJ\_EMP' selected, which is circled in red. Below this are two sections: 'Source Reference' and 'Target Reference'. Each section has a 'Table Reference:' dropdown menu and a table with 'Source Column' and 'Target Column' headers. In the 'Source Reference' section, the table contains 'EMP\_ID' in both columns. In the 'Target Reference' section, the table contains 'PROJ\_ID' in both columns. There are 'New...', 'Add', and 'Remove' buttons for each section.

Use the **Relation Table** field to select a database table to define this mapping.

### 34.2.2 How to Configure a Relation Table Using Java

Many-to-many mappings represent the relationships between a collection of source objects and a collection of target objects. This requires an intermediate table that manages the associations between the source and target records.

Many-to-many mappings are instances of the `ManyToManyMapping` class and requires the following elements:

- The attribute mapped, set by using the `setAttributeName` method.
- The reference class, set by using the `setReferenceClass` method.
- The relation table, set by using the `setRelationTableName()` method.
- The foreign key information (for noncomposite target primary keys), which you specify by calling the `setSourceRelationKeyFieldName` and `setTargetRelationKeyFieldName` methods.
- The foreign key information if the source or target primary keys are composite, which you specify by sending the `addSourceRelationKeyFieldName` or `addTargetRelationKeyFieldName` methods.

#### **Example 34–2 Configuring a Relational Table**

```
public void customize(ClassDescriptor descriptor) {
    // In the Employee class, create the mapping that references Project class
    ManyToManyMapping manyToManyMapping = new ManyToManyMapping();
    manyToManyMapping.setAttributeName("projects");
    manyToManyMapping.setReferenceClass(Project.class);

    // Configure the relational table
    manyToManyMapping.setRelationTableName("PROJ_EMP");
    manyToManyMapping.setSourceRelationKeyFieldName("EMPID");
    manyToManyMapping.setTargetRelationKeyFieldName("PROJID");

    // Add mapping to descriptor
}
```

```
        descriptor.addMapping(manyToManyMapping);  
    }
```

In addition to the API that [Example 34–2](#) illustrates, other common API for use with many-to-many mappings include the following:

- `useBasicIndirection`: implements TopLink value holder indirection.
- `useTransparentCollection`: if you use transparent indirection, this element places a special collection in the source object's attribute.
- `dontUseIndirection`: implements no indirection.

For more information about the available methods for `ManyToManyMapping`, see the *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.

## Configuring a Relational Aggregate Collection Mapping

This chapter describes the various components that you must configure in order to use a relational aggregate collection mapping.

---

**Note:** To use a relational aggregate collection mapping with TopLink Workbench, you must use an amendment method (see [Section 119.35, "Configuring Amendment Methods"](#)).

---

This chapter includes the following section:

- [Introduction to Relational Aggregate Collection Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 35.1 Introduction to Relational Aggregate Collection Mapping Configuration

[Table 35–1](#) lists the configurable options for a relational aggregate collection mapping.

**Table 35–1** Configurable Options for Relational Aggregate Collection Mapping

Option	Oracle JDeveloper	TopLink Workbench	Java
Database field (see <a href="#">Section 28.3, "Configuring a Database Field"</a> )			✓
Reference descriptor (see <a href="#">Section 28.4, "Configuring Reference Descriptor"</a> )			✓
Container policy (see <a href="#">Section 121.14, "Configuring Container Policy"</a> )			✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )			✓
Read-only mapping (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )			✓
Batch reading (see <a href="#">Section 28.5, "Configuring Batch Reading"</a> )			✓
Bidirectional relationship (see <a href="#">Section 121.18, "Configuring Bidirectional Relationship"</a> )			✓

**Table 35–1 (Cont.) Configurable Options for Relational Aggregate Collection Mapping**

Option	Oracle JDeveloper	TopLink Workbench	Java
Query key order (see <a href="#">Section 28.6, "Configuring Query Key Order"</a> )			✓
Table and field references (see <a href="#">Section 28.7, "Configuring Table and Field References (Foreign and Target Foreign Keys)"</a> )			✓

[Example 35–1](#) shows how to create an aggregate collection mapping and add it to a descriptor using Java code.

**Example 35–1 Aggregate Collection Mapping**

```
public void customize(ClassDescriptor descriptor) {
    AggregateCollectionMapping mapping = new AggregateCollectionMapping();

    // configure mapping
    ...

    // add mapping to descriptor
    descriptor.addMapping(mapping);
}

```

For more information, see the following:

- [Section 27.9, "Aggregate Collection Mapping"](#)
- [Chapter 28, "Configuring a Relational Mapping"](#)
- [Part XXVIII, "Creation and Configuration of Mappings"](#)



## Configuring a Relational Direct Collection Mapping

This chapter describes the various components that you must configure in order to use a relational direct collection mapping.

This chapter includes the following sections:

- [Introduction to Relational Direct Collection Mapping Configuration](#)
- [Configuring Target Table](#)
- [Configuring Direct Value Field](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 36.1 Introduction to Relational Direct Collection Mapping Configuration

[Table 36–1](#) lists the configurable options for a relational direct collection mapping.

**Table 36–1** Configurable Options for Relational Direct Collection Mapping

Option	Oracle JDeveloper	TopLink Workbench	Java
Target table (see <a href="#">Section 36.2, "Configuring Target Table"</a> )	✓	✓	✓
Direct value field (see <a href="#">Section 36.3, "Configuring Direct Value Field"</a> )	✓	✓	✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Read-only mapping (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Batch reading (see <a href="#">Section 28.5, "Configuring Batch Reading"</a> )	✓	✓	✓
Indirection (lazy loading) (see <a href="#">Section 121.3, "Configuring Indirection (Lazy Loading)"</a> )	✓	✓	✓
Container policy (see <a href="#">Section 121.14, "Configuring Container Policy"</a> )	✓	✓	✓
Mapping comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	
Serialized object converter (see <a href="#">Section 121.9, "Configuring a Serialized Object Converter"</a> )	✓	✓	✓
Type conversion converter (see <a href="#">Section 121.10, "Configuring a Type Conversion Converter"</a> )	✓	✓	✓

**Table 36–1 (Cont.) Configurable Options for Relational Direct Collection Mapping**

Option	Oracle JDeveloper	TopLink Workbench	Java
Object type converter (see Section 121.11, "Configuring an Object Type Converter")	✓	✓	✓
Table and field references (see Section 28.7, "Configuring Table and Field References (Foreign and Target Foreign Keys)")	✓	✓	✓

Example 36–1 shows how to create a direct collection mapping and add it to a descriptor using Java code.

**Example 36–1 Direct Collection Mapping**

```
public void customize(ClassDescriptor descriptor) {
    DirectCollectionMapping mapping = new DirectCollectionMapping();

    // configure mapping
    ...

    // add mapping to descriptor
    descriptor.addMapping(mapping);
}
```

For more information, see the following:

- Section 27.10, "Direct Collection Mapping"
- Chapter 28, "Configuring a Relational Mapping"
- Part XXVIII, "Creation and Configuration of Mappings"

For information on using JPA to configure direct collection mappings, see "How to Use the @BasicCollection Annotation" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#How\\_to\\_Use\\_the\\_.40BasicCollection\\_Annotation](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#How_to_Use_the_.40BasicCollection_Annotation).

## 36.2 Configuring Target Table

Each direct collection stores reference information in a target table. In Figure 27–6, the RESPONS table contains the primary key and object of the instance owning the collection. You must create this table in your database.

### 36.2.1 How to Configure a Target Table Using TopLink Workbench

To specify the direct collection specifics, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

**Figure 36–1 General Tab, Target Table Options**

The screenshot shows a configuration window with three tabs: 'General', 'Converter', and 'Table Reference'. The 'General' tab is active. The 'Target Table' dropdown menu is highlighted with a red circle. Below it is the 'Direct Value Field' dropdown. There are two checkboxes: 'Method Accessing' (unchecked) and 'Read-Only' (unchecked). Below these are two dropdown menus for 'Get Method' and 'Set Method'. There are two more checkboxes: 'Batch Reading' (unchecked) and 'Use Indirection' (checked). Below these are two radio buttons: 'ValueHolder' (selected) and 'Transparent' (unselected). At the bottom, there is an 'Advanced...' button and a 'Comment' text area.

Use the **Target Table** list to select the table that contains the reference fields for the direct collection mapping.

## 36.2.2 How to Configure a Target Table Using Java

Direct collection mappings store collections of Java objects that are not TopLink-enabled. Direct collections usually store Java types, such as `String`.

Direct collection mappings are instances of the `DirectCollectionMapping` class and require the following elements:

- The attribute mapped, set by using the `setAttributeName` method.
- The database table that holds the values to be stored in the collection, set by using the `setReferenceTableName` method.
- The field in the reference table from which the values are read and placed into the collection; this is called the direct field. Set it using the `setDirectFieldName` method.
- The foreign key information, which you specify using the `setReferenceKeyFieldName` method and passing the name of the field that is a foreign reference to the primary key of the source object

---

**Note:** If the target primary key is composite, call the `addReferenceKeyFieldName` method for each of the fields that make up the key.

---

### Example 36–2 Configuring a Simple Direct Collection Mapping

```
public void customize(ClassDescriptor descriptor) {
    DirectCollectionMapping directCollectionMapping =
        new DirectCollectionMapping();
    directCollectionMapping.setAttributeName ("responsibilitiesList");
    directCollectionMapping.setReferenceTableName ("RESPONS"); // target table
    directCollectionMapping.setDirectFieldName ("DESCRIP");
    directCollectionMapping.setReferenceKeyFieldName ("EMP_ID");
    directCollectionMapping.useCollectionClass (Collection.class); // default
}
```

```
// add this mapping to descriptor
descriptor.addMapping (directCollectionMapping);
}
```

In addition to the API that [Example 36–2](#) illustrates, other common API for use with direct collection mappings include the following:

- `useBasicIndirection`: implements `TopLink` value holder indirection.
- `useTransparentCollection`: if you use transparent indirection, this element places a special collection in the source object's attribute.
- `dontUseIndirection`: implements no indirection.

For more information about the available methods for `DirectCollectionMapping`, see the *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.

## 36.3 Configuring Direct Value Field

The direct value field, located in the reference table, stores the primitive data value. In [Figure 27–6](#), the `DESCRIP` field stores the collection.

### 36.3.1 How to Configure a Direct Value Field Using TopLink Workbench

To specify the direct collection specifics, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

**Figure 36–2** General Tab, Direct Value Field

The screenshot shows the 'General' tab of the configuration editor. It features several dropdown menus and checkboxes. The 'Direct Value Field' dropdown is highlighted with a red circle. Below it are checkboxes for 'Method Accessing', 'Read-Only', and 'Batch Reading'. There are also radio buttons for 'ValueHolder' (selected) and 'Transparent'. At the bottom, there is an 'Advanced...' button and a 'Comment' text area.

Use the **Direct Value Field** list to select the field from the **Target Table** table that contains the object of the collection.

### 36.3.2 How to Configure Direct Value Field Using Java

[Example 36–2](#), "Configuring a Simple Direct Collection Mapping" demonstrates how to create and configure a direct collection mapping, including the setting of a direct field. The example also shows how to add this mapping to the descriptor.

## Configuring a Relational Aggregate Object Mapping

This chapter describes the various components that you must configure in order to use a relational aggregate object mapping.

---

**Note:** You configure the relational aggregate object mapping in the source object's descriptor. However, before doing so, you must designate the target object's descriptor as an aggregate (see [Section 23.6, "Configuring a Relational Descriptor as a Class or Aggregate Type"](#)).

---

This chapter includes the following sections:

- [Introduction to Relational Aggregate Object Mapping Configuration](#)
- [Configuring Aggregate Fields](#)
- [Configuring Allowing Null Values](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 37.1 Introduction to Relational Aggregate Object Mapping Configuration

[Table 37-1](#) lists the configurable options for a relational aggregate object mapping.

**Table 37-1** Configurable Options for Relational Aggregate Object Mapping

Option	Oracle JDeveloper	TopLink Workbench	Java
Reference descriptor (see <a href="#">Section 28.4, "Configuring Reference Descriptor"</a> )	✓	✓	✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Read-only mapping (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Allowing null values (see <a href="#">Section 37.3, "Configuring Allowing Null Values"</a> )	✓	✓	✓
Mapping comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	

**Table 37–1 (Cont.) Configurable Options for Relational Aggregate Object Mapping**

Option	Oracle JDeveloper	TopLink Workbench	Java
Aggregate fields (see <a href="#">Section 37.2, "Configuring Aggregate Fields"</a> )	✓	✓	✓

[Example 37–1](#) shows how to create an aggregate object mapping and add it to a descriptor using Java code.

**Example 37–1 Aggregate Object Mapping**

```
public void customize(ClassDescriptor descriptor) {
    AggregateObjectMapping mapping = new AggregateObjectMapping();

    // configure mapping
    ...

    // add mapping to descriptor
    descriptor.addMapping(mapping);
}
```

For more information, see the following:

- [Section 27.12, "Aggregate Object Mapping"](#)
- [Chapter 28, "Configuring a Relational Mapping"](#)
- [Part XXVIII, "Creation and Configuration of Mappings"](#)

## 37.2 Configuring Aggregate Fields

When you designate a descriptor as an aggregate, TopLink allows you to specify a mapping type for each field in the target class, but defers associating the field with a database table until you configure the aggregate object mapping in the source class descriptor. In other words, the target class descriptor defines how each target class field is mapped but the source class descriptor defines where each target class field is mapped.

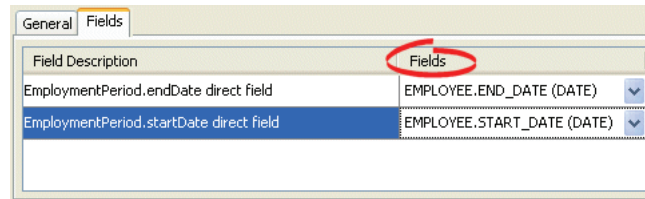
This section explains how to configure the source class descriptor to define where each target class field is mapped.

For more information on how to configure the target class descriptor to define how each target class field is mapped, see [Section 23.6, "Configuring a Relational Descriptor as a Class or Aggregate Type"](#).

### 37.2.1 How to Configure Aggregate Fields Using TopLink Workbench

To specify the mapped fields of an aggregate mapping, use this procedure.

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **Fields** tab. The Fields tab appears.

**Figure 37–1 Fields Tab**

Use the following information to complete each field on the tab:

Field	Description
Field Description	This column shows the name of the fields from the target object, whose descriptor is designated as an aggregate (see <a href="#">Section 23.6, "Configuring a Relational Descriptor as a Class or Aggregate Type"</a> ). These are for display only and cannot be changed.
Fields	Use this column to select the source object database table field that TopLink will map to the corresponding target object field.

### 37.2.2 How to Configure Aggregate Fields Using Java

Using the `AggregateObjectMapping` method `addFieldNameTranslation` you can set a field name translation that maps from a field name in the source table to a field name in the aggregate descriptor

For more information about the available methods for `AggregateObjectMapping`, see the *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.

## 37.3 Configuring Allowing Null Values

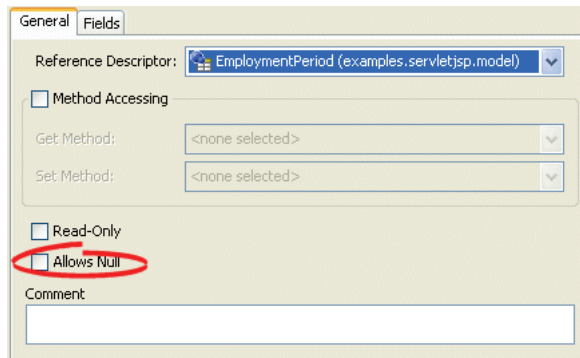
If all the fields in the database row for the aggregate object are `null`, then, by default, TopLink places `null` in the appropriate source object, as opposed to filling an aggregate object with `null` values.

### 37.3.1 How to Configure Allowing Null Values Using TopLink Workbench

To allow a mapping to contain a null value, use this procedure.

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

**Figure 37–2 General Tab, Allow Null Option**



Select the **Allows Null** option to allow this mapping to contain a null value.

### 37.3.2 How to Configure Allowing Null Values Using Java

You can configure whether or not to allow null values using the `AggregateObjectMapping` methods `allowNull` and `dontAllowNull`.

For more information about the available methods for `AggregateObjectMapping`, see the *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.



## Configuring a Relational Direct Map Mapping

This chapter describes the various components that you must configure in order to use a relational direct map mapping.

This chapter includes the following sections:

- [Introduction to Relational Direct Map Mapping Configuration](#)
- [Configuring Direct Value Field](#)
- [Configuring Direct Key Field](#)
- [Configuring Key Converters](#)
- [Configuring Value Converters](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 38.1 Introduction to Relational Direct Map Mapping Configuration

Table 38–1 lists the configurable options for a relational direct map mapping.

**Table 38–1 Configurable Options for Relational Direct Map Mapping**

Option	Oracle JDeveloper	TopLink Workbench	Java
Target table (see <a href="#">Section 36.2, "Configuring Target Table"</a> )	✓	✓	✓
Direct value field (see <a href="#">Section 38.2, "Configuring Direct Value Field"</a> )	✓	✓	✓
Direct key field (see <a href="#">Section 38.3, "Configuring Direct Key Field"</a> )	✓	✓	✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Read-only mapping (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Batch reading (see <a href="#">Section 28.5, "Configuring Batch Reading"</a> )	✓	✓	✓
Indirection (lazy loading) (see <a href="#">Section 121.3, "Configuring Indirection (Lazy Loading)"</a> )	✓	✓	✓
Container policy (see <a href="#">Section 121.14, "Configuring Container Policy"</a> )	✓	✓	✓
Mapping comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	

**Table 38–1 (Cont.) Configurable Options for Relational Direct Map Mapping**

Option	Oracle JDeveloper	TopLink Workbench	Java
Key converters (see Section 38.4, "Configuring Key Converters")	✓	✓	✓
Value converters (see Section 38.5, "Configuring Value Converters")	✓	✓	✓
Table and field references (see Section 28.7, "Configuring Table and Field References (Foreign and Target Foreign Keys)")	✓	✓	✓

Example 38–1 shows how to create a direct map mapping and add it to a descriptor using Java code.

**Example 38–1 Direct Map Mapping**

```
public void customize(ClassDescriptor descriptor) {
    DirectMapMapping mapping = new DirectMapMapping();

    // configure mapping
    ...

    // add mapping to descriptor
    descriptor.addMapping(mapping);
}
```

For more information, see the following:

- Section 27.11, "Direct Map Mapping"
- Chapter 28, "Configuring a Relational Mapping"
- Part XXVIII, "Creation and Configuration of Mappings"

For information on using JPA to configure direct map mappings, see "How to Use the @BasicMap Annotation" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#How\\_to\\_Use\\_the\\_.40BasicMap\\_Annotation](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#How_to_Use_the_.40BasicMap_Annotation).

## 38.2 Configuring Direct Value Field

The direct value field in the reference table stores the primitive data value of the map value. If the value's object value and database value are different types, use a converter (see Section 38.5, "Configuring Value Converters").

### 38.2.1 How to Configure Direct Value Fields Using TopLink Workbench

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

**Figure 38–1 General Tab, Direct Value Field**

The screenshot shows a configuration window with three tabs: 'General', 'Converter', and 'Table Reference'. The 'General' tab is active. It contains the following elements:

- Target Table:** A dropdown menu with '<none selected>' selected.
- Direct Value Field:** A dropdown menu with '<none selected>' selected, circled in red.
- Direct Key Field:** A dropdown menu with '<none selected>' selected.
- Method Accessing:** A checkbox that is unchecked.
- Get Method:** A dropdown menu with '<none selected>' selected.
- Set Method:** A dropdown menu with '<none selected>' selected.
- Read-Only:** A checkbox that is unchecked.
- Batch Reading:** A checkbox that is unchecked.
- Use Indirection:** A checkbox that is checked.
- ValueHolder:** A radio button that is selected.
- Transparent:** A radio button that is unselected.
- Advanced...:** A button.
- Comment:** A text input field.

Use the **Direct Value Field** list to select the field from the **Target Table** table that contains the object of the direct map mapping.

## 38.2.2 How to Configure Direct Value Fields Using Java

Use the `DirectMapMapping` method `setDirectFieldName` to set the direct fields for your mapping.

For more information about the available methods for `DirectMapMapping`, see the *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.

## 38.3 Configuring Direct Key Field

The direct key field in the reference table stores the primitive data value of the map key. If the key's object value and database value are different types, use a converter (see [Section 38.4, "Configuring Key Converters"](#)).

### 38.3.1 How to Configure Direct Key Field Using TopLink Workbench

To specify the direct key field in the reference table, use this procedure.

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

**Figure 38–2 General Tab, Direct Key Field**

The screenshot shows a configuration window with three tabs: 'General', 'Converter', and 'Table Reference'. The 'General' tab is active. It contains several dropdown menus and checkboxes. The 'Direct Key Field' dropdown is circled in red. Below it are checkboxes for 'Method Accessing', 'Read-Only', and 'Batch Reading'. There are also checkboxes for 'Use Indirection' (checked) and radio buttons for 'ValueHolder' (selected) and 'Transparent'. At the bottom, there is an 'Advanced...' button and a 'Comment' text area.

Use the **Direct Key Field** list to select the key from the **Target Table** table that contains the object of the direct map mapping.

### 38.3.2 How to Configure Direct Key Field Using Java

Use the `DirectMapMapping` method `setDirectKeyFieldName` to set the direct key field for your mapping.

For more information about the available methods for `DirectMapMapping`, see the *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.

## 38.4 Configuring Key Converters

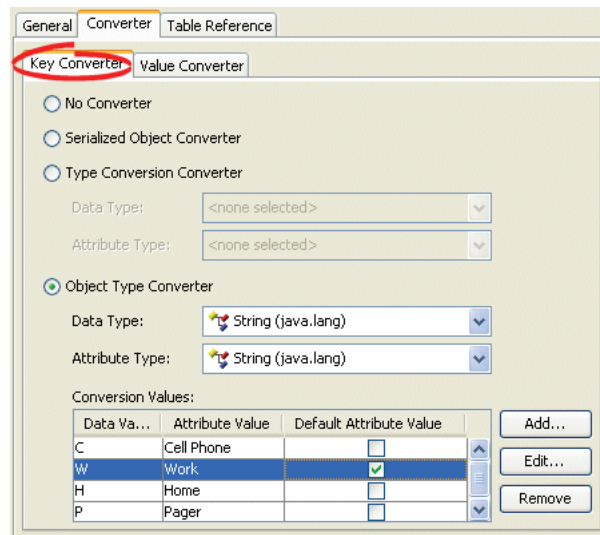
If the key's object value and database value are different types, use a converter. TopLink supports the following key converters:

- [Section 17.2.6.1, "Serialized Object Converter"](#)
- [Section 17.2.6.2, "Type Conversion Converter"](#)
- [Section 17.2.6.3, "Object Type Converter"](#)

### 38.4.1 How to Configure Key Converters Using TopLink Workbench

Use this procedure to specify the converter for a direct map mapping key:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **Converter** tab. The Converter tab appears.
3. Click the **Key Converter** tab. The Key Converter tab appears.

**Figure 38–3 Converter Tab, Key Converter Subtab**

Converter	Description
No Converter	Do not use a Key Converter for this mapping.
Serialized Object Converter	See <a href="#">Section 121.9</a> , "Configuring a Serialized Object Converter".
Type Conversion Converter	See <a href="#">Section 121.10</a> , "Configuring a Type Conversion Converter".
Object Type Converter	See <a href="#">Section 121.11</a> , "Configuring an Object Type Converter".

### 38.4.2 How to Configure Key Converters Using Java

You can configure whether or not to allow null values using the `DirectMapMapping` method `setKeyConverter`.

For more information about the available methods for `DirectMapMapping`, see the *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.

## 38.5 Configuring Value Converters

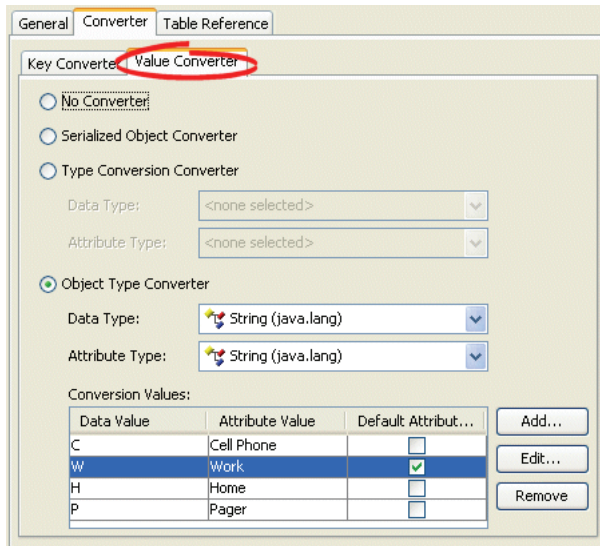
If the value's object value and database value are different types, use a converter. TopLink supports the following value converters:

- [Section 17.2.6.1](#), "Serialized Object Converter"
- [Section 17.2.6.2](#), "Type Conversion Converter"
- [Section 17.2.6.3](#), "Object Type Converter"

### 38.5.1 How to Configure Value Converters Using TopLink Workbench

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **Converter** tab. The Converter tab appears.
3. Click the **Value Converter** tab. The Value Converter tab appears.

**Figure 38-4 Converter Tab, Value Converter Subtab**



Converter	Description
No Converter	Do not use a Value Converter for this mapping.
Serialized Object Converter	See Section 121.9, "Configuring a Serialized Object Converter".
Type Conversion Converter	See Section 121.10, "Configuring a Type Conversion Converter".
Object Type Converter	See Section 121.11, "Configuring an Object Type Converter".

## Configuring a Relational Transformation Mapping

This chapter describes the various components that you must configure in order to use a relational transformation mapping.

This chapter includes the following section:

- [Introduction to Relational Transformation Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 39.1 Introduction to Relational Transformation Mapping Configuration

[Table 39–1](#) lists the configurable options for a relational transformation mapping.

**Table 39–1** Configurable Options for Relational Transformation Mapping

Option	Oracle JDeveloper	TopLink Workbench	Java
Attribute transformer (see <a href="#">Section 121.15, "Configuring Attribute Transformer"</a> )	✓	✓	✓
Field transformer associations (see <a href="#">Section 121.16, "Configuring Field Transformer Associations"</a> )	✓	✓	✓
Indirection (lazy loading) (see <a href="#">Section 121.3, "Configuring Indirection (Lazy Loading)"</a> )	✓	✓	✓
Mutable <a href="#">Section 121.17, "Configuring Mutable Mappings"</a>	✓	✓	✓
Mapping comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Read-only mapping (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓

[Example 39–1](#) shows how to create a transformation mapping and add it to a descriptor using Java code.

**Example 39–1** Transformation Mapping

```
public void customize(ClassDescriptor descriptor) {
    TransformationMapping mapping = new TransformationMapping();
```

```
    // configure mapping
    ...

    // add mapping to descriptor
    descriptor.addMapping(mapping);
}
```

For more information, see the following:

- [Section 27.13, "Transformation Mapping"](#)
- [Chapter 28, "Configuring a Relational Mapping"](#)
- [Part XXVIII, "Creation and Configuration of Mappings"](#)



# Part XIII

---

## Object-Relational Data Type Mappings

An object-relational data type mapping transforms certain object data member types to structured data source representations optimized for storage in specialized object-relational data type databases (such as Oracle Database). Object-relational data type mappings let you map an object model into an object-relational data type data model.

This part contains the following chapters:

- [Chapter 40, "Introduction to Object-Relational Data Type Mappings"](#)  
This chapter describes each of the different TopLink object-relational data type mapping types and important object-relational data type mapping concepts.
- [Chapter 41, "Configuring an Object-Relational Data Type Mapping"](#)  
This chapter explains how to configure TopLink object-relational data type mapping options common to two or more object-relational data type mapping types.
- [Chapter 42, "Configuring an Object-Relational Data Type Structure Mapping"](#)  
This chapter explains how to configure a structure object-relational data type mapping.
- [Chapter 43, "Configuring an Object-Relational Data Type Reference Mapping"](#)  
This chapter explains how to configure a reference object-relational data type mapping.
- [Chapter 44, "Configuring an Object-Relational Data Type Array Mapping"](#)  
This chapter explains how to configure an array object-relational data type mapping.
- [Chapter 45, "Configuring an Object-Relational Data Type Object Array Mapping"](#)  
This chapter explains how to configure an object array object-relational data type mapping.
- [Chapter 46, "Configuring an Object-Relational Data Type Nested Table Mapping"](#)  
This chapter explains how to configure a nested table object-relational data type mapping.



---

# Introduction to Object-Relational Data Type Mappings

An object-relational data type mapping transforms certain object data member types to structured data source representations optimized for storage in specialized object-relational data type databases (such as Oracle Database). Object-relational data type mappings let you map an object model into an object-relational data type data model.

Do not confuse object-relational data type mappings with relational mappings (see [Chapter 27, "Introduction to Relational Mappings"](#)). A relational mapping transforms any object data member type to a corresponding relational database (SQL) data source representation in any supported relational database. Relational mappings let you map an object model into a relational data model. In general, you can use relational mappings with any supported relational database. You can only use object-relational data type mappings with specialized object-relational data type databases optimized to support object-relational data type data source representations.

This chapter includes the following section:

- [Object-Relational Data Type Mapping Types](#)

For information on mapping concepts and features common to more than one type of TopLink mappings, see [Chapter 17, "Introduction to Mappings"](#).

## 40.1 Object-Relational Data Type Mapping Types

TopLink supports the object-relational data type mappings listed in [Table 40-1](#).

These mappings allow for an object model to persist in an object-relational data type data model. Currently, neither Oracle JDeveloper TopLink Editor nor TopLink Workbench support object-relational data type mappings—they must be defined in code or through amendment methods.

**Table 40–1 TopLink Object-Relationship Mapping Types**

Type of Mapping	Description	Oracle JDeveloper	TopLink Workbench	Java
Object-relational data type structure mapping (see <a href="#">Section 40.1.1</a> , "Object-Relational Data Type Structure Mapping")	Map to object-relational data type aggregate structures (the <code>Struct</code> type in JDBC and the <code>OBJECT</code> type in Oracle Database)			✓
Object-relational data type reference mapping (see <a href="#">Section 40.1.2</a> , "Object-Relational Data Type Reference Mapping")	Map to object-relational data type references (the <code>Ref</code> type in JDBC and the <code>REF</code> type in Oracle Database)			✓
Object-relational data type array mapping (see <a href="#">Section 40.1.3</a> , "Object-Relational Data Type Array Mapping")	Map a collection of primitive data to object-relational data type array data types (the <code>Array</code> type in JDBC and the <code>VARRAY</code> type in Oracle Database).			✓
Object-relational data type object array mapping (see <a href="#">Section 40.1.4</a> , "Object-Relational Data Type Object Array Mapping")	Map to object-relational data type array data types (the <code>Array</code> type in JDBC and the <code>VARRAY</code> type in Oracle Database).			✓
Object-relational data type nested table mapping (see <a href="#">Section 40.1.5</a> , "Object-Relational Data Type Nested Table Mapping")	Map to object-relational data type nested tables (the <code>Array</code> type in JDBC and the <code>NESTED TABLE</code> type in Oracle Database)			✓

### 40.1.1 Object-Relational Data Type Structure Mapping

In an object-relational data type data model, **structures** are user-defined data types or object types. This is similar to a Java class—it defines attributes or fields in which each attribute is one of the following:

- A primitive data type.
- Another structure.
- Reference to another structure.

TopLink maps nested structures with the `StructureMapping` class. The structure mapping supports null values and shared aggregates without requiring additional settings (because of the object-relational data type support of the database).

See [Chapter 42](#), "Configuring an Object-Relational Data Type Structure Mapping" for more information.

### 40.1.2 Object-Relational Data Type Reference Mapping

In an object-relational data type data model, structures reference each other through **refs**—not through foreign keys (as in a traditional data model). Refs are based on the target structure's `ObjectID`. They represent an object reference in Java.

TopLink maps refs with the `ReferenceMapping` class. The reference mapping does not require foreign key information (because of the object-relational data type support of the database).

See [Chapter 43](#), "Configuring an Object-Relational Data Type Reference Mapping" for more information.

### 40.1.3 Object-Relational Data Type Array Mapping

In an object-relational data type data model, structures can contain **arrays** (collections of other data types). These arrays can contain primitive data types or collections of other structures.

TopLink maps arrays of primitive data types with the `ArrayMapping` class. An array mapping maps to object-relational data type array data types (the `Array` type in JDBC and the `VARRAY` type in Oracle Database). To map a collection of aggregate structures, use an object array mapping (see [Section 40.1.4, "Object-Relational Data Type Object Array Mapping"](#)).

The object-relational data type database stores the arrays with their parent structure in the same table. To store information in a separate table from the parent structure's table, use a nested table mapping (see [Section 40.1.5, "Object-Relational Data Type Nested Table Mapping"](#)).

All elements in the array must be the same data type. The number of elements in an array controls the size of the array. An Oracle Database allows arrays of variable sizes (the `VARRAY` type).

See [Chapter 44, "Configuring an Object-Relational Data Type Array Mapping"](#) for more information.

### 40.1.4 Object-Relational Data Type Object Array Mapping

In an object-relational data type data model, structures can contain *arrays* (collections of other data types). These arrays can contain primitive data types or collections of other structures.

TopLink maps arrays of structures with the `ObjectArrayMapping` class. An object array mapping defines a collection-aggregated relationship, in which the target objects share the same row as the source object.

You must associate this mapping to an attribute in the parent class.

See [Chapter 45, "Configuring an Object-Relational Data Type Object Array Mapping"](#) for more information.

### 40.1.5 Object-Relational Data Type Nested Table Mapping

**Nested table** types model an unordered set of elements. These elements may be built-in or user-defined types. You can view a nested table as a single-column table or, if the nested table is an object type, as a multicolumn table (with a column for each attribute of the object type).

TopLink maps nested tables with the `NestedTableMapping` class. It represents a collection of object references in Java. Because of the object-relational data type support of the database, nested table mapping does not require foreign key information (as with a one-to-many mapping) or a relational table (as with a many-to-many mapping).

Typically, nested tables represent a one-to-many or many-to-many relationship of references to another independent structure. They support querying and joining better than the `VARRAY` types that are in-lined to the parent table. TopLink supports mapping a nested table of `REF` types only. TopLink does not support nested tables of basic or other structured data types—use array (see [Section 40.1.3, "Object-Relational Data Type Array Mapping"](#)) or object array (see [Section 40.1.4, "Object-Relational Data Type Object Array Mapping"](#)) mappings instead.

See [Chapter 46, "Configuring an Object-Relational Data Type Nested Table Mapping"](#) for more information.

---



---

## Configuring an Object-Relational Data Type Mapping

This chapter describes how to configure an object-relational data type mapping.

This chapter includes the following sections:

- [Introduction to Object-Relational Data Type Mapping Configuration](#)
- [Configuring Common Object-Relational Data Type Mapping Options](#)
- [Configuring Reference Class](#)
- [Configuring Attribute Name](#)
- [Configuring Field Name](#)
- [Configuring Structure Name](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 41.1 Introduction to Object-Relational Data Type Mapping Configuration

[Table 41–1](#) lists the types of object-relational data type mappings that you can configure and provides a cross-reference to the type-specific chapter that lists the configurable options supported by that type.

**Table 41–1** *Configuring Object-Relational Data Type Mappings*

<b>If you are creating...</b>	<b>See Also...</b>
Object-relational data type structure mapping (see <a href="#">Section 40.1.1</a> , "Object-Relational Data Type Structure Mapping")	<a href="#">Chapter 42, "Configuring an Object-Relational Data Type Structure Mapping"</a>
Object-relational data type reference mapping (see <a href="#">Section 40.1.2</a> , "Object-Relational Data Type Reference Mapping")	<a href="#">Chapter 43, "Configuring an Object-Relational Data Type Reference Mapping"</a>
Object-relational data type array mapping (see <a href="#">Section 40.1.3</a> , "Object-Relational Data Type Array Mapping")	<a href="#">Chapter 44, "Configuring an Object-Relational Data Type Array Mapping"</a>

**Table 41–1 (Cont.) Configuring Object-Relational Data Type Mappings**

If you are creating...	See Also...
Object-relational data type object array mapping (see Section 40.1.4, "Object-Relational Data Type Object Array Mapping")	Chapter 45, "Configuring an Object-Relational Data Type Object Array Mapping"
Object-relational data type nested table mapping (see Section 40.1.5, "Object-Relational Data Type Nested Table Mapping")	Chapter 46, "Configuring an Object-Relational Data Type Nested Table Mapping"

For more information, see the following:

- Chapter 17, "Introduction to Mappings"
- Chapter 27, "Introduction to Relational Mappings"

## 41.2 Configuring Common Object-Relational Data Type Mapping Options

Table 41–2 lists the configurable options shared by two or more object-relational data type mapping types. In addition to the configurable options described here, you must also configure the options described for the specific object-relational data type mapping types (see Section 40.1, "Object-Relational Data Type Mapping Types"), as shown in Table 41–1.

**Table 41–2 Common Options for Object-Relational Data Type Mappings**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Reference class (see Section 41.3, "Configuring Reference Class")			✓
Attribute name (see Section 41.4, "Configuring Attribute Name")			✓
Field name (see Section 41.5, "Configuring Field Name")			✓
Structure name (see Section 41.6, "Configuring Structure Name")			✓
Read-only (see Section 121.2, "Configuring Read-Only Mappings")			✓
Method or direct field access (see Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level")			✓
Indirection (lazy loading) (see Section 121.3, "Configuring Indirection (Lazy Loading)")			✓
Container policy (see Section 121.14, "Configuring Container Policy")			✓

## 41.3 Configuring Reference Class

When mapping an attribute that involves a relationship to another class, you must specify the reference class—the Java class to which the mapped attribute refers.

Table 41–3 summarizes which object-relational data type mappings support this option.



**Table 41–3 Mapping Support for Reference Class**

Mapping	How to Use Oracle JDeveloper	How to Use TopLink Workbench	How to Configure Reference Class Using Java
Object-relational data type structure mapping (see <a href="#">Section 40.1.1</a> , "Object-Relational Data Type Structure Mapping")			✓
Object-relational data type reference mapping (see <a href="#">Section 40.1.2</a> , "Object-Relational Data Type Reference Mapping")			✓
Object-relational data type array mapping (see <a href="#">Section 40.1.3</a> , "Object-Relational Data Type Array Mapping")			
Object-relational data type object array mapping (see <a href="#">Section 40.1.4</a> , "Object-Relational Data Type Object Array Mapping")			✓
Object-relational data type nested table mapping (see <a href="#">Section 40.1.5</a> , "Object-Relational Data Type Nested Table Mapping")			✓

### 41.3.1 How to Configure Reference Class Using Java

Use `oracle.toplink.mappings.ForeignReferenceMapping` method `setReferenceClass` to specify the target class of the attribute being mapped.

[Example 41–1](#) shows how to use this method with a `ReferenceMapping` that maps the `manager` attribute of the `Employee` class.

#### **Example 41–1 Configuring Reference Class in Java**

```
public void customize(ClassDescriptor descriptor) {
    ReferenceMapping managerMapping = new ReferenceMapping();

    managerMapping.setReferenceClass("Employee.class"); // set reference class
    managerMapping.setAttributeName("manager");

    // add this mapping to descriptor
    descriptor.addMapping(managerMapping);
}
```

For more information, see the *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.

## 41.4 Configuring Attribute Name

All object-relational data type mappings map an attribute in a Java object to field in the database. The attribute name is the name of the attribute being mapped. The name is as specified in the reference class (see [Section 41.3](#), "Configuring Reference Class").

[Table 41–4](#) summarizes which object-relational data type mappings support this option.

**Table 41–4 Mapping Support for Attribute Name**

Mapping	How to Use Oracle JDeveloper	How to Use TopLink Workbench	How to Configure Attribute Name Using Java
Object-relational data type structure mapping (see <a href="#">Section 40.1.1</a> , "Object-Relational Data Type Structure Mapping")			✓
Object-relational data type reference mapping (see <a href="#">Section 40.1.2</a> , "Object-Relational Data Type Reference Mapping")			✓
Object-relational data type array mapping (see <a href="#">Section 40.1.3</a> , "Object-Relational Data Type Array Mapping")			✓
Object-relational data type object array mapping (see <a href="#">Section 40.1.4</a> , "Object-Relational Data Type Object Array Mapping")			✓
Object-relational data type nested table mapping (see <a href="#">Section 40.1.5</a> , "Object-Relational Data Type Nested Table Mapping")			✓

### 41.4.1 How to Configure Attribute Name Using Java

Use `oracle.toplink.mappings.DatabaseMapping` method `setAttributeName` to specify the name of the attribute being mapped.

[Example 41–2](#) shows how to use this method with a `ReferenceMapping` that maps the `manager` attribute of the `Employee` class.

#### **Example 41–2 Configuring Attribute Name in Java**

```
public void customize(ClassDescriptor descriptor) {
    ReferenceMapping managerMapping = new new ReferenceMapping();
    managerMapping.setReferenceClass("Employee.class");
    managerMapping.setAttributeName("manager"); // set attribute name

    // add this mapping to descriptor
    descriptor.addMapping (managerMapping);
}
```

For more information, see the *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.

## 41.5 Configuring Field Name

All object-relational data type mappings require the name of database field to which their specified attribute is mapped. This field name can be the column name of a database table or the name of a field in an object type created on the database.

[Table 41–5](#) summarizes which object-relational data type mappings support this option.

**Table 41–5 Mapping Support for Field Name**

Mapping	How to Use Oracle JDeveloper	How to Use TopLink Workbench	How to Configure Field Name Using Java
Object-relational data type structure mapping (see Section 40.1.1, "Object-Relational Data Type Structure Mapping")			✓
Object-relational data type reference mapping (see Section 40.1.2, "Object-Relational Data Type Reference Mapping")			✓
Object-relational data type array mapping (see Section 40.1.3, "Object-Relational Data Type Array Mapping")			✓
Object-relational data type object array mapping (see Section 40.1.4, "Object-Relational Data Type Object Array Mapping")			✓
Object-relational data type nested table mapping (see Section 40.1.5, "Object-Relational Data Type Nested Table Mapping")			✓

### 41.5.1 How to Configure Field Name Using Java

Use the object-relational data type mapping method `setFieldName` to specify the database field to which the attribute is mapped.

[Example 41–3](#) shows how to use this method with an `ObjectArrayMapping` that maps the `Employee` class attribute `phone` to database field name `PHONE_NUMBER`.

#### **Example 41–3 Configuring Field Name in Java**

```
public void customize(ClassDescriptor descriptor) {
    ObjectArrayMapping phonesMapping = new ObjectArrayMapping();
    phonesMapping.setReferenceClass("Employee.class");
    phonesMapping.setAttributeName("phone");
    phonesMapping.setFieldName("PHONE_NUMBER"); // set field name

    // add this mapping to descriptor
    descriptor.addMapping(phonesMapping);
}
```

For more information, see the *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.

## 41.6 Configuring Structure Name

Certain object-relational data type mappings require the specification of the data type or structure name of the field being mapped. The structure name is the name of the array or table type that defines the field.

[Table 41–6](#) summarizes which object-relational data type mappings support this option.

**Table 41–6 Mapping Support for Structure Name**

<b>Mapping</b>	<b>How to Use Oracle JDeveloper</b>	<b>How to Use TopLink Workbench</b>	<b>How to Configure Structure Name Using Java</b>
Object-relational data type structure mapping (see <a href="#">Section 40.1.1</a> , "Object-Relational Data Type Structure Mapping")			
Object-relational data type reference mapping (see <a href="#">Section 40.1.2</a> , "Object-Relational Data Type Reference Mapping")			
Object-relational data type array mapping (see <a href="#">Section 40.1.3</a> , "Object-Relational Data Type Array Mapping")			✓
Object-relational data type object array mapping (see <a href="#">Section 40.1.4</a> , "Object-Relational Data Type Object Array Mapping")			✓
Object-relational data type nested table mapping (see <a href="#">Section 40.1.5</a> , "Object-Relational Data Type Nested Table Mapping")			✓

### 41.6.1 How to Configure Structure Name Using Java

Use the object-relational data type mapping method `setStructureName` to specify the structure of the attribute being mapped.

[Example 41–4](#) shows how to use this method with an `ObjectArrayMapping` that maps the `Employee` class attribute `phones` to database field name `PHONE_NUMBERS` of type `PHONE_ARRAY_TYPE`.

#### **Example 41–4 Configuring Structure Name in Java**

```
public void customize(ClassDescriptor descriptor) {
    ObjectArrayMapping phonesMapping = new ObjectArrayMapping();
    phonesMapping.setReferenceClass("Employee.class");
    phonesMapping.setAttributeName("phones");
    phonesMapping.setFieldName("PHONE_NUMBERS");
    phonesMapping.setStructureName("PHONE_ARRAY_TYPE"); // set structure name

    // add this mapping to descriptor
    descriptor.addMapping(phonesMapping);
}
```

For more information, see the *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.

## Configuring an Object-Relational Data Type Structure Mapping

This chapter describes the various components that you must configure in order to use an object-relational data type structure mapping.

This chapter includes the following section:

- [Introduction to Object-Relational Data Type Structure Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 42.1 Introduction to Object-Relational Data Type Structure Mapping Configuration

[Table 42–1](#) lists the configurable options for an object-relational data type structure mapping.

**Table 42–1 Configurable Options for Object-Relational Data Type Structure Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Reference class (see <a href="#">Section 41.3, "Configuring Reference Class"</a> )			✓
Attribute name (see <a href="#">Section 41.4, "Configuring Attribute Name"</a> )			✓
Field name (see <a href="#">Section 41.5, "Configuring Field Name"</a> )			✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )			✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )			✓

For more information, see the following:

- [Section 40.1.1, "Object-Relational Data Type Structure Mapping"](#)
- [Chapter 41, "Configuring an Object-Relational Data Type Mapping"](#)



## Configuring an Object-Relational Data Type Reference Mapping

This chapter describes the various components that you must configure in order to use an object-relational data type reference mapping.

This chapter includes the following section:

- [Introduction to Object-Relational Data Type Reference Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 43.1 Introduction to Object-Relational Data Type Reference Mapping Configuration

[Table 43–1](#) lists the configurable options for an object-relational data type reference mapping.

**Table 43–1** Configurable Options for Object-Relational Data Type Reference Mapping

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Reference class (see <a href="#">Section 41.3, "Configuring Reference Class"</a> )			✓
Attribute name (see <a href="#">Section 41.4, "Configuring Attribute Name"</a> )			✓
Field name (see <a href="#">Section 41.5, "Configuring Field Name"</a> )			✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )			✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )			✓
Private or independent relationships (see <a href="#">Section 121.7, "Configuring Private or Independent Relationships"</a> )			✓
Indirection (lazy loading) (see <a href="#">Section 121.3, "Configuring Indirection (Lazy Loading)"</a> )			✓

For more information, see the following:

- [Section 40.1.2, "Object-Relational Data Type Reference Mapping"](#)
- [Chapter 41, "Configuring an Object-Relational Data Type Mapping"](#)





## Configuring an Object-Relational Data Type Array Mapping

This chapter describes the various components that you must configure in order to use an object-relational data type array mapping.

---

**Note:** To map a collection of aggregate structures, use an object-relational data type object array mapping (see [Section 40.1.4, "Object-Relational Data Type Object Array Mapping"](#)). To store information in a separate table from the parent structure's table, use an object-relational data type nested table mapping (see [Section 40.1.5, "Object-Relational Data Type Nested Table Mapping"](#)).

---

This chapter includes the following section:

- [Introduction to Object-Relational Data Type Array Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 44.1 Introduction to Object-Relational Data Type Array Mapping Configuration

[Table 44-1](#) lists the configurable options for an object-relational data type array mapping.

**Table 44-1** Configurable Options for Object-Relational Data Type Array Mapping

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Attribute name (see <a href="#">Section 41.4, "Configuring Attribute Name"</a> )			✓
Field name (see <a href="#">Section 41.5, "Configuring Field Name"</a> )			✓
Structure name (see <a href="#">Section 41.6, "Configuring Structure Name"</a> )			✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )			✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )			✓

**Table 44–1 (Cont.) Configurable Options for Object-Relational Data Type Array Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Serialized object converter (see <a href="#">Section 121.9</a> , "Configuring a Serialized Object Converter")			✓
Type conversion converter (see <a href="#">Section 121.10</a> , "Configuring a Type Conversion Converter")			✓
Object type converter (see <a href="#">Section 121.11</a> , "Configuring an Object Type Converter")			✓
Container policy (see <a href="#">Section 121.14</a> , "Configuring Container Policy")			✓

For more information, see the following:

- [Section 40.1.3, "Object-Relational Data Type Array Mapping"](#)
- [Chapter 41, "Configuring an Object-Relational Data Type Mapping"](#)

## Configuring an Object-Relational Data Type Object Array Mapping

This chapter describes the various components that you must configure in order to use an object-relational data type object array mapping.

This chapter includes the following section:

- [Introduction to Object-Relational Data Type Object Array Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 45.1 Introduction to Object-Relational Data Type Object Array Mapping Configuration

[Table 45–1](#) lists the configurable options for an object-relational data type object array mapping.

**Table 45–1** Configurable Options for Object-Relational Data Type Object Array Mapping

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Reference class (see <a href="#">Section 41.3, "Configuring Reference Class"</a> )			✓
Attribute name (see <a href="#">Section 41.4, "Configuring Attribute Name"</a> )			✓
Field name (see <a href="#">Section 41.5, "Configuring Field Name"</a> )			✓
Structure name (see <a href="#">Section 41.6, "Configuring Structure Name"</a> )			✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )			✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )			✓
Container policy (see <a href="#">Section 121.14, "Configuring Container Policy"</a> )			✓

For more information, see the following:

- [Section 40.1.4, "Object-Relational Data Type Object Array Mapping"](#)
- [Chapter 41, "Configuring an Object-Relational Data Type Mapping"](#)



## Configuring an Object-Relational Data Type Nested Table Mapping

This chapter describes the various components that you must configure in order to use an object-relational data type nested table mapping.

---

**Note:** For an equivalent mapping for basic or other structured data types, use object-relational data type array (see [Section 40.1.3, "Object-Relational Data Type Array Mapping"](#)) or object array (see [Section 40.1.4, "Object-Relational Data Type Object Array Mapping"](#)) mappings.

---

This chapter includes the following section:

- [Introduction to Object-Relational Data Type Nested Table Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 46.1 Introduction to Object-Relational Data Type Nested Table Mapping Configuration

[Table 46-1](#) lists the configurable options for an object-relational data type nested table mapping.

**Table 46-1** Configurable Options for Object-Relational Data Type Nested Table Mapping

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Reference class (see <a href="#">Section 41.3, "Configuring Reference Class"</a> )			✓
Attribute name (see <a href="#">Section 41.4, "Configuring Attribute Name"</a> )			✓
Field name (see <a href="#">Section 41.5, "Configuring Field Name"</a> )			✓
Structure name (see <a href="#">Section 41.6, "Configuring Structure Name"</a> )			✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )			✓

**Table 46–1 (Cont.) Configurable Options for Object-Relational Data Type Nested Table**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Method or direct field access (see <a href="#">Section 121.6</a> , "Configuring Method or Direct Field Accessing at the Mapping Level")			✓
Indirection (lazy loading) (see <a href="#">Section 121.3</a> , "Configuring Indirection (Lazy Loading)")			✓
Container policy (see <a href="#">Section 121.14</a> , "Configuring Container Policy")			✓

For more information, see the following:

- [Section 40.1.5, "Object-Relational Data Type Nested Table Mapping"](#)
- [Chapter 41, "Configuring an Object-Relational Data Type Mapping"](#)

# Part XIV

---

## XML Projects

This part describes XML projects and contains the following chapters:

- [Chapter 47, "Introduction to XML Projects"](#)  
This chapter introduces XML project concepts.
- [Chapter 48, "Creating an XML Project"](#)  
This chapter explains how to create XML projects.
- [Chapter 49, "Configuring an XML Project"](#)  
This chapter explains how to configure XML projects.





---



---

## Introduction to XML Projects

This chapter provides an overview of XML projects and their components.

This chapter includes the following section:

- [XML Project Concepts](#)

For information on project concepts and features common to more than one type of TopLink projects, see [Chapter 15, "Introduction to Projects"](#).

### 47.1 XML Project Concepts

Use an XML project for nontransactional, nonpersistent (in-memory) conversions between Java objects and XML documents using JAXB (see [Section 47.1.1, "TopLink Support for Java Architecture for XML Binding \(JAXB\)"](#) and [Section 47.1.2, "JAXB Validation"](#)). Both Oracle JDeveloper TopLink Editor and TopLink Workbench provide complete support for creating XML projects.

The TopLink runtime performs XML data conversion based on one or more XML schemas. In an XML project, TopLink Workbench directly references schemas in the deployment XML, and exports mappings configured with respect to the schemas you specify. For information on how to use TopLink Workbench with XML schemas, see [Section 5.6, "Using XML Schemas"](#). For information on how TopLink supports XML namespaces, see [Section 15.4, "XML Namespaces Overview"](#).

[Table 47-1](#) describes the components of an XML project.

**Table 47-1** XML Project Components

Component	Supported Types
Data Source	None
Descriptors	For more information, see <a href="#">Section 50.1, "XML Descriptor Concepts"</a> .
Mappings	For more information, see the following: <ul style="list-style-type: none"> <li>■ <a href="#">Part VIII, "Mappings"</a></li> <li>■ <a href="#">Part XVI, "XML Mappings"</a></li> </ul>

---



---

**Note:** In an XML project, you do not use TopLink queries and expressions.

---



---

## 47.1.1 TopLink Support for Java Architecture for XML Binding (JAXB)

JAXB defines annotations to control the mapping of Java objects to XML, but it also defines a default set of mappings. Using the defaults, TopLink can marshal a set of objects into XML, and unmarshal an XML document into objects. JAXB provides a standard Java object-to-XML API. For more information, see <http://java.sun.com/xml/jaxb/index.html>.

TopLink provides an extra layer of functions on top of JAXB. It allows for the creation and subsequent manipulation of mappings (in the form of a TopLink runtime project or TopLink Workbench project) from an existing object model, without requiring the recompilation of the JAXB object model.

An essential component of this function is the TopLink JAXB compiler. Using the TopLink JAXB compiler, you can generate both a TopLink XML project and JAXB-compliant object model classes from your XML schema.

The TopLink JAXB compiler simplifies JAXB application development with TopLink by automatically generating (see [Section 48.2, "Creating an XML Project from an XML Schema"](#)) both the required JAXB files (see [Section 47.1.1.2, "Working with JAXB-Specific Generated Files"](#)) and the TopLink files from your XML schema (XSD) document.

For more information on using the JAXB and TopLink-specific run-time classes, see [Section 47.1.1.3, "Using TopLink JAXB Compiler-Generated Files at Run Time"](#).

For information on marshalling POJOs with JAXB, see <http://www.oracle.com/technology/products/ias/toplink/preview/ho-w-to/JAXBwithPOJOs.html>.

### 47.1.1.1 Generating TopLink Project and XML Schema Using JAXB Annotations

The TopLink JAXB compiler generates a TopLink project and an XML schema using JAXB annotations that [Table 47–2](#) lists.

**Table 47–2 JAXB Annotations Supported by TopLink**

Annotation	Description	Level
<code>XmlRootElement</code>	Indicates that a class should be mapped to a root-level element in a schema. The element name and namespace are specified in this annotation.	Class
<code>XmlElement</code>	Indicates that a particular attribute on a Java class should be mapped to an XML element in the schema. The name and namespace can be specified by this annotation.	Field
<code>XmlAttribute</code>	Indicates that the Java attribute should map to an XML attribute in the schema. Name and namespace should be provided.	Field
<code>XmlElementWrapper</code>	Specifies a wrapper element around another element or attribute. You can use this annotation to create a grouping element around a collection.	Field
<code>XmlList</code>	Indicates that a collection property should map to a space-separated list in XML.	Field

**Table 47-2 (Cont.) JAXB Annotations Supported by TopLink**

Annotation	Description	Level
<code>XmlType</code>	Defines the complex-type for a class. Using this annotation's <code>propOrder</code> property, you can specify the order in which to map elements. The <code>propOrder</code> property also determines if the schema should contain a <code>sequence</code> or an <code>all</code> . Depending on the structure of the class, it will either map to a <code>ComplexType</code> , a <code>SimpleType</code> , or a <code>ComplexType</code> with <code>SimpleContent</code> .	Class
<code>XmlTransient</code>	Indicates that a mapping should not be generated for a particular field. This is a marker annotation.	Field
<code>XmlSchema</code>	Specifies the target namespace for a schema. You can also use this annotation to configure namespace prefix mappings with the <code>XmlNs</code> annotation.	Package
<code>XmlNs</code>	Appears only in an <code>XmlSchema</code> annotation to specify namespace-prefix mappings.	Package
<code>XmlValue</code>	Maps an attribute to a text node under the parent class (for example, an Xpath of "text()"). Also indicates that the owning class should map to either a <code>SimpleType</code> or a <code>ComplexType</code> with <code>SimpleContent</code> .	Field
<code>XmlEnum</code>	Indicates that a JDK 1.5 Enum type should map to a simple type with enumeration facets in the schema. The base schema type is specified on this annotation.	Field
<code>XmlEnumValue</code>	Lets you specify the enumeration facets to be used in the schema, if they are to be different from the string values of the Enum constants specified in Java.	Field
<code>XmlAccessorType</code>	Specifies how the classes' attributes should be processed. The following are valid values: <ul style="list-style-type: none"> <li>■ <code>FIELD</code>—Process all the public and/or private fields of the class.</li> <li>■ <code>PROPERTY</code>—Process all the public or private <code>get</code> and <code>set</code> method pairs on the class.</li> <li>■ <code>PUBLIC_MEMBER</code>—Process all the public fields and public <code>get</code> and <code>set</code> method pairs on the class.</li> <li>■ <code>NONE</code>—Only process members that are annotated with JAXB annotations.</li> </ul>	Package or class

**Table 47-2 (Cont.) JAXB Annotations Supported by TopLink**

Annotation	Description	Level
<code>XmlAccessorType</code>	Specifies the order in which properties are to be processed. The following are valid values: <ul style="list-style-type: none"> <li>■ DEFAULT</li> <li>■ ALPHABETICAL</li> </ul>	Package or class
<code>XmlSchemaType</code>	If specified at a property level, indicates the schema type that should be used in schema generation. If used as part of an <code>XmlSchemaTypes</code> annotation, overrides default schema types at a package level.	Property or package
<code>XmlSchemaTypes</code>	Contains a collection of <code>XmlSchemaType</code> annotations. Each one specifies a Java class and a XML schema type pair that should be used as a default for this package.	Package
<code>XmlAnyAttribute</code>	Specifies that a Map property should be mapped to an <code>xs:any</code> attribute in the schema. For more information, see <code>XMLAnyAttributeMapping</code> (see <a href="#">Section 53.18, "XML Any Attribute Mapping"</a> )	Field

For more information, see Chapter 8 of JAXB 2.0 Specification at <http://jcp.org/aboutJava/communityprocess/pfd/jsr222/index.html>

---



---

**Note:** The TopLink project is generated from a collection of annotated Java classes with support for relationships, collection-style mappings, and JDK 1.5 enumerations.

The schema is generated from a set of annotated Java classes with support for relationships.

---



---

### 47.1.1.2 Working with JAXB-Specific Generated Files

The TopLink JAXB compiler generates the following JAXB-specific files from your XSD:

- [Implementation Classes](#)

The JAXB runtime uses these files as specified by the JAXB specification.

All JAXB-specific files are generated in the output directory you define, and in the subdirectories implied by the target package name you define. For more information about TopLink JAXB binding compiler options, see [Section 48.2, "Creating an XML Project from an XML Schema"](#).

Before you compile your generated classes, be sure to configure your IDE classpath to include `<ORACLE_HOME>\lib\xml.jar`. For an example, see [Chapter 7, "Using an Integrated Development Environment"](#).

**47.1.1.2.1 Implementation Classes** All implementation classes are named according to the content, element, or implementation name attribute defined in the XSD.

The generated implementation classes are simple domain classes, with private attributes for each JAXB property, and public `get` and `set` methods that return or set attribute values.

### 47.1.1.3 Using TopLink JAXB Compiler-Generated Files at Run Time

At run time, you can access the TopLink JAXB compiler-generated files by doing the following:

- Using TopLink XMLContext (see [Section 47.1.1.3.1, "How to Use TopLink XMLContext"](#))
- Using TopLink XMLBinder (see [Section 47.1.1.3.3, "How to Use TopLink XMLBinder"](#))
- Using TopLink JAXBContext (see [Section 47.1.1.3.4, "How to Use JAXBContext"](#))

**47.1.1.3.1 How to Use TopLink XMLContext** TopLink provides an `oracle.toplink.ox.XMLContext` class using which you can create instances of TopLink XMLMarshaller, XMLUnmarshaller, XMLBinder (see [Section 47.1.1.3.3, "How to Use TopLink XMLBinder"](#)), and XMLValidator.

The XMLContext is thread-safe. For example, if multiple threads accessing the same XMLContext object request an XMLMarshaller, each will receive their own instance of XMLMarshaller, so any state that the XMLMarshaller maintains will be unique to that process. By using the XMLContext, you can use TopLink XML in multithreaded architectures, such as the binding layer for Web services.

Create the XMLContext using its constructor method and by passing in the session name defined in the `sessions.xml` file, as the following example shows:

```
XMLContext context = new XMLContext("mysession");
```

You can also create the XMLContext from multiple sessions using a colon separated list of session names, as the following example shows:

```
XMLContext context = new XMLContext("session1:session2:session3");
```

Use the XMLContext to create a TopLink XMLMarshaller, XMLUnmarshaller, XMLBinder, and XMLValidator, as follows:

```
XMLMarshaller marshaller = context.createMarshaller();
marshaller.marshal(myObject, outputStream);
marshaller.setFormattedOutput(true);
```

```
XMLUnmarshaller unmarshaller = context.createUnmarshaller();
Employee emp = (Employee)unmarshaller.unmarshal(new File("employee.xml"));
```

```
XMLBinder binder = context.createBinder();
Address add = (Address)binder.unmarshal(myElement);
```

```
XMLValidator validator = context.createValidator();
boolean isValid = validator.validate(emp);
```

Using the XMLContext `getDocumentPreservationPolicy` method, you can retrieve this context's document preservation policy in a form of the `DocumentPreservationPolicy` object. This object's API lets you specify the position of newly added to the node elements, as well as disable the addition of new elements.

**47.1.1.3.2 How to Use Marshal and Unmarshal Events** You can provide TopLink XMLMarshaller and XMLUnmarshaller with additional functionality at run time by registering them with a listener to handle specific event callbacks. This allows for

extra processing on a business object either immediately before, or immediately after an object is written to or read from XML.

There are two types of event callbacks that you can handle in two different ways:

1. To handle listener-based callbacks, set an event handler on an instance of `XMLMarshaller` or `XMLUnmarshaller` that implements a required interface, such as `XMLMarshalListener` or `XMLUnmarshalListener`. The events are triggered on the marshaller or unmarshaller's listener for any classes being marshalled or unmarshalled.
2. To handle class-specific callbacks, you need to provide the required callback methods on your business objects.

---

---

**Note:** If you specify both the listener and the business object callbacks, the class-specific method will be invoked before the listener event.

---

---

[Example 47-1](#) shows how to create your custom event listeners.

**Example 47-1 Implementing the `TopLink XMLMarshalListener` and `XMLUnmarshalListener` Interfaces**

```
public class EmployeeMarshalListener implements XMLMarshalListener {

    public void beforeMarshal(Object target) {
        // do something
    }

    public void afterMarshal(Object target) {
        // do something
    }
}

public class EmployeeUnmarshalListener implements XMLUnmarshalListener {

    public void beforeUnmarshal(Object target, Object parent) {
        // do something
    }

    public void afterUnmarshal(Object target, Object parent) {
        // do something
    }
}
```

[Example 47-2](#) and [Example 47-3](#) show how to use the listeners in your application.

**Example 47-2 Using the Marshal Listener**

```
...
XMLMarshaller marshaller = context.createMarshaller();
marshaller.setMarshalListener(new EmployeeMarshalListener());
marshaller.marshal(myObject, System.out);
...
```

**Example 47-3 Using the Unmarshal Listener**

```
...
XMLUnmarshaller unmarshaller = context.createUnmarshaller();
```

```

unmarshaller.setUnmarshalListener(new EmployeeUnmarshalListener());
Object myObject = unmarshaller.unmarshal(myFile);
...

```

**47.1.1.3.3 How to Use TopLink XMLBinder** XMLBinder is a run-time class that allows you to preserve a document that you have unmarshalled, as well as to resynchronize that document with the unmarshalled objects at any time.

---

**Note:** This functionality is based on the JAXB binder API (`javax.xml.bind.Binder<XmlNode>`). This is an addition to the design-time method of document preservation.

---

When the XMLBinder unmarshalls XML nodes into mapping objects, and then performs an update operation, it preserves not only the order of elements, but also the comments from an original XML document using the cached value. This way, both the returned node and the cached node are identical and reflect the preserved document. When adding new elements, TopLink XMLBinder places them at the correct location (relative to other mapped content) in the node.

When unmarshalling a document that contains only unmapped content, setting some values and then marshalling, the XMLBinder adds new elements before existing unmapped data, such as comments and processing instructions.

[Example 47-4](#) demonstrates how you can unmarshal a document using an instance of an XMLBinder.

**Example 47-4 Unmarshalling a Document Using XMLBinder**

```

XMLContext conext = new XMLContext(myProject);
XMLBinder binder = context.createBinder();
Employee emp = (Employee) binder.unmarshal(myDocument);

```

In the preceding example, `emp` is the root object that was unmarshalled from the provided document. The binder maintains references to the original XML document as well as objects generated during the unmarshal operation.

[Example 47-5](#) demonstrates how you can make changes to the object (`Employee`) and update the XML document using an instance of an XMLBinder.

**Example 47-5 Making Changes to an Object and to Updating XML Using XMLBinder**

```

...
emp.setPhoneNumber("123-4567");
binder.updateXML(emp);

```

In the preceding example, the `updateXML` method will update the cached node in the binder. Note that the cached node preserves the document, including comments, as the following example shows:

```

<employee>
  <!--comment1 -->
  <name>John Smith </name>
  <phone-number>123-4567</phone-number>
  <!--comment2 -->
</employee>

```

[Example 47-6](#) demonstrates how you can obtain an associated node for a subobject (`Address`) of the `Employee` using an instance of an XMLBinder.

**Example 47-6 Obtaining an Associated Node Using XMLBinder**

```

...

```

```
Address addr = emp.getAddress();
Node addressNode = binder.getXMLNode(addr);
```

In the preceding example, the returned node (`addressNode`) is the XML node in the original XML document that was used to build this employee's `Address` object.

[Example 47-7](#) demonstrates how you can make changes to an XML node and update objects (`Address`) of the `Employee` using an instance of an `XMLBinder`.

**Example 47-7 Making Changes to an XML Node and Updating Objects Using XMLBinder**

```
...
addressNode.setAttribute("apt-no", "1527");
Address updatedAddressNode = binder.updateObject(addressNode);
```

In the preceding example, the address returned from the binder operation is the original `Address` object created during the unmarshal operation, but now it contains the updated apartment number information from the XML document.

**47.1.1.3.4 How to Use JAXBContext** You can create an instance of `JAXBContext` from a collection of classes that are to be bound to XML. This will generate a `TopLink` project from the classes dynamically at run time.

Using the instance of `JAXBContext` you can obtain `Marshaller` and `Unmarshaller` instances to operate on those classes, as [Example 47-8](#) demonstrates. Note that this example assumes that you configure your application classpath to include your domain object class files.

**Example 47-8 Creating and Using JAXBContext**

```
Class[] classes = {Employee.class, Address.class, Department.class};
JAXBContext jaxbContext = JAXBContext.newInstance(classes);
Marshaller marshaller = jaxbContext.createMarshaller();
marshaller.marshal(myEmployee, myOutput);
```

---

---

**Note:** The `JAXBContext` object is thread-safe.

---

---

**47.1.1.3.5 How to Use JAXBElement** `TopLink` lets you marshal to and unmarshal from `JAXBElement` types. The `javax.xml.bind.JAXBElement` class provides access to the following basic properties of an XML element:

- its qualified name, which is composed of *{target namespace}* and *{name}*
- its value, which is an instance of the Java class binding of its *{type definition}*
- whether or not the element's content is *{nillable}*

`TopLink` supports the following JAXB element marshal API defined in the `Marshaller`:

- `marshal(java.lang.Object jaxbElement, java.io.Writer writer)`
- `marshal(java.lang.Object jaxbElement, java.io.OutputStream os)`
- `marshal(java.lang.Object jaxbElement, org.xml.sax.ContentHandler)`
- `marshal(java.lang.Object jaxbElement, javax.xml.transform.Result)`



- `marshal(java.lang.Object jaxbElement, org.w3c.dom.Node)`
- `marshal(java.lang.Object jaxbElement, javax.xml.stream.XMLStreamWriter writer)`

---

**Note:** If the first parameter is not a `JAXBElement`, the `marshal` operation will throw an `oracle.toplink.exceptions.XMLMarshalException.MARSHALL_EXCEPTION`.

---

TopLink provides implementation of the following JAXB element unmarshal API defined in the `Unmarshaller`:

- `<T> JAXBElement<T> unmarshal(org.w3c.dom.Node node, Class<T> declaredType)`
- `<T> JAXBElement<T> unmarshal(javax.xml.transform.Source source, Class<T> declaredType)`
- `<T> JAXBElement<T> unmarshal(javax.xml.stream.XMLStreamReader streamReader, Class<T> declaredType)`
- `<T> JAXBElement<T> unmarshal(javax.xml.stream.XMLEventReader eventReader, Class<T> declaredType)`

### 47.1.2 JAXB Validation

TopLink can validate both complete object trees and subtrees against the XML schema that was used to generate the implementation classes. In addition, TopLink will validate both root objects (objects that correspond to the root element of the XML document) and nonroot objects against the schema used to generate the object's implementation class.

When validating an object tree, TopLink performs the following checks (in order):

1. Check that element appears in the document at the specified location.
2. If **maxOccurs** or **minOccurs** is specified, check number of elements.
3. If **type** is specified, check that element value satisfies the type constraints.
4. If a **fixed value** is specified, check that the element value matches it.
5. If **restrictions** (length, patterns, enumerations, and so on) are specified, check that the element value satisfies it.
6. If an **ID** type is specified during a `validateRoot` operation, check that the ID value is unique in the document.
7. If an **IDREF** type is specified during a `validateRoot` operation, check that the ID referenced exists in the document.

If validation errors are encountered, TopLink stops validating the object tree and creates a `Validation` object, according to the JAXB specification. If an error occurs in a subobject, TopLink will not validate further down that object's subtree.

For more information on using TopLink XML to perform validation, see [Section 47.1.1.3, "Using TopLink JAXB Compiler-Generated Files at Run Time"](#).

For additional information on JAXB and validation, refer to the JAXB specification at <http://java.sun.com/xml/jaxb/>.

---

---

## Creating an XML Project

This chapter describes the various components that you must configure in order to create an XML project.

This chapter includes the following sections:

- [Introduction to XML Project Creation](#)
- [Creating an XML Project from an XML Schema](#)

For information on how to create more than one type of TopLink projects, see [Chapter 116, "Creating a Project"](#).

### 48.1 Introduction to XML Project Creation

You can create a project using Oracle JDeveloper, TopLink Workbench, or Java code.

Oracle recommends using either Oracle JDeveloper or TopLink Workbench to create projects and generate deployment XML or Java source versions of the project for use at run time. For more information on how to create a project using TopLink Workbench, see [Section 116.1.2, "How to Create a Project Using TopLink Workbench"](#). For information on how to create a project using Java, see [Section 116.1.3, "How to Create a Project Using Java"](#).

You can use TopLink to create an XML project, if you have an XML schema (XSD) document, but no object model yet (see [Section 48.2, "Creating an XML Project from an XML Schema"](#)). If you have both XSD and object model classes, you can create an XML project using the procedure described in [Section 116.1.2, "How to Create a Project Using TopLink Workbench"](#).

For more information, see [Chapter 47, "Introduction to XML Projects"](#).

### 48.2 Creating an XML Project from an XML Schema

If you have an existing data model (XML schema document), but you do not have a corresponding object model (Java classes for domain objects), use this procedure to create your TopLink project and automatically generate the corresponding object model.

---

---

**Note:** If you have both XSD and object model classes, you can create an XML project using the procedure described in [Section 116.1.2, "How to Create a Project Using TopLink Workbench"](#).

---

---

Using the TopLink JAXB compiler simplifies JAXB application development with TopLink by automatically generating both the required JAXB files and the TopLink files from your XML schema (XSD) document. Once generated, you can open the Oracle JDeveloper or TopLink Workbench project to fine-tune XML mappings without having to recompile your JAXB object model.

You can use the TopLink JAXB compiler from Oracle JDeveloper TopLink Editor, TopLink Workbench (see [Section 48.2.1, "How to Create an XML Project from an XML Schema Using TopLink Workbench"](#)), or from the command line (see [Section 48.2.2, "How to Create an XML Project from an XML Schema Using the Command Line"](#)).

---

**Note:** Before you compile your generated classes, be sure to configure your IDE classpath to include `<ORACLE_HOME>\lib\xml.jar`. For example, see [Chapter 7, "Using an Integrated Development Environment"](#).

---

For more information, see the following:

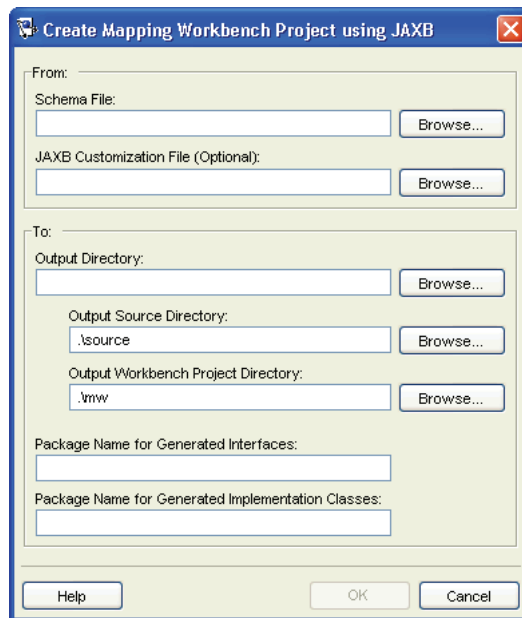
- [Section 47.1.1, "TopLink Support for Java Architecture for XML Binding \(JAXB\)"](#)
- [Section 47.1.1.3, "Using TopLink JAXB Compiler-Generated Files at Run Time"](#)

## 48.2.1 How to Create an XML Project from an XML Schema Using TopLink Workbench

To create a new, mapped TopLink Workbench project from an XML schema using JAXB, use this procedure:

1. From TopLink Workbench, select **File > New > Project > From XML Schema (JAXB)**.

**Figure 48–1 Create TopLink Workbench Project using JAXB Dialog Box**



Use the following information to enter data in each field of this dialog box:

Field	Description
<b>From</b>	Use these fields to specify your existing JAXB information.
<b>Schema File</b>	Click <b>Browse</b> and select the fully qualified path to your XSD file.
<b>JAXB Customization File</b>	This is an optional setting. It can be used if you have a standard JAXB configuration file that you wish to use to override the default JAXB compiler behavior. The JAXB customization file contains binding declarations for customizing the default binding between an XSD component and its Java representation.
<b>To</b>	Use these fields to specify the location and options of the TopLink Workbench project.
<b>Output Directory</b>	Click <b>Browse</b> and select the path to the directory into which generated files are written. All paths used in the project are relative to this directory.
<b>Output Source Directory</b>	Click <b>Browse</b> and select the path to the directory (relative to the <b>Output Directory</b> ) into which generated interfaces, implementation classes, and deployment files are written. Default: directory named <code>source</code> in the specified output directory.
<b>Output Workbench Project Directory</b>	Click <b>Browse</b> and select the path to the directory (relative to the <b>Output Directory</b> ) into which the TopLink Workbench project files are written. Default: directory named <code>mw</code> in the specified output directory.
<b>Package Name for Generated Interfaces</b>	The optional name of the package to which generated interfaces belong. This defines your context path. If it is not specified, a package name of <code>jaxbderived.&lt;schema name&gt;</code> is used where <code>&lt;schema name&gt;</code> is the name of the schema specified by the <b>Schema File</b> field.
<b>Package Name for Generated Implementation Classes</b>	The optional name of the package to which generated implementation classes belong. This defines your context path. If it is not specified, a package name of <code>jaxbderived.&lt;schema name&gt;</code> is used where <code>&lt;schema name&gt;</code> is the name of the schema specified by the <b>Schema File</b> field.

The TopLink JAXB compiler generates JAXB-specific files (see [Section 47.1.1.2, "Working with JAXB-Specific Generated Files"](#)) and TopLink-specific files.

## 48.2.2 How to Create an XML Project from an XML Schema Using the Command Line

To create a new, mapped Oracle TopLink Workbench project from an XML schema using JAXB from the command line, use the `tljxb.cmd` or `tljxb.sh` file (located in the `<TOPLINK_HOME>/bin` directory) as follows:

1. Using a text editor, edit the `tljxb.cmd` or `tljxb.sh` file to set proxy settings (if required).

If you are using a schema that imports another schema by URL and you are operating behind a proxy, then you must uncomment the lines shown in [Example 48–1](#) or [Example 48–2](#) and edit them to set your proxy host (name or IP address) and port:

### Example 48–1 Proxy Settings in `tljxb.cmd`

```
@REM set JVM_ARGS=%JVM_ARGS% -DproxySet=true -Dhttp.proxyHost= -Dhttp.proxyPort=
```

### Example 48–2 Proxy Settings in `tljxb.sh`

```
# JVM_ARGS="{JVM_ARGS}" -DproxySet=true -Dhttp.proxyHost= -Dhttp.proxyPort=
```

- Execute the `tljxb.cmd` or `tljxb.sh` file (located in the `<TOPLINK_HOME>/bin` directory).

The TopLink JAXB compiler generates JAXB-specific files (see [Section 47.1.1.2, "Working with JAXB-Specific Generated Files"](#)) and TopLink-specific files.

[Example 48–3](#) illustrates how to generate an object model from a schema using the Toplink JAXB compiler. [Table 48–1](#) lists the compiler arguments.

**Example 48–3 Generating an Object Model from a Schema with `tljxb.cmd`**

```
tljxb.cmd -sourceDir ./app/src -generateWorkbench -workbenchDir ./app/mw -schema
purchaseOrder.xsd -targetPkg examples.ox.model.if -implClassPkg
examples.ox.model.impl
```

**Table 48–1 TopLink JAXB Binding Compiler Arguments**

Argument	Description	Optional?
<code>-help</code>	Prints this usage information.	Yes
<code>-version</code>	Prints the release version of the TopLink JAXB compiler.	Yes
<code>-sourceDir</code>	The path to the directory into which generated interfaces, implementation classes, and deployment files are written. Default: directory named <code>source</code> in the specified output directory.	Yes
<code>-generateWorkbench</code>	Generate a TopLink Workbench project and necessary project files. If omitted, only runtime information is generated.	Yes
<code>-workbenchDir</code>	The path to the directory into which the TopLink Workbench project files are written. This argument requires the <code>-generateWorkbench</code> argument. Default: directory named <code>mw</code> in the specified output directory.	Yes
<code>-schema</code>	The fully qualified path to your XSD file.	No
<code>-targetPkg</code>	The name of the package to which both generated interfaces and classes belong. This defines your context path. To specify a different package for implementation classes, set the <code>-implClassPkg</code> argument. Default: a package name of <code>jaxbderived.&lt;schema name&gt;</code> where <code>&lt;schema name&gt;</code> is the name of the schema specified by the <code>-schema</code> argument.	Yes
<code>-implClassPkg</code>	The name of the package to which generated implementation classes belong. If this option is set, interfaces belong to the package specified by the <code>-targetPkg</code> argument. This defines your context path.	Yes
<code>-interface</code>	Generate only interfaces. This argument is optional. Default: generate both interfaces and implementation classes.	Yes
<code>-verbose</code>	The interfaces and classes generated. This argument is optional. Default: not verbose.	Yes
<code>-customize</code>	The fully qualified path and file name of a standard JAXB customization file that you can use to override default JAXB compiler behavior.	Yes

## Configuring an XML Project

This chapter describes the various components that you must configure to use an XML project.

This chapter includes the following section:

- [Introduction to XML Project Configuration](#)

For information on how to configure TopLink project options common to two or more project types, see [Chapter 117, "Configuring a Project"](#).

### 49.1 Introduction to XML Project Configuration

[Table 49–1](#) lists the configurable options for XML projects.

**Table 49–1** Configurable Options for XML Projects

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Project save location (see <a href="#">Section 117.2, "Configuring Project Save Location"</a> )		✓	
Project classpath (see <a href="#">Section 117.3, "Configuring Project Classpath"</a> )		✓	
Project comments (see <a href="#">Section 117.14, "Configuring Project Comments"</a> )	✓	✓	
Method or direct field access (see <a href="#">Section 117.4, "Configuring Method or Direct Field Access at the Project Level"</a> )	✓	✓	
Project deployment XML options (see <a href="#">Section 117.8, "Configuring Project Deployment XML Options"</a> )		✓	
XML parser platform (see <a href="#">Section 8.2.2.1, "Configuring XML Parser Platform"</a> )			✓
Importing an XML schema (see <a href="#">Section 5.6.3, "How to Import an XML Schema"</a> )	✓	✓	
XML schema namespace (see <a href="#">Section 5.6.5, "How to Configure XML Schema Namespace"</a> )	✓	✓	✓
Model Java source code options (see <a href="#">Section 117.9, "Configuring Model Java Source Code Options"</a> )	✓	✓	
Default descriptor advanced properties (see <a href="#">Section 117.6, "Configuring Default Descriptor Advanced Properties"</a> )	✓	✓	

For more information, see [Chapter 47, "Introduction to XML Projects"](#).





# Part XV

---

## XML Descriptors

This part contains general information about XML descriptors, as well as detailed information on how to create and configure these descriptors.

This part contains the following chapters:

- [Chapter 50, "Introduction to XML Descriptors"](#)

This chapter introduces options specific to an XML descriptor.

- [Chapter 51, "Creating an XML Descriptor"](#)

This chapter explains how to create descriptor options specific to an XML descriptor.

- [Chapter 52, "Configuring an XML Descriptor"](#)

This chapter explains how to configure descriptor options specific to an XML descriptor.



---

---

## Introduction to XML Descriptors

This chapter introduces options specific to an XML descriptor.

This chapter includes the following section:

- [XML Descriptor Concepts](#)

For information on descriptor concepts and features common to more than one type of TopLink descriptors, see [Chapter 16, "Introduction to Descriptors"](#).

### 50.1 XML Descriptor Concepts

XML descriptors describe Java objects that you map to simple and complex types defined by an XML schema document (XSD).

Using XML descriptors in an XML project, you can configure XML mappings (see [Section 53.1, "XML Mapping Types"](#)), in memory, to XML elements defined by an XSD.

For more information, see the following:

- [Chapter 51, "Creating an XML Descriptor"](#)
- [Chapter 52, "Configuring an XML Descriptor"](#)

#### 50.1.1 XML Descriptors and Aggregation

When working with descriptors for a parent (source) and a child (target) objects, you have to accomplish the following:

- if the source object exists, then you must ensure that the target object also exists;
- if the source object is destroyed, then you must ensure that the target object is also destroyed.

For more information, see [Section 16.2.5, "Descriptors and Aggregation"](#).

In your XML project, designate the descriptors for the source and target objects to reflect this relationship as [Composite Descriptors in XML Projects](#).

##### 50.1.1.1 Composite Descriptors in XML Projects

In an XML project, descriptors are always composites.

Because XML descriptors are always composites, you can configure inheritance for an XML descriptor without considering its type (see [Section 16.3, "Descriptors and Inheritance"](#)).



---

---

## Creating an XML Descriptor

This chapter explains how to create descriptor options specific to an XML descriptor.

This chapter includes the following sections:

- [Introduction to XML Descriptor Creation](#)
- [Creating an XML Descriptor](#)

For information on how to create more than one type of descriptor, see [Chapter 118](#), "Creating a Descriptor".

### 51.1 Introduction to XML Descriptor Creation

After you create a descriptor, you must configure its various options (see [Chapter 119](#), "Configuring a Descriptor") and use it to define mappings.

For complete information on the various types of mapping that TopLink supports, see [Chapter 17](#), "Introduction to Mappings" and [Chapter 120](#), "Creating a Mapping".

For complete information on the various types of descriptor that TopLink supports, see [Section 16.1](#), "Descriptor Types".

For more information, see the following:

- [Chapter 16](#), "Introduction to Descriptors"
- [Chapter 50](#), "Introduction to XML Descriptors"

### 51.2 Creating an XML Descriptor

You can create an XML descriptor using Oracle JDeveloper TopLink Editor, TopLink Workbench (see [Section 51.2.1](#), "How to Create an XML Descriptor Using TopLink Workbench"), or Java code (see [Section 51.2.2](#), "How to Create an XML Descriptor Using Java").

#### 51.2.1 How to Create an XML Descriptor Using TopLink Workbench



When you add a class to an XML project (see [Section 117.3](#), "Configuring Project Classpath"), TopLink Workbench creates an XML descriptor for the class.

An XML descriptor is always a composite type.

#### 51.2.2 How to Create an XML Descriptor Using Java

[Example 51-1](#) shows how to create an XML descriptor using Java code.

**Example 51-1 Creating an XML Descriptor in Java**

```
XMLDescriptor descriptor = new XMLDescriptor();  
descriptor.setJavaClass(YourClass.class);
```

---

---

**Note:** Use the `oracle.toplink.ox.XMLDescriptor` class. Do not use the deprecated `oracle.toplink.xml.XMLDescriptor` class.

---

---

## Configuring an XML Descriptor

This chapter explains how to configure descriptor options specific to an XML descriptor.

This chapter includes the following sections:

- [Introduction to XML Descriptor Configuration](#)
- [Configuring Schema Context for an XML Descriptor](#)
- [Configuring for Complex Type of anyType](#)
- [Configuring Default Root Element](#)
- [Configuring Document Preservation](#)

For information on how to configure descriptor options common to two or more descriptor types, see [Chapter 119, "Configuring a Descriptor"](#).

### 52.1 Introduction to XML Descriptor Configuration

[Table 52–1](#) lists the default configurable options for an XML descriptor.

**Table 52–1** Configuration Options for XML Descriptors

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
XML schema namespace (see <a href="#">Section 5.6.5, "How to Configure XML Schema Namespace"</a> )	✓	✓	✓
XML schema reference (see <a href="#">Section 5.6.4, "How to Configure an XML Schema Reference"</a> )	✓	✓	✓
Schema context (see <a href="#">Section 52.2, "Configuring Schema Context for an XML Descriptor"</a> )	✓	✓	✓
Complex type of anyType (see <a href="#">Section 52.3, "Configuring for Complex Type of anyType"</a> )	✓	✓	
Default root element (see <a href="#">Section 52.4, "Configuring Default Root Element"</a> )	✓	✓	✓
Document preservation (see <a href="#">Section 52.5, "Configuring Document Preservation"</a> )	✓	✓	✓
Comments (see <a href="#">Section 119.6, "Configuring Descriptor Comments"</a> )	✓	✓	
Classes (see <a href="#">Section 5.7.2, "How to Configure Classes"</a> )	✓	✓	
Inheritance for a child class descriptor (see <a href="#">Section 119.20, "Configuring Inheritance for a Child (Branch or Leaf) Class Descriptor"</a> )	✓	✓	✓
Inheritance for a parent descriptor (see <a href="#">Section 119.21, "Configuring Inheritance for a Parent (Root) Descriptor"</a> )	✓	✓	✓

**Table 52–1 (Cont.) Configuration Options for XML Descriptors**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Inherited attribute mapping in a subclass (see Section 119.23, "Configuring Inherited Attribute Mapping in a Subclass")	✓	✓	✓
Instantiation policy (see Section 119.28, "Configuring Instantiation Policy")	✓	✓	✓
Copy policy (see Section 119.29, "Configuring Copy Policy")	✓	✓	✓
Amendment methods (see Section 119.35, "Configuring Amendment Methods")	✓	✓	✓
Mapping (see Section 121, "Configuring a Mapping")	✓	✓	✓

For more information, see [Chapter 50, "Introduction to XML Descriptors"](#).

## 52.2 Configuring Schema Context for an XML Descriptor

Oracle JDeveloper TopLink Editor and TopLink Workbench use the schema context to associate the XML descriptor reference class with a simple or complex type in one of the schemas associated with the XML project (see [Section 5.6.4, "How to Configure an XML Schema Reference"](#)). This allows Oracle JDeveloper TopLink Editor and TopLink Workbench to display the appropriate attributes available for mapping in that context.

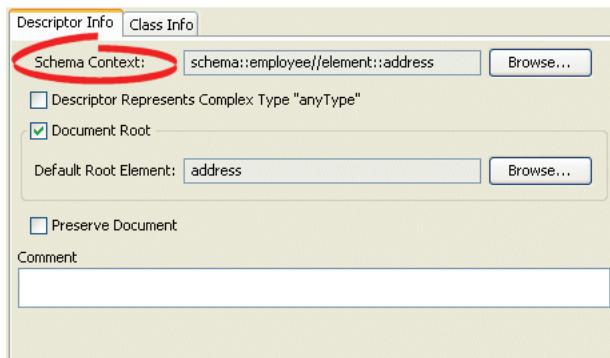
You must configure the schema context for an XML descriptor regardless of whether or not you are using Oracle JDeveloper or TopLink Workbench.

The TopLink runtime uses the schema context to validate XML fragments.

### 52.2.1 How to Configure Schema Context for an XML Descriptor Using TopLink Workbench

To associate an XML descriptor with a specific schema complex type, use this procedure:

1. Select an XML descriptor in the **Navigator**. Its properties appear in the Editor.
2. Click the **Descriptor Info** tab. The Descriptor Info tab appears.

**Figure 52–1 Descriptor Info Tab, Schema Context Option**

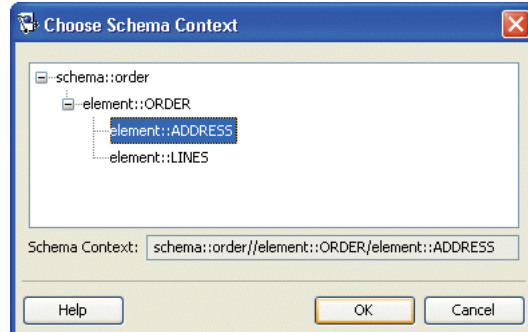
Click **Browse** to select the schema element to associate with this descriptor. For more information, see [Section 52.2.1.1, "Choosing a Schema Context"](#).



### 52.2.1.1 Choosing a Schema Context

Use the Choose Schema Context dialog box to select a specific schema element (such as when mapping an element).

**Figure 52–2 Choose Schema Context Dialog Box**



Select a schema element and click **OK**.

## 52.2.2 How to Configure Schema Context for an XML Descriptor Using Java

To configure an XML descriptor with a schema context using Java, create a descriptor amendment method (see [Section 119.35, "Configuring Amendment Methods"](#)) that uses `XMLSchemaReference` method `setSchemaContext`, as [Example 52–1](#) shows.

**Example 52–1 Configuring Schema Context**

```
public void addToDescriptor(ClassDescriptor descriptor) {
    descriptor.getSchemaReference().setSchemaContext(xpath);
}
```

## 52.3 Configuring for Complex Type of anyType

This attribute applies only to TopLink Workbench. Use this option to solve "No schema context is specified" problems (see [Section 5.3.5, "How to Use the Problems Window"](#)) for an XML descriptor that does not represent an element in your XML schema.

In general, TopLink Workbench assumes that every XML descriptor must have a schema context (see [Section 52.2, "Configuring Schema Context for an XML Descriptor"](#)). However, if a class in your project does not relate to an element in your schema, then it does not have a schema context.

For example, consider the schema that [Example 52–2](#) shows.

**Example 52–2 Schema Using `xsd:anyType`**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="customer" type="customer-type"/>
  <xsd:complexType name="customer-type">
    <xsd:sequence>
      <xsd:element name="contact-method" type="xsd:anyType"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="address">
    <xsd:complexType>
      <xsd:sequence>
```

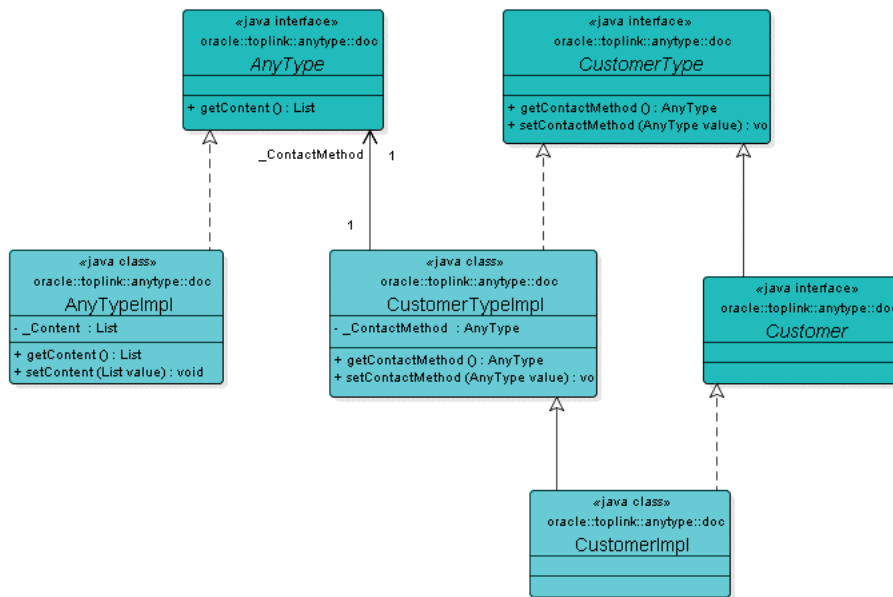
```

        <xsd:element name="street" type="xsd:string"/>
        <xsd:element name="city" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="phone-number" type="xsd:string"/>
</xsd:schema>

```

Because element `contact-method` is of type `xsd:anyType`, your project requires a class to represent that type, such as class `AnyTypeImpl` shown in Figure 52-3. Because this class does not relate to any complex type in your schema, it has no schema context. In this example, you would select this option for the `AnyTypeImpl` class.

**Figure 52-3 Class Representing `xsd:anyType`**




---

**Note:** See also Section 54.4, "Configuring Maps to Wildcard".

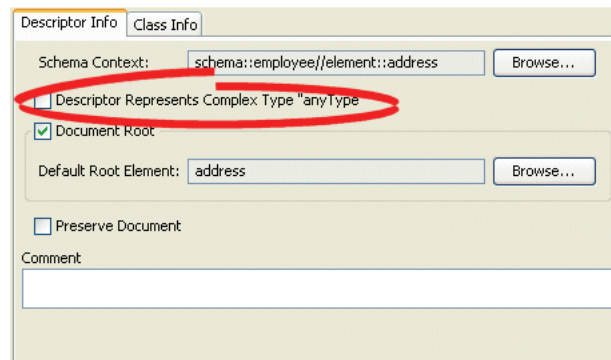
---

For more information, see Section 53.2.5, "xs:any and xs:anyType Support".

### 52.3.1 How to Configure Complex Type of anyType Using TopLink Workbench

To specify that the descriptor represents a complex type of `anyType`, use this procedure:

1. Select a descriptor in the **Navigator**. Its properties appear in the Editor.
2. Click the **Descriptor Info** tab. The Descriptor Info tab appears.

**Figure 52–4 Descriptor Info Tab, Complex Type "anyType" Option**

Select the **Descriptor Represents Complex Type "anyType"** option to specify this descriptor as the root element.

## 52.4 Configuring Default Root Element

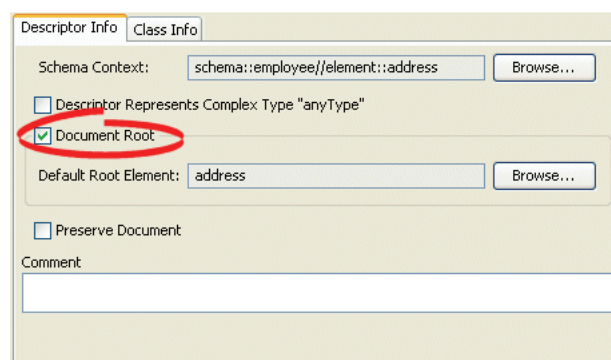
The default root element is the name that TopLink uses for the root element when marshalling objects for this descriptor to, and unmarshalling from, an XML document. Descriptors used only in composite relationship mappings do not require a default root element.

For more information, see [Section 16.2.12, "Default Root Element"](#).

### 52.4.1 How to Configure Default Root Element Using TopLink Workbench

To specify a schema element as the default root element for the descriptor, use this procedure:

1. Select a descriptor in the **Navigator**. Its properties appear in the Editor.
2. Click the **Descriptor Info** tab. The Descriptor Info tab appears.

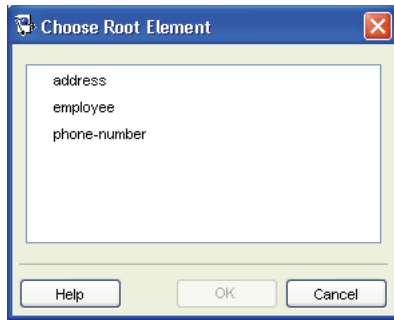
**Figure 52–5 Descriptor Info Tab, Default Root Option**

Select the **Default Root Element** option to specify this descriptor as the root element.

Click **Browse** to select the schema element to identify as the root element for this descriptor. See [Section 52.4.1.1, "Choosing a Root Element"](#) for more information.

#### 52.4.1.1 Choosing a Root Element

Use the Choose Root Element dialog box to select a specific root element.

**Figure 52–6 Choose Root Element Dialog Box**

Select the root element and click **OK**.

## 52.5 Configuring Document Preservation

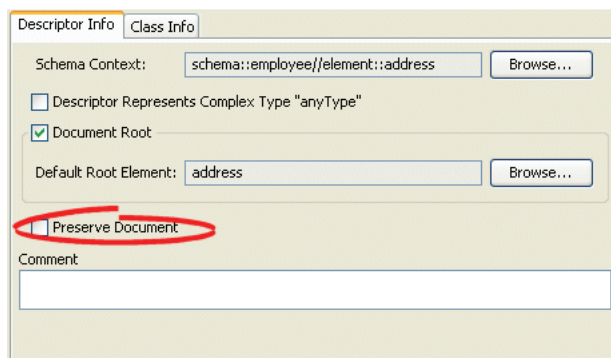
TopLink lets you preserve any "extra" data in your XML source that is not required to map to an object model (such as comments, processing instructions, or unmapped elements).

This permits round-tripping from XML to objects and back to XML without losing any data.

### 52.5.1 How to Configure Document Preservation Using TopLink Workbench

To preserve the entire XML source document, use this procedure:

1. Select a descriptor in the **Navigator**. Its properties appear in the Editor.
2. Click the **Descriptor Info** tab. The Descriptor Info tab appears.

**Figure 52–7 Descriptor Info Tab, Preserve Document Option**

Select the **Preserve Document** option to maintain any extra information from the source XML document that TopLink does not require (such as comments).

### 52.5.2 How to Configure Document Preservation Using Java

To configure an XML descriptor to maintain any extra information from the source XML document that TopLink does not require (such as comments) using Java, create a descriptor amendment method (see [Section 119.35, "Configuring Amendment Methods"](#)) that configures the descriptor using `XMLDescriptor` method `setShouldPreserveDocument`.

# Part XVI

---

## XML Mappings

An XML mapping transforms object data members to the XML nodes of an XML document, whose structure is defined by an XML schema document (XSD).

This part contains the following chapters:

- [Chapter 53, "Introduction to XML Mappings"](#)

This chapter describes each of the different TopLink XML mapping types and important XML mapping concepts.
- [Chapter 54, "Configuring an XML Mapping"](#)

This chapter explains how to configure TopLink XML mapping options common to two or more XML mapping types.
- [Chapter 55, "Configuring an XML Direct Mapping"](#)

This chapter explains how to configure a direct XML mapping.
- [Chapter 56, "Configuring an XML Composite Direct Collection Mapping"](#)

This chapter explains how to configure a composite direct collection XML mapping.
- [Chapter 57, "Configuring an XML Composite Object Mapping"](#)

This chapter explains how to configure a composite object XML mapping including an XML mapping to a single named complex type of type `xs:anyType`.
- [Chapter 58, "Configuring an XML Composite Collection Mapping"](#)

This chapter explains how to configure a composite collection XML mapping.
- [Chapter 59, "Configuring an XML Any Object Mapping"](#)

This chapter explains how to configure an XML mapping to a single unnamed complex type specified as `xs:any`.
- [Chapter 60, "Configuring an XML Any Collection Mapping"](#)

This chapter explains how to configure an XML mapping to an unnamed sequence of complex types specified as `xs:any`, an unnamed sequence of complex types of type `xs:anyType`, or a root element of type `xs:anyType`.
- [Chapter 61, "Configuring an XML Transformation Mapping"](#)

This chapter explains how to configure a transformation XML mapping.



---

---

## Introduction to XML Mappings

An XML mapping transforms object data members to the XML nodes of an XML document whose structure is defined by an XML schema document (XSD).

This chapter includes the following sections:

- XML Mapping Types
- XML Mapping Concepts
- XML Direct Mapping
- XML Composite Direct Collection Mapping
- XML Composite Object Mapping
- XML Composite Collection Mapping
- XML Any Object Mapping
- XML Any Collection Mapping
- XML Transformation Mapping
- XML Object Reference Mapping
- XML Collection Reference Mapping
- XML Binary Data Mapping
- XML Binary Data Collection Mapping
- XML Fragment Mapping
- XML Fragment Collection Mapping
- XML Choice Object Mapping
- XML Choice Collection Mapping
- XML Any Attribute Mapping

For information on mapping concepts and features common to more than one type of TopLink mappings, see [Chapter 17, "Introduction to Mappings"](#).

### 53.1 XML Mapping Types

TopLink supports the XML mappings listed in [Table 53-1](#).

**Table 53–1 TopLink XML Mapping Types**

Mapping Type	Description	Oracle JDeveloper	TopLink Workbench	Java
XML direct mapping (see <a href="#">Section 53.3, "XML Direct Mapping"</a> )	Map a simple object attribute to an XML attribute or text node.	✓	✓	✓
XML composite direct collection mapping (see <a href="#">Section 53.4, "XML Composite Direct Collection Mapping"</a> )	Map a collection of simple object attributes to XML attributes or text nodes.	✓	✓	✓
XML composite object mapping (see <a href="#">Section 53.5, "XML Composite Object Mapping"</a> )	Map any attribute that contains a single object to an XML element. The TopLink runtime uses the descriptor for the referenced object to populate the contents of that element.	✓	✓	✓
XML composite collection mapping (see <a href="#">Section 53.6, "XML Composite Collection Mapping"</a> )	Map an attribute that contains a homogenous collection of objects to multiple XML elements. The TopLink runtime uses the descriptor for the referenced object to populate the contents of those elements.	✓	✓	✓
XML any object mapping (see <a href="#">Section 53.7, "XML Any Object Mapping"</a> )	The any object XML mapping is similar to the composite object XML mapping (see <a href="#">Section 53.5, "XML Composite Object Mapping"</a> ), except that the reference object may be of different types (including <code>String</code> ), not necessarily related to each other through inheritance or a common interface.	✓	✓	✓
XML any collection mapping (see <a href="#">Section 53.8, "XML Any Collection Mapping"</a> )	The any collection XML mapping is similar to the composite collection XML mapping (see <a href="#">Section 53.6, "XML Composite Collection Mapping"</a> ) except that the referenced objects may be of different types (including <code>String</code> ), not necessarily related to each other through inheritance or a common interface.	✓	✓	✓
XML transformation mapping (see <a href="#">Section 53.9, "XML Transformation Mapping"</a> )	Create custom mappings where one or more XML nodes can be used to create the object to be stored in a Java class's attribute.	✓	✓	✓
XML object reference mapping (see <a href="#">Section 53.10, "XML Object Reference Mapping"</a> )	Map a given element in an XML document to another element in that same XML document using key(s).  Use this mapping when several objects reference the same instance of another object.	✓		✓
XML collection reference mapping (see <a href="#">Section 53.11, "XML Collection Reference Mapping"</a> )	This mapping is similar to the XML object reference mapping (see <a href="#">Section 53.10, "XML Object Reference Mapping"</a> ), except that it deals with collections instead of single objects.  Use this mapping when several objects reference the same instance of another object.	✓		✓
XML binary data mapping (see <a href="#">Section 53.12, "XML Binary Data Mapping"</a> )	Handle binary data: this mapping maps binary data in the object model to XML.  Use this mapping to enable writing of binary data directly as inline binary data (base64 BLOB), or passing through as a MtOM or SwaRef attachment.	✓		✓



**Table 53–1 (Cont.) TopLink XML Mapping Types**

Mapping Type	Description	Oracle JDeveloper	TopLink Workbench	Java
XML binary data collection mapping (see <a href="#">Section 53.13</a> , "XML Binary Data Collection Mapping")	This mapping is similar to the XML binary data mapping (see <a href="#">Section 53.12</a> , "XML Binary Data Mapping"), except that it maps a collection of binary data in the object model to XML.	✓		✓
XML fragment mapping (see <a href="#">Section 53.14</a> , "XML Fragment Mapping")	Keep a part of an XML tree as a node.	✓		✓
XML fragment collection mapping (see <a href="#">Section 53.15</a> , "XML Fragment Collection Mapping")	This mapping is similar to the XML fragment mapping (see <a href="#">Section 53.14</a> , "XML Fragment Mapping"), except that it allows you to keep a part of an XML tree as a collection of nodes.	✓		✓
XML choice object mapping (see <a href="#">Section 53.16</a> , "XML Choice Object Mapping")	Map a single attribute to a number of different elements in an XML document. Use this mapping to map to single choices or substitution groups in an XML schema.	✓		✓
XML choice collection mapping (see <a href="#">Section 53.17</a> , "XML Choice Collection Mapping")	This mapping is similar to the XML choice object mapping (see <a href="#">Section 53.16</a> , "XML Choice Object Mapping"), except that you use it to handle reading and writing of XML documents containing a collection of choice or substitution group elements.	✓		✓
XML any attribute mapping (see <a href="#">Section 53.18</a> , "XML Any Attribute Mapping")	Map to an attribute in an object to any XML attributes contained on a specific element in the XML document.	✓		✓

## 53.2 XML Mapping Concepts

You can map the attributes of a Java object to a combination of XML simple and complex types using a wide variety of XML mapping types.

TopLink stores XML mappings for each class in the class descriptor. TopLink uses the descriptor to instantiate objects mapped from an XML document and to store new or modified objects as an XML document.

To configure XML mappings, Oracle recommends that you use Oracle JDeveloper TopLink Editor or TopLink Workbench and their GUI environment to set the descriptor properties and configure the mappings.

This section describes concepts unique to TopLink XML mappings, including the following:

- [Mapping to Simple and Complex Types](#)
- [Mapping Order](#)
- [XPath Support](#)
- [xsd:list and xsd:union Support](#)
- [xs:any and xs:anyType Support](#)
- [jaxb:class Support](#)
- [Typesafe Enumeration Support](#)
- [Mapping Extensions](#)
- [Key On Source-Based Mapping Support](#)
- [Substitution Groups](#)

- [Mixed Content Mapping](#)
- [XML Adapter](#)

### 53.2.1 Mapping to Simple and Complex Types

Consider the XML document shown in [Example 53–1](#).

**Example 53–1 XML Document**

```
<EMPLOYEE ID="123">
  <NAME>Jane Doe</NAME>
  <ADDRESS>
    <STREET>123 Any St.</STREET>
    <CITY>MyCity</CITY>
  </ADDRESS>
</EMPLOYEE>
```

In general, using TopLink XML mappings, you can map a Java class to a simple type (such as NAME) or to a complex type (such as ADDRESS).

Specifically, you can map a Java object's simple attributes to XML attributes (such as ID) and text nodes (such as NAME). You can also map a Java object's relationships to XML elements (such as ADDRESS).

[Table 53–2](#) summarizes the XML simple and complex types supported by each TopLink XML mapping.

**Table 53–2 XML Mapping Support for XML Simple and Complex Types**

Mapping	XML Attribute	XML Text Node	XML Element
XML direct (see <a href="#">Section 53.3, "XML Direct Mapping"</a> )	✓	✓	
XML composite direct collection (see <a href="#">Section 53.4, "XML Composite Direct Collection Mapping"</a> )	✓	✓	
XML composite object (see <a href="#">Section 53.5, "XML Composite Object Mapping"</a> )			✓
XML composite collection (see <a href="#">Section 53.6, "XML Composite Collection Mapping"</a> )			✓
XML any object (see <a href="#">Section 53.7, "XML Any Object Mapping"</a> )			✓
XML any collection (see <a href="#">Section 53.8, "XML Any Collection Mapping"</a> )			✓
XML transformation (see <a href="#">Section 53.9, "XML Transformation Mapping"</a> )	✓	✓	✓
XML object reference (see <a href="#">Section 53.10, "XML Object Reference Mapping"</a> )	✓		
XML collection reference (see <a href="#">Section 53.11, "XML Collection Reference Mapping"</a> )	✓		
XML binary data (see <a href="#">Section 53.12, "XML Binary Data Mapping"</a> )	✓	✓	
XML binary data collection (see <a href="#">Section 53.13, "XML Binary Data Collection Mapping"</a> )	✓	✓	

**Table 53–2 (Cont.) XML Mapping Support for XML Simple and Complex Types**

Mapping	XML Attribute	XML Text Node	XML Element
XML fragment (see <a href="#">Section 53.14</a> , "XML Fragment Mapping")	✓	✓	
XML fragment collection (see <a href="#">Section 53.15</a> , "XML Fragment Collection Mapping")	✓	✓	
XML choice object (see <a href="#">Section 53.16</a> , "XML Choice Object Mapping")	✓		✓
XML choice collection (see <a href="#">Section 53.17</a> , "XML Choice Collection Mapping")	✓		✓
XML any attribute (see <a href="#">Section 53.18</a> , "XML Any Attribute Mapping")	✓		

### 53.2.2 Mapping Order

Unlike relational database mappings, the order in which mappings are persisted in XML is significant.

The order in which you define XML mappings in TopLink (whether in Oracle JDeveloper TopLink Editor, TopLink Workbench, or in Java code) including the order in which you define mapping components such as `Transformers` (see [Section 53.9](#), "XML Transformation Mapping") is reflected in the order, in which TopLink persists data in an XML document.

### 53.2.3 XPath Support

TopLink uses XPath statements to efficiently map the attributes of a Java object to locations in an XML document.

The following are main characteristics of XPath:

- Each XPath statement is relative to the context node specified in the descriptor.
- The XPath may contain node type, path, and positional information.
- The XPath is specified on a mapping using the `setXPath` method.

For more information about using XPath with XML mappings, see [Section 17.2.7](#), "Mappings and XPath".

### 53.2.4 xsd:list and xsd:union Support

You can use XML direct (see [Section 53.3](#), "XML Direct Mapping") and composite direct collection (see [Section 53.4](#), "XML Composite Direct Collection Mapping") mappings, as well as their subclasses, to map to `xsd:list` and `xsd:union` types in an XML document.

For more information, see [Section 17.2.8](#), "Mappings and `xsd:list` and `xsd:union` Types".

### 53.2.5 xs:any and xs:anyType Support

In an XML schema, you can define elements and complex types that correspond to any data type using `xs:any` and `xs:anyType`. You can map objects to such elements and complex types using XML mappings `XMLAnyObjectMapping` and `XMLAnyCollectionMapping`.

Table 53–3 lists the XML mappings to use with common applications of `xs:any` and `xs:anyType`. For more details, see the specified XML mapping type.

**Table 53–3 XML Mappings and XML Schema `xs:any` and `xs:anyType`**

Use XML Mapping...	To Map XML Schema Definition...
See <a href="#">Section 53.7, "XML Any Object Mapping"</a>	Element with a single <sup>1</sup> unnamed complex type specified as <code>xs:any</code> .
See <a href="#">Section 53.8, "XML Any Collection Mapping"</a>	Element with an unnamed sequence <sup>2</sup> of complex types specified as <code>xs:any</code> .  Element with a named sequence <sup>2</sup> of complex types of type <code>xs:anyType</code> .  Root element of type <code>xs:anyType</code> .

<sup>1</sup> `minOccurs` and `maxOccurs` are both equal to 1.

<sup>2</sup> `maxOccurs` is greater than 1.

### 53.2.6 `jaxb:class` Support

You can configure an XML composite object mapping (see [Section 53.5, "XML Composite Object Mapping"](#)) and its subclasses to accommodate `jaxb:class` customizations with the following XSD structures:

- `all`
- `sequence`
- `choice`
- `group`

For more information, see [Section 17.2.9, "Mappings and the `jaxb:class` Customization"](#).

### 53.2.7 Typesafe Enumeration Support

You can map a Java attribute to such a typesafe enumeration using the `JAXBTypesafeEnumConverter` with an `XMLDirectMapping`, `XMLCompositeDirectCollectionMapping` or their subclasses with XML documents.

For more information, see [Section 17.2.10, "Mappings and JAXB Typesafe Enumerations"](#)

### 53.2.8 Mapping Extensions

If existing TopLink XML mappings do not meet your needs, you can create custom XML mappings using XML mapping extensions, including object type, serialized object, type conversion converters, and a simple type translator. For more information, see [Section 17.2.6, "Mapping Converters and Transformers"](#).

### 53.2.9 Key On Source-Based Mapping Support

TopLink XML support for key on source-based mapping lets you use one-to-one and one-to-many mappings to map a given element in an XML document to another element in that same XML document using key(s).

You use this mapping when several objects reference the same instance of another object.

Use the `oracle.toplink.ox.mappings.XMLObjectReferenceMapping` and `XMLCollectionReferenceMapping` for the key on source.

You configure this mappings using the deployment XML and a project class.

For more information, see the following:

- [XML Object Reference Mapping](#)
- [XML Collection Reference Mapping](#)

### 53.2.10 Substitution Groups

Substitution groups is a mechanism provided by the XML schema. Using substitution groups, you can substitute elements for other elements. For more information, see *XML Schema Primer* at

<http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/#SubsGroups>

Use `TopLinkXMLChoiceObjectMapping` (see [Section 53.16, "XML Choice Object Mapping"](#)) and `XMLChoiceCollectionMapping` (see [Section 53.17, "XML Choice Collection Mapping"](#)) to handle substitution groups.

### 53.2.11 Mixed Content Mapping

To handle mixed content, such as reading in XML text nodes as strings, use `TopLinkXMLAnyCollectionMapping` (see [Section 53.8, "XML Any Collection Mapping"](#)).

Enable this functionality through the `setMixedContent` method.

### 53.2.12 XML Adapter

TopLink supports the use of the `javax.xml.bind.annotation.adapters.XmlAdapter<ValueType, BoundType>` and its subclasses, which allow for arbitrary Java classes to be used with JAXB.

The `XmlAdapter` enables adaptation of a Java type for custom marshaling through its `marshal` and `unmarshal` methods.

For more information, refer to JAXB 2.0 Specification at

<https://jaxb.dev.java.net/nonav/jaxb20-pfd/api/index.html>

## 53.3 XML Direct Mapping

XML direct mappings map a Java attribute directly to XML text nodes. You can use an XML direct mapping in the following scenarios:

- [Mapping to a Text Node](#)
- [Mapping to an Attribute](#)
- [Mapping to a Specified Schema Type](#)
- [Mapping to a List Field with an XML Direct Mapping](#)
- [Mapping to a Union Field with an XML Direct Mapping](#)
- [Mapping to a Union of Lists with an XML Direct Mapping](#)
- [Mapping to a Union of Unions with an XML Direct Mapping](#)
- [Mapping with a Simple Type Translator](#)

See [Chapter 55, "Configuring an XML Direct Mapping"](#) for more information.

---



---

**Note:** Do not confuse an XML direct mapping with a relational direct-to-XMLType mapping (see [Section 27.4, "Direct-to-XMLType Mapping"](#)).

---



---

### 53.3.1 Mapping to a Text Node

This section describes using an XML direct mapping when:

- [Mapping to a Simple Text Node](#)
- [Mapping to a Text Node in a Simple Sequence](#)
- [Mapping to a Text Node in a Subelement](#)
- [Mapping to a Text Node by Position](#)

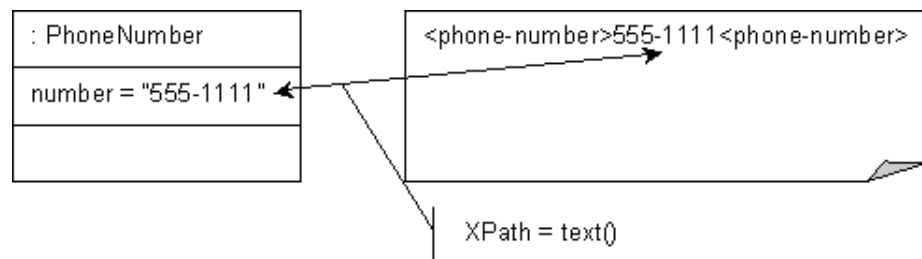
#### 53.3.1.1 Mapping to a Simple Text Node

Given the XML schema in [Example 53–2](#), [Figure 53–1](#) illustrates an XML direct mapping to a simple text node in a corresponding XML document. [Example 53–3](#) shows how to configure this mapping in Java.

**Example 53–2 Schema for XML Direct Mapping to Simple Text Node**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="phone-number" type="xsd:string"/>
</xsd:schema>
```

**Figure 53–1 XML Direct Mapping to Simple Text Node**



**Example 53–3 Java for XML Direct Mapping to Simple Text Node**

```
XMLDirectMapping numberMapping = new XMLDirectMapping();
numberMapping.setAttributeName("number");
numberMapping.setXPath("text()");
```

#### 53.3.1.2 Mapping to a Text Node in a Simple Sequence

Given the XML schema in [Example 53–4](#), [Figure 53–2](#) illustrates an XML direct mapping to individual text nodes in a sequence in a corresponding XML document. [Example 53–5](#) shows how to configure this mapping in Java.

**Example 53–4 Schema for XML Direct Mapping to a Text Node in a Simple Sequence**

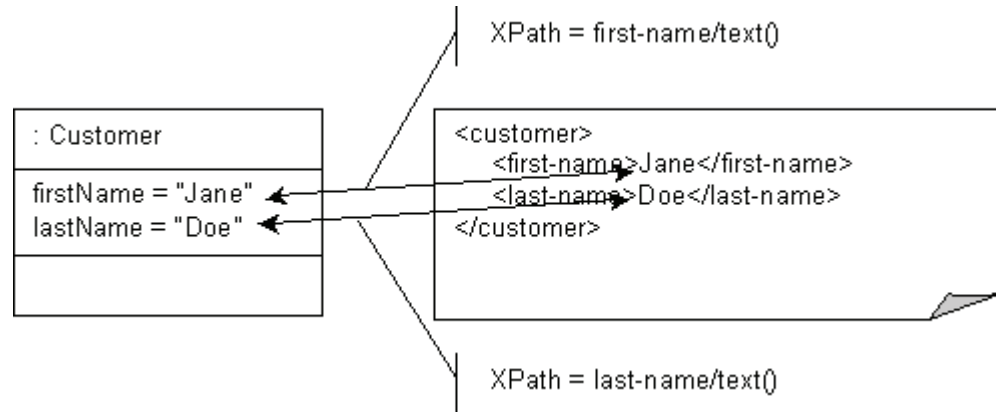
```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="customer" type="customer-type"/>
  <xsd:complexType name="customer-type">
    <xsd:sequence>
      <xsd:element name="first-name" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

```

        <xsd:element name="last-name" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>
</xsd:schema>

```

**Figure 53–2 XML Direct Mapping to a Text Node in a Simple Sequence**



**Example 53–5 Java for XML Direct Mapping to a Text Node in a Simple Sequence**

```

XMLDirectMapping firstNameMapping = new XMLDirectMapping();
firstNameMapping.setAttributeName("firstName");
firstNameMapping.setXPath("first-name/text()");

```

```

XMLDirectMapping lastNameMapping = new XMLDirectMapping();
lastNameMapping.setAttributeName("lastName");
lastNameMapping.setXPath("last-name/text()");

```

### 53.3.1.3 Mapping to a Text Node in a Subelement

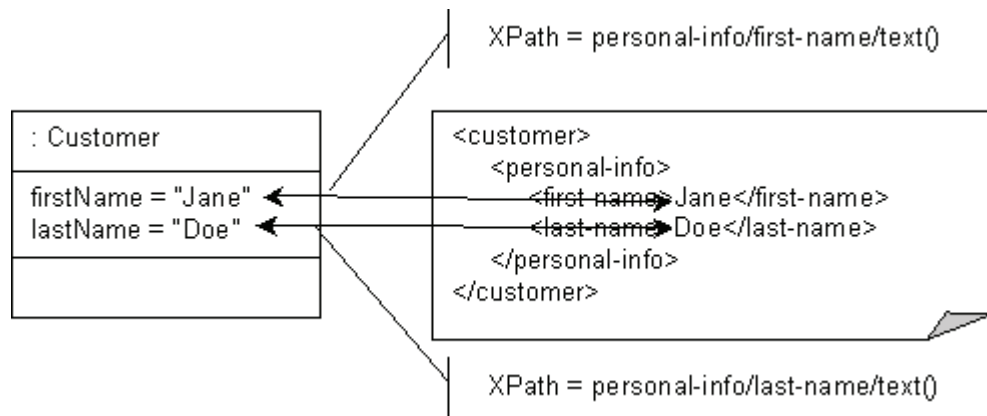
Given the XML schema in [Example 53–6](#), [Figure 53–3](#) illustrates an XML direct mapping to a text node in a subelement in a corresponding XML document. [Example 53–7](#) shows how to configure this mapping in Java.

**Example 53–6 Schema for XML Direct Mapping to a Text Node in a Subelement**

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="customer" type="customer-type"/>
    <xsd:complexType name="customer-type">
        <xsd:sequence>
            <xsd:element name="personal-info">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="first-name" type="xsd:string"/>
                        <xsd:element name="last-name" type="xsd:string"/>
                    <xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>

```

**Figure 53–3 XML Direct Mapping to a Text Node in a Subelement****Example 53–7 Java for XML Direct Mapping to a Text Node in a Subelement**

```
XMLDirectMapping firstNameMapping = new XMLDirectMapping();
firstNameMapping.setAttributeName("firstName");
firstNameMapping.setXPath("personal-info/first-name/text()");

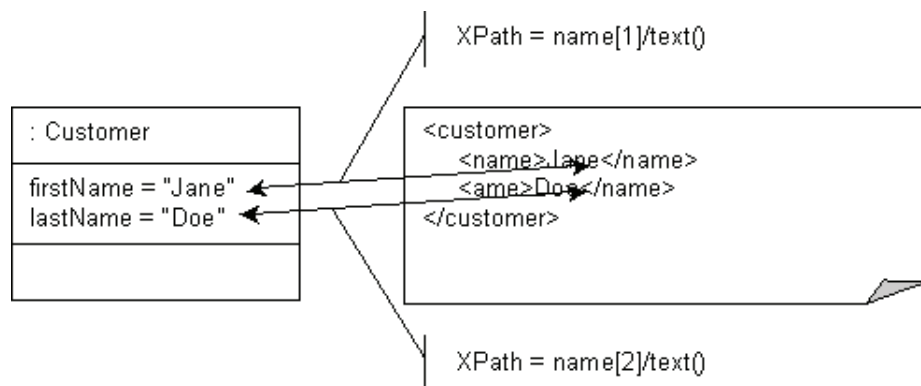
XMLDirectMapping lastNameMapping = new XMLDirectMapping();
lastNameMapping.setAttributeName("lastName");
lastNameMapping.setXPath("personal-info/last-name/text()");
```

### 53.3.1.4 Mapping to a Text Node by Position

Given the XML schema in [Example 53–8](#), [Figure 53–4](#) illustrates an XML direct mapping to a text node by position in a corresponding XML document. [Example 53–9](#) shows how to configure this mapping in Java.

**Example 53–8 Schema for XML Direct Mapping to Text Node by Position**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="customer" type="customer-type"/>
  <xsd:complexType name="customer-type">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string" maxOccurs="2"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

**Figure 53–4 XML Direct Mapping to Text Node by Position**



**Example 53–9 Java for XML Direct Mapping to Text Node by Position**

```
XMLDirectMapping firstNameMapping = new XMLDirectMapping();
firstNameMapping.setAttributeName("firstName");
firstNameMapping.setXPath("name[1]/text()");

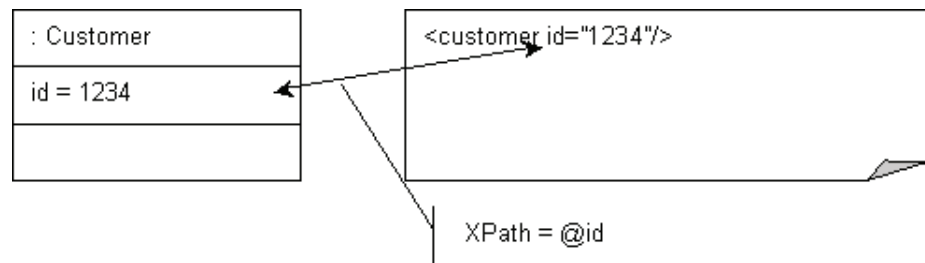
XMLDirectMapping lastNameMapping = new XMLDirectMapping();
lastNameMapping.setAttributeName("lastName");
lastNameMapping.setXPath("name[2]/text()");
```

**53.3.2 Mapping to an Attribute**

Given the XML schema in [Example 53–10](#), [Figure 53–5](#) illustrates an XML direct mapping to a text node by position in a corresponding XML document. [Example 53–11](#) shows how to configure this mapping in Java.

**Example 53–10 Schema for XML Direct Mapping to an Attribute**

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="customer" type="customer-type" />
  <xsd:complexType name="customer-type">
    <xsd:attribute name="id" type="xsd:integer" />
  </xsd:complexType>
</xsd:schema>
```

**Figure 53–5 XML Direct Mapping to an Attribute****Example 53–11 Java for XML Direct Mapping to an Attribute**

```
XMLDirectMapping idMapping = new XMLDirectMapping();
idMapping.setAttributeName("id");
idMapping.setXPath("@id");
```

**53.3.3 Mapping to a Specified Schema Type**

In most cases, TopLink can determine the target format in the XML document. However, there are cases where you must specify which one of a number of possible targets TopLink should use. For example, a `java.util.Calendar` could be marshalled to a schema `date`, `time`, or `dateTime` node, or a `byte[]` could be marshalled to a schema `hexBinary` or `base64Binary` node.

Given the XML schema in [Example 53–12](#), [Figure 53–6](#) illustrates an XML direct mapping to a text node by position in a corresponding XML document. [Example 53–13](#) shows how to configure this mapping in Java.

**Example 53–12 Schema for XML Direct Mapping to a Specified Schema Type**

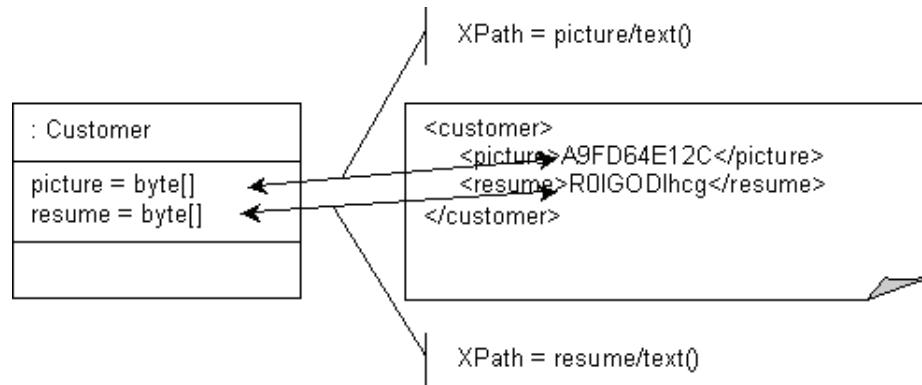
```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="customer" type="customer-type" />
  <xsd:complexType name="customer-type">
    <xsd:sequence>
```

```

    <xsd:element name="picture" type="xsd:hexBinary"/>
    <xsd:element name="resume" type="xsd:base64Binary"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>

```

**Figure 53–6 XML Direct Mapping to a Specified Schema Type**



**Example 53–13 Java for XML Direct Mapping to a Specified Schema Type**

```

XMLDirectMapping pictureMapping = new XMLDirectMapping();
pictureMapping.setAttributeName("picture");
pictureMapping.setXPath("picture/text()");
XMLField pictureField = (XMLField) pictureMapping.getField();
pictureField.setSchemaType(XMLConstants.HEX_BINARY_QNAME);

XMLDirectMapping resumeMapping = new XMLDirectMapping();
resumeMapping.setAttributeName("resume");
resumeMapping.setXPath("resume/text()");
XMLField resumeField = (XMLField) resumeMapping.getField();
resumeField.setSchemaType(XMLConstants.BASE_64_BINARY_QNAME);

```

### 53.3.4 Mapping to a List Field with an XML Direct Mapping

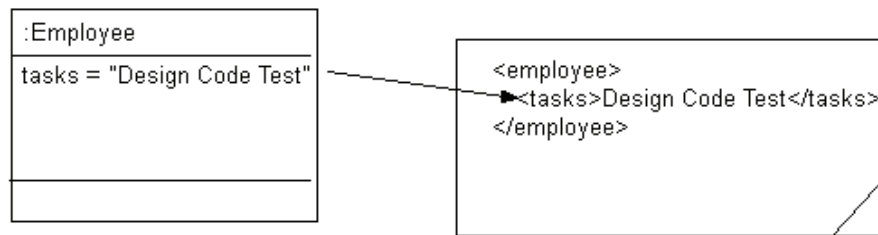
Given the XML schema in [Example 53–14](#), [Figure 53–7](#) illustrates an XML direct mapping to an `xsd:list` type in a corresponding XML document when you represent the list in your object model as a `String` of white space delimited tokens. [Example 53–15](#) shows how to configure this mapping in Java.

**Example 53–14 Schema for XML Direct Mapping to a List Field**

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="employee" type="employee-type"/>
  <xsd:complexType name="employee-type">
    <xsd:sequence>
      <xsd:element name="tasks" type="tasks-type"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:simpleType name="tasks-type">
    <xsd:list itemType="xsd:string"/>
  </xsd:simpleType>
</xsd:schema>

```

**Figure 53–7 XML Direct Mapping to a List Field****Example 53–15 Java for XML Direct Mapping to a List Field Node**

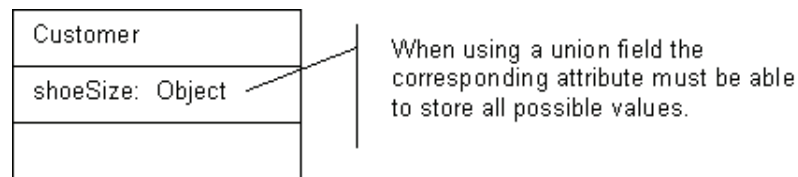
```
XMLDirectMapping tasksMapping = new XMLDirectMapping();
tasksMapping.setAttributeName("tasks");
XMLField myField = new XMLField("tasks/text()"); // pass in the XPath
myField.setUsesSingleNode(true);
tasksMapping.setField(myField);
```

### 53.3.5 Mapping to a Union Field with an XML Direct Mapping

Given the XML schema in [Example 53–16](#), [Figure 53–8](#) illustrates a Java class that can be mapped to a corresponding XML document. Note the `shoeSize` attribute in this class: when using a union field, the corresponding attribute must be able to store all possible values.

**Example 53–16 Schema for XML Direct Mapping to a Union Field**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="customer" type="customer-type"/>
  <xsd:complexType name="customer-type">
    <xsd:sequence>
      <xsd:element name="shoe-size" type="size-type"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:simpleType name="size-type">
    <xsd:union memberTypes="xsd:decimal xsd:string"/>
  </xsd:simpleType>
</xsd:schema>
```

**Figure 53–8 Java Class for XML Direct Mapping to a Union Field**

[Figure 53–9](#) illustrates an XML direct mapping to a union field in an XML document that conforms to the schema in [Example 53–16](#). When TopLink unmarshalls the XML document, it tries each of the union types until it can make a successful conversion. The first schema type in the union is `xsd:decimal`. Because "10.5" is a valid decimal, TopLink converts the value to the appropriate type. If the `Object` attribute is specific enough to trigger an appropriate value, TopLink will use that type instead. Otherwise, TopLink uses a default (in this case `BigDecimal`). You can override this behavior in Java code.

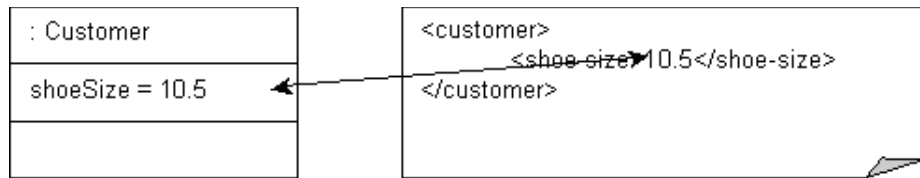
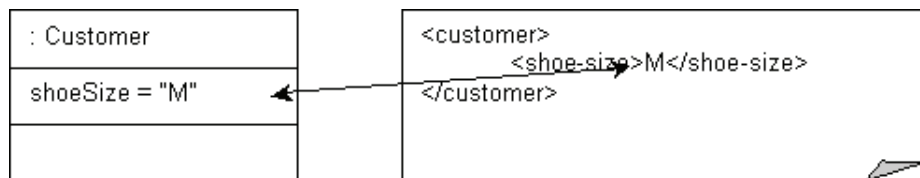
**Figure 53–9 XML Direct Mapping to the First Valid Union Type**

Figure 53–10 illustrates an XML direct mapping to union field in another XML document that conforms to the schema in Example 53–16. In this document, the value "M" is not a valid `xsd:decimal` type so the next union type is tried. The next union type is `xsd:string` and a conversion can be done.

**Figure 53–10 XML Direct Mapping to Another Valid Union Type**

Example 53–17 shows how to configure this mapping in Java.

**Example 53–17 Java for XML Direct Mapping to a Union Type**

```
XMLDirectMapping shoeSizeMapping = new XMLDirectMapping();
shoeSizeMapping.setAttributeName("shoeSize");
XMLUnionField shoeSizeField = new XMLUnionField();
shoeSizeField.setXPath("shoe-size/text()");
shoeSizeField.addSchemaType(XMLConstants.DECIMAL_QNAME);
shoeSizeField.addSchemaType(XMLConstants.STRING_QNAME);
shoeSizeMapping.setField(shoeSizeField);
```

To override the default conversion, use the `XMLUnionField` method `addConversion`:

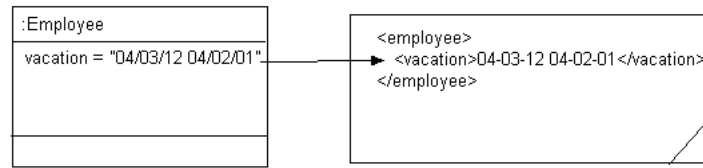
```
shoeSizeField.addConversion(XMLConstants.DECIMAL_QNAME, Float.class);
```

### 53.3.6 Mapping to a Union of Lists with an XML Direct Mapping

Given the XML schema in Example 53–18, Figure 53–11 illustrates an XML direct mapping to a union of lists in a corresponding XML document. Example 53–19 shows how to configure this mapping in Java.

**Example 53–18 Schema for XML Direct Mapping to Union of Lists**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="vacation" type="unionOfLists"/>
  <xsd:simpleType name="unionOfLists">
    <xsd:union memberTypes="xsd:double">
      <xsd:simpleType>
        <xsd:list itemType="xsd:date"/>
      </xsd:simpleType>
      <xsd:simpleType>
        <xsd:list itemType="xsd:integer"/>
      </xsd:simpleType>
    </xsd:union>
  </xsd:simpleType>
</xsd:schema>
```

**Figure 53–11 XML Direct Mapping to Union of Lists**

Note that in this example, valid XML documents contain either all `xsd:double`, all `xsd:date`, or all `xsd:integer` values.

**Example 53–19 Java for XML Direct Mapping to Union of Lists**

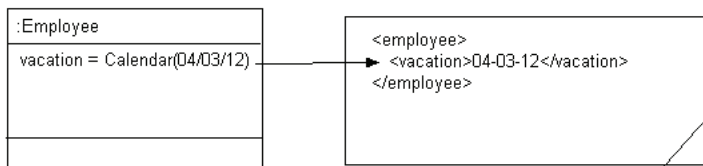
```
XMLDirectMapping mapping = new XMLDirectMapping();
mapping.setAttributeName("vacation");
mapping.setXPath("UnionOfLists/text()");
```

**53.3.7 Mapping to a Union of Unions with an XML Direct Mapping**

Given the XML schema in [Example 53–20](#), [Figure 53–12](#) illustrates a Java class that can be mapped to a corresponding XML document. [Example 53–21](#) shows how to configure this mapping in Java.

**Example 53–20 Schema for XML Direct Mapping to a Union of Unions**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="vacation" type="unionOfUnions"/>
  <xsd:simpleType name="unionOfUnions">
    <xsd:union>
      <xsd:simpleType>
        <xsd:union>
          <xsd:simpleType>
            <xsd:list itemType="xsd:date"/>
          </xsd:simpleType>
          <xsd:simpleType>
            <xsd:list itemType="xsd:integer"/>
          </xsd:simpleType>
        </xsd:union>
      </xsd:simpleType>
      <xsd:simpleType>
        <xsd:union>
          <xsd:simpleType>
            <xsd:list itemType="xsd:string"/>
          </xsd:simpleType>
          <xsd:simpleType>
            <xsd:list itemType="xsd:float"/>
          </xsd:simpleType>
        </xsd:union>
      </xsd:simpleType>
    </xsd:union>
  </xsd:simpleType>
</xsd:schema>
```

**Figure 53–12 Java Class for XML Direct Mapping to a Union of Unions****Example 53–21 Java for XML Direct Mapping to a Union of Unions**

```
XMLDirectMapping vacationMapping = new XMLDirectMapping();
vacationMapping.setAttributeName("vacation");
XMLUnionField vacationField = new XMLUnionField();
vacationField.setXPath("vacation/text()");
vacationField.addSchemaType(XMLConstants.DATE_QNAME);
vacationField.addSchemaType(XMLConstants.INTEGER_QNAME);
vacationField.addSchemaType(XMLConstants.STRING_QNAME);
vacationField.addSchemaType(XMLConstants.FLOAT_QNAME);
vacationMapping.setField(vacationField);
```

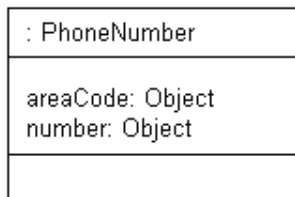
### 53.3.8 Mapping with a Simple Type Translator

If the type of a node is not defined in your XML schema, you can configure an XML direct mapping to use the `xsi:type` attribute to provide type information.

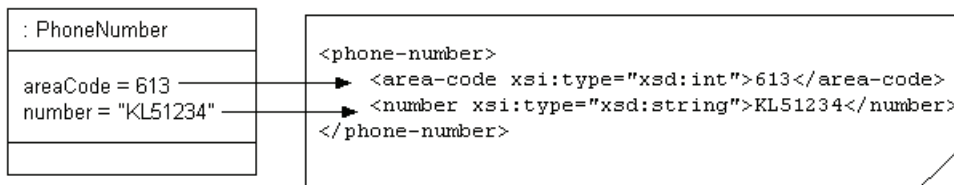
Given the XML schema fragment in [Example 53–22](#), [Figure 53–13](#) illustrates a Java class that can be mapped to a corresponding XML document.

**Example 53–22 Schema for XML Direct Mapping with Simple Type Translator**

```
...
<xs:element name="area-code" type="anySimpleType"/>
<xs:element name="number" type="anySimpleType"/>
...
```

**Figure 53–13 Java Class for XML Direct Mapping with Simple Type Translator**

[Figure 53–14](#) illustrates an XML direct mapping with a simple type translator in an XML document that conforms to the schema in [Example 53–22](#).

**Figure 53–14 XML Direct Mapping with a Simple Type Translator**

[Example 53–23](#) shows how to configure this mapping in Java.

**Example 53–23 Java for XML Direct Mapping with Simple Type Translator**

```
XMLDirectMapping numberMapping = new XMLDirectMapping();
numberMapping.setAttributeName("number");
numberMapping.setXPath("number/text()");
XMLField numberField = (XMLField) numberMapping.getField();
numberField.setIsTypedTextField(true);
```

For more information, see [Section 17.2.6.4, "Simple Type Translator"](#).

## 53.4 XML Composite Direct Collection Mapping

XML composite direct collection mappings map a Java collection of simple object attributes to XML attributes and text nodes. Use multiplicity settings to specify an element as a collection. The XML schema allows you to define minimum and maximum occurrences. You can use a composite direct collection XML mapping in the following scenarios:

- [Mapping to Multiple Text Nodes](#)
- [Mapping to Multiple Attributes](#)
- [Mapping to a Single Text Node with an XML Composite Direct Collection Mapping](#)
- [Mapping to a Single Attribute with an XML Composite Direct Collection Mapping](#)
- [Mapping to a List of Unions with an XML Composite Direct Collection Mapping](#)
- [Mapping to a Union of Lists with an XML Composite Direct Collection Mapping](#)
- [Specifying the Content Type of a Collection with an XML Composite Direct Collection Mapping](#)

See [Chapter 56, "Configuring an XML Composite Direct Collection Mapping"](#) for more information.

### 53.4.1 Mapping to Multiple Text Nodes

This section describes using a composite direct collection XML mapping when doing the following:

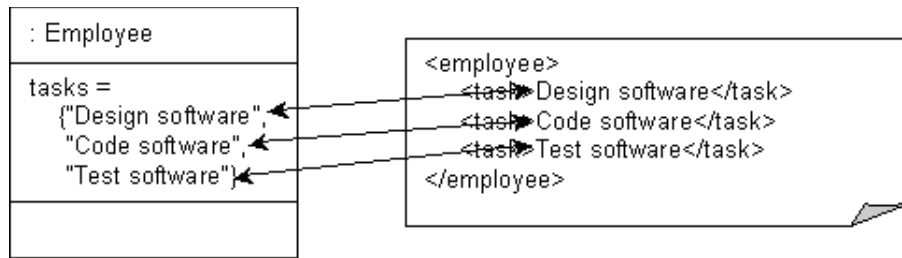
- [Mapping to a Simple Sequence](#)
- [Mapping to a Sequence in a Subelement](#)

#### 53.4.1.1 Mapping to a Simple Sequence

Given the XML schema in [Example 53–24](#), [Figure 53–15](#) illustrates a composite direct collection XML mapping to a simple sequence of text nodes in a corresponding XML document. [Example 53–25](#) shows how to configure this mapping in Java.

**Example 53–24 Schema for Composite Direct Collection XML Mapping to a Simple Sequence**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="employee" type="employee-type"/>
  <xsd:complexType name="employee-type">
    <xsd:sequence>
      <xsd:element name="task" type="xsd:string" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

**Figure 53–15 Composite Direct Collection XML Mapping to a Simple Sequence****Example 53–25 Java for Composite Direct Collection XML Mapping to a Simple Sequence**

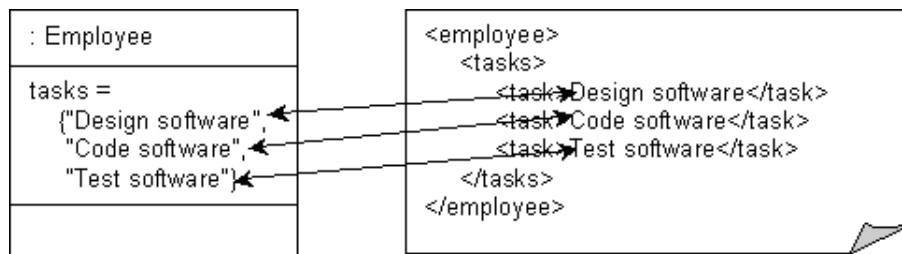
```
XMLCompositeDirectCollectionMapping tasksMapping = new XMLCompositeDirectCollectionMapping();
tasksMapping.setAttributeName("tasks");
tasksMapping.setXPath("task/text()");
```

### 53.4.1.2 Mapping to a Sequence in a Subelement

Given the XML schema in [Example 53–26](#), [Figure 53–16](#) illustrates a composite direct collection XML mapping to a sequence of text nodes in a subelement in a corresponding XML document. [Example 53–27](#) shows how to configure this mapping in Java.

**Example 53–26 Schema for Composite Direct Collection XML Mapping to a Subelement Sequence**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="employee" type="employee-type"/>
  <xsd:complexType name="employee-type">
    <xsd:sequence>
      <xsd:element name="tasks">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="task" type="xsd:string" maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

**Figure 53–16 Composite Direct Collection XML Mapping to a Subelement Sequence****Example 53–27 Java for Composite Direct Collection XML Mapping to a Subelement Sequence**

```
XMLCompositeDirectCollectionMapping tasksMapping = new XMLCompositeDirectCollectionMapping();
tasksMapping.setAttributeName("tasks");
tasksMapping.setXPath("tasks/task/text()");
```



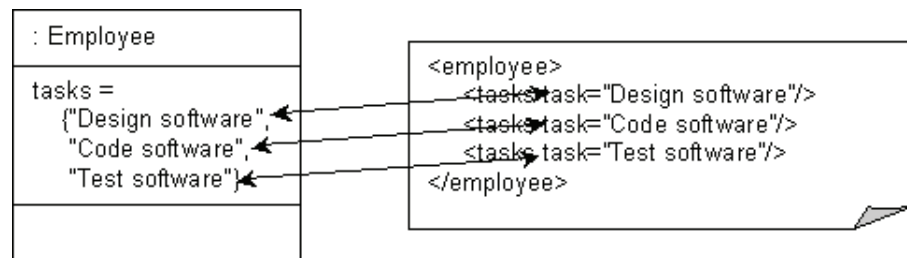
## 53.4.2 Mapping to Multiple Attributes

Given the XML schema in [Example 53–28](#), [Figure 53–17](#) illustrates a composite direct collection XML mapping to a sequence of text nodes in a subelement in a corresponding XML document. [Example 53–29](#) shows how to configure this mapping in Java.

### **Example 53–28 Schema for Composite Direct Collection XML Mapping to Multiple Attributes**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="employee" type="employee-type"/>
  <xsd:complexType name="employee-type">
    <xsd:sequence>
      <xsd:element name="tasks" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:attribute name="task" type="xsd:string"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

**Figure 53–17 Composite Direct Collection XML Mapping to Multiple Attributes**



### **Example 53–29 Java for Composite Direct Collection XML Mapping to Multiple Attributes**

```
XMLCompositeDirectCollectionMapping tasksMapping = new XMLCompositeDirectCollectionMapping();
tasksMapping.setAttributeName("tasks/@task");
tasksMapping.setXPath("task/text()");
```

## 53.4.3 Mapping to a Single Text Node with an XML Composite Direct Collection Mapping

When you map a collection to a single node, the contents of the node is treated as a space-separated list.

Given the XML schema in [Example 53–30](#), [Figure 53–18](#) illustrates a composite direct collection XML mapping to a single text node in a corresponding XML document. [Example 53–31](#) shows how to configure this mapping in Java.

### **Example 53–30 Schema for XML Composite Direct Collection Mapping to a Single Text Node**

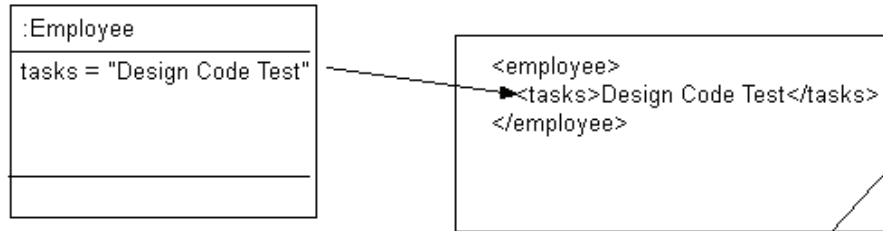
```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="employee" type="employee-type"/>
  <xsd:complexType name="employee-type">
    <xsd:sequence>
      <xsd:element name="tasks" type="tasks-type"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

```

</xsd:complexType>
<xsd:simpleType name="tasks-type">
  <xsd:list itemType="xsd:string"/>
</xsd:simpleType>
</xsd:schema>

```

**Figure 53–18 XML Composite Direct Collection Mapping to a Single Text Node**



**Example 53–31 Java for XML Composite Direct Collection Mapping to a Single Text Node**

```

XMLCompositeDirectCollectionMapping tasksMapping = new XMLCompositeDirectCollectionMapping();
tasksMapping.setAttributeName("tasks");
tasksMapping.setXPath("tasks/text()");
tasksMapping.setUsesSingleNode(true);

```

#### 53.4.4 Mapping to a Single Attribute with an XML Composite Direct Collection Mapping

Given the XML schema in [Example 53–32](#), [Figure 53–19](#) illustrates a composite direct collection XML mapping to a single attribute in a corresponding XML document. [Example 53–33](#) shows how to configure this mapping in Java.

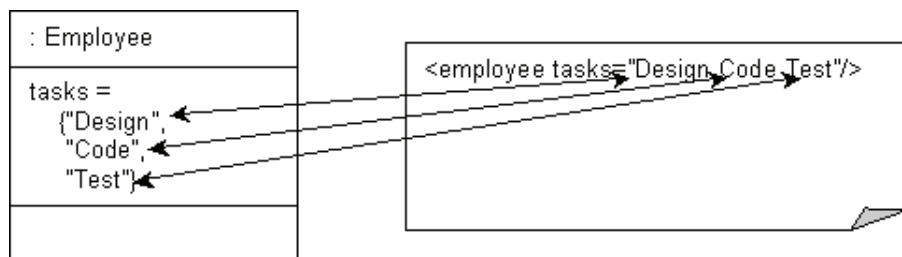
**Example 53–32 Schema for XML Composite Direct Collection Mapping to a Single Attribute**

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="employee" type="employee-type"/>
  <xsd:complexType name="employee-type">
    <xsd:attribute name="tasks" type="tasks-type"/>
  </xsd:complexType>
  <xsd:simpleType name="tasks-type">
    <xsd:list itemType="xsd:string"/>
  </xsd:simpleType>
</xsd:schema>

```

**Figure 53–19 XML Composite Direct Collection Mapping to a Single Attribute**



**Example 53–33 Java for XML Composite Direct Collection Mapping to a Single Attribute**

```

XMLCompositeDirectCollectionMapping tasksMapping = new XMLCompositeDirectCollectionMapping();
tasksMapping.setAttributeName("tasks");

```

```
tasksMapping.setXPath("@tasks");
tasksMapping.setUsesSingleNode(true);
```

### 53.4.5 Mapping to a List of Unions with an XML Composite Direct Collection Mapping

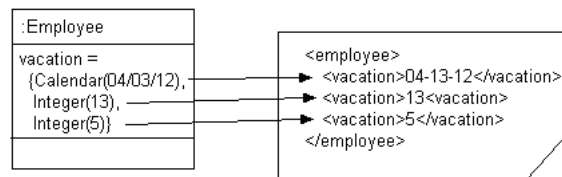
Given the XML schema in [Example 53–34](#), [Figure 53–20](#) illustrates a composite direct collection XML mapping to a list of unions in a corresponding XML document.

[Example 53–35](#) shows how to configure this mapping in Java.

#### **Example 53–34 Schema for XML Composite Direct Collection Mapping to List of Unions**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="vacation" type="listOfUnions"/>
  <xsd:simpleType name="listOfUnions">
    <xsd:list>
      <xsd:simpleType>
        <xsd:union memberTypes="xsd:date xsd:integer"/>
      </xsd:simpleType>
    </xsd:list>
  </xsd:simpleType>
</xsd:schema>
```

**Figure 53–20 Composite XML Direct Collection Mapping to List of Unions**



#### **Example 53–35 Java for XML Composite Direct Collection Mapping to List of Unions**

```
XMLCompositeDirectCollectionMapping mapping = new XMLCompositeDirectCollectionMapping();
mapping.setAttributeName("myattribute");
XMLUnionField field = new XMLUnionField("listOfUnions/text()");
mapping.addSchemaType(new QName(url, "int"));
mapping.addSchemaType(new QName(url, "date"));
mapping.setField(field);
mapping.useSingleElement(false);
```

### 53.4.6 Mapping to a Union of Lists with an XML Composite Direct Collection Mapping

Given the XML schema in [Example 53–36](#), [Figure 53–21](#) illustrates an XML composite direct collection mapping to a union of lists in a corresponding XML document.

[Example 53–37](#) shows how to configure this mapping in Java.

#### **Example 53–36 Schema for XML Composite Direct Collection Mapping to a Union of Lists**

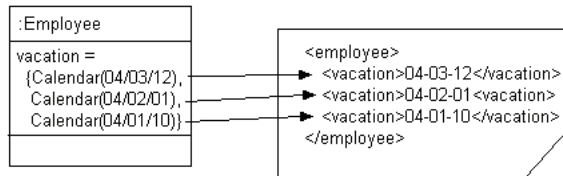
```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="vacation" type="unionOfLists"/>
  <xsd:simpleType name="unionOfLists">
    <xsd:union memberTypes="xsd:double">
      <xsd:simpleType>
        <xsd:list itemType="xsd:date"/>
      </xsd:simpleType>
      <xsd:simpleType>
        <xsd:list itemType="xsd:integer"/>
      </xsd:simpleType>
    </xsd:union>
  </xsd:simpleType>
</xsd:schema>
```

```

    </xsd:union>
  </xsd:simpleType>
</xsd:schema>

```

**Figure 53–21 XML Composite Direct Collection Mapping to a Union of Lists**



Note that in this example, valid XML documents contain either all `xsd:double`, all `xsd:date`, or all `xsd:integer` values.

**Example 53–37 Java for XML Composite Direct Collection Mapping to a Union of Lists**

```

XMLCompositeDirectCollectionMapping mapping = new XMLCompositeDirectCollectionMapping();
mapping.setAttributeName("myAttribute");
mapping.useSingleElement(false);
XMLUnionField unionField = new XMLUnionField("UnionOfLists/text()");
field.addSchemaType(new QName(url, "integer"));
field.addSchemaType(new QName(url, "date"));
field.addSchemaType(new QName(url, "double"));
field.setUsesSingleNode(false);

```

### 53.4.7 Specifying the Content Type of a Collection with an XML Composite Direct Collection Mapping

By default, TopLink will treat the node values read by a composite direct collection XML mapping as objects of type `String`. You can override this behavior by specifying the type of the collection's contents.

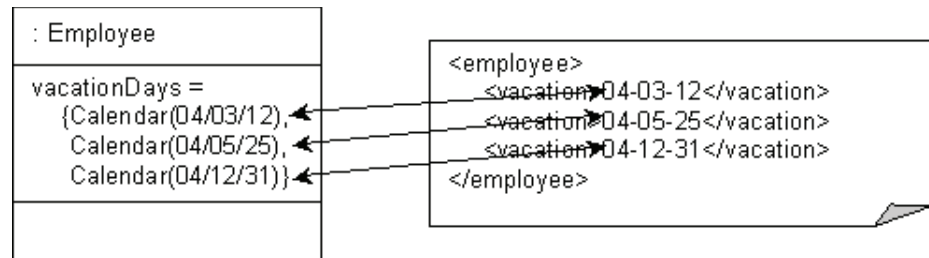
Given the XML schema in [Example 53–38](#), [Figure 53–22](#) illustrates an XML composite direct collection mapping to a simple sequence in a corresponding XML document. The mapping is configured to specify the content type of the collection as `Calendar`. [Example 53–39](#) shows how to configure this mapping in Java.

**Example 53–38 Schema for XML Composite Direct Collection Mapping with Specified Content Type**

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="employee" type="employee-type" />
  <xsd:complexType name="employee-type">
    <xsd:sequence>
      <xsd:element name="vacation" type="xsd:string" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>

```

**Figure 53–22 XML Composite Direct Collection Mapping with Specified Content Type****Example 53–39 Java for XML Composite Direct Collection Mapping with Specified Content Type**

```
XMLCompositeDirectCollectionMapping tasksMapping = new XMLCompositeDirectCollectionMapping();
tasksMapping.setAttributeName("vacationDays");
tasksMapping.setXPath("vacation/text()");
tasksMapping.setAttributeElementClass(Calendar.class);
```

## 53.5 XML Composite Object Mapping

XML composite object mappings represent a relationship between two classes. In XML, the "owned" class may be nested with the element tag representing the "owning" class. You can use a composite object XML mapping in the following scenarios:

- Mapping into the Parent Record
- Mapping to an Element
- Mapping to Different Elements by Element Name
- Mapping to Different Elements by Element Position

See [Chapter 57, "Configuring an XML Composite Object Mapping"](#) for more information.

### 53.5.1 Mapping into the Parent Record

The composite object may be mapped to the same record as the parent.

---

**Note:** The nodes mapped to by the composite object must be sequential.

---

Given the XML schema in [Example 53–40](#), [Figure 53–23](#) illustrates an XML composite object mapping into the parent record in a corresponding XML document.

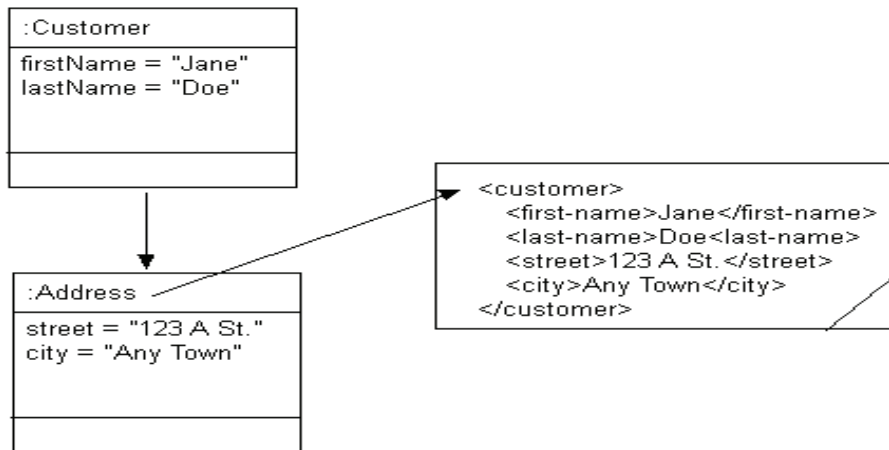
[Example 53–41](#) shows how to configure this mapping in Java.

**Example 53–40 Schema for XML Composite Object Mapping into the Parent Record**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="customer" type="customer-type"/>
  <xsd:complexType name="customer-type">
    <xsd:sequence>
      <xsd:element name="first-name" type="xsd:string"/>
      <xsd:element name="last-name" type="xsd:string"/>
      <xsd:element name="street" type="xsd:string"/>
      <xsd:element name="city" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
```

```
</xsd:schema>
```

**Figure 53–23 XML Composite Object Mapping into the Parent Record**



**Example 53–41 Java for XML Composite Object Mapping into the Parent Record**

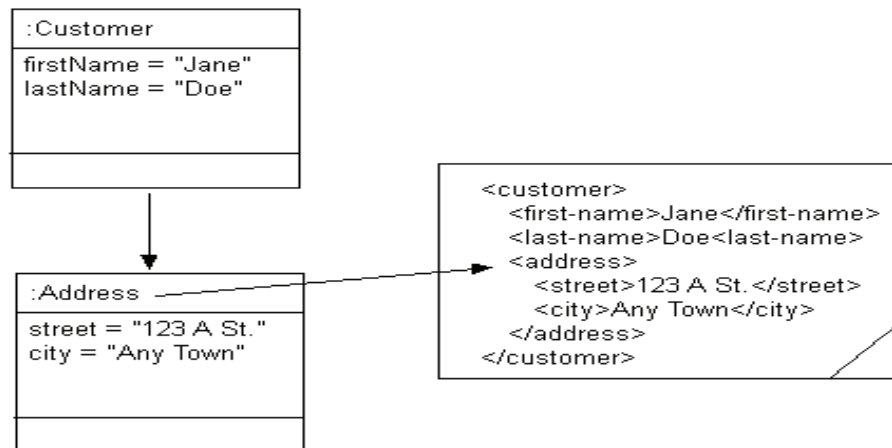
```
XMLCompositeObjectMapping addressMapping = new XMLCompositeObjectMapping();
addressMapping.setAttributeName("address");
addressMapping.setXPath(".");
addressMapping.setReferenceClass(Address.class);
```

## 53.5.2 Mapping to an Element

Given the XML schema in [Example 53–42](#), [Figure 53–24](#) illustrates an XML composite object mapping to an element in a corresponding XML document. [Example 53–43](#) shows how to configure this mapping in Java.

**Example 53–42 Schema for XML Composite Object Mapping to an Element**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="customer" type="customer-type"/>
  <xsd:complexType name="customer-type">
    <xsd:sequence>
      <xsd:element name="first-name" type="xsd:string"/>
      <xsd:element name="last-name" type="xsd:string"/>
      <xsd:element name="address">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="street" type="xsd:string"/>
            <xsd:element name="city" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

**Figure 53–24 XML Composite Object Mapping to an Element****Example 53–43 Java for XML Composite Object Mapping to an Element**

```
XMLCompositeObjectMapping addressMapping = new XMLCompositeObjectMapping();
addressMapping.setAttributeName("address");
addressMapping.setXPath("address");
addressMapping.setReferenceClass(Address.class);
```

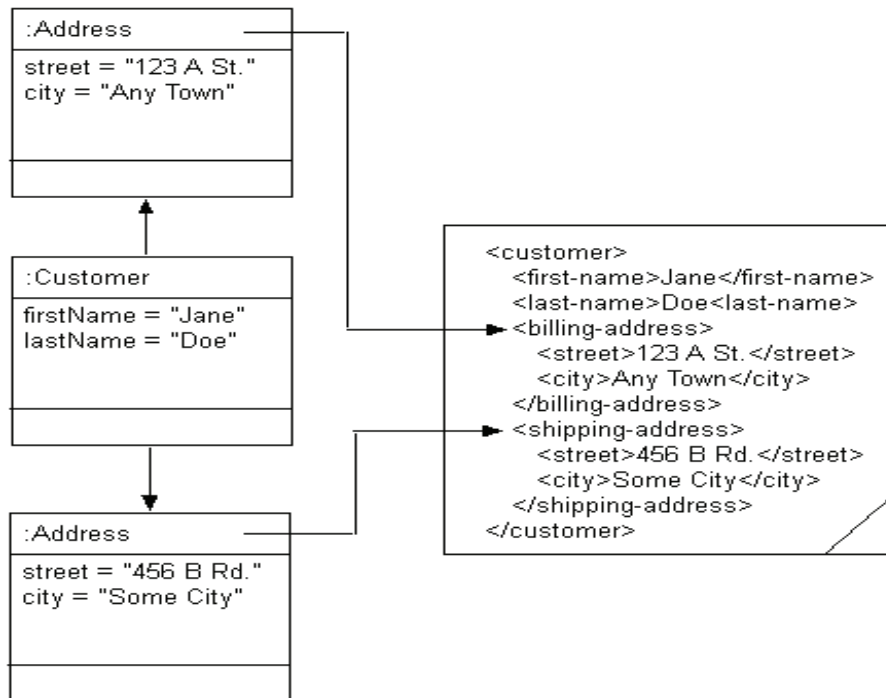
### 53.5.3 Mapping to Different Elements by Element Name

An object may have multiple composite object mappings to the same reference class. Each composite object mapping must have a unique XPath. This example uses unique XPaths by name.

Given the XML schema in [Example 53–44](#), [Figure 53–25](#) illustrates an XML composite object mapping to different elements by name in a corresponding XML document. [Example 53–45](#) shows how to configure this mapping in Java.

**Example 53–44 Schema for XML Composite Object Mapping to Elements by Name**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="customer" type="customer-type"/>
  <xsd:complexType name="customer-type">
    <xsd:sequence>
      <xsd:element name="first-name" type="xsd:string"/>
      <xsd:element name="last-name" type="xsd:string"/>
      <xsd:element name="billing-address" type="address-type"/>
      <xsd:element name="shipping-address" type="address-type"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="address-type">
    <xsd:sequence>
      <xsd:element name="street" type="xsd:string"/>
      <xsd:element name="city" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

**Figure 53–25 XML Composite Object Mapping to Elements by Name****Example 53–45 Java for XML Composite Object Mapping to Elements by Name**

```
XMLCompositeObjectMapping billingAddressMapping = new XMLCompositeObjectMapping();
billingAddressMapping.setAttributeName("billingAddress");
billingAddressMapping.setXPath("billing-address");
billingAddressMapping.setReferenceClass(Address.class);
```

```
XMLCompositeObjectMapping shippingAddressMapping = new XMLCompositeObjectMapping();
shippingAddressMapping.setAttributeName("shippingAddress");
shippingAddressMapping.setXPath("shipping-address");
shippingAddressMapping.setReferenceClass(Address.class);
```

### 53.5.4 Mapping to Different Elements by Element Position

An object may have multiple composite object mappings to the same reference class. Each composite object mapping must have a unique XPath. This example uses unique XPaths by position.

Given the XML schema in [Example 53–46](#), [Figure 53–26](#) illustrates an XML composite object mapping to different elements by position in a corresponding XML document. [Example 53–47](#) shows how to configure this mapping in Java.

**Example 53–46 Schema for XML Composite Object Mapping to Elements by Position**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="customer" type="customer-type"/>
  <xsd:complexType name="customer-type">
    <xsd:sequence>
      <xsd:element name="first-name" type="xsd:string"/>
      <xsd:element name="last-name" type="xsd:string"/>
      <xsd:element name="address" maxOccurs="2">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="street" type="xsd:string"/>

```

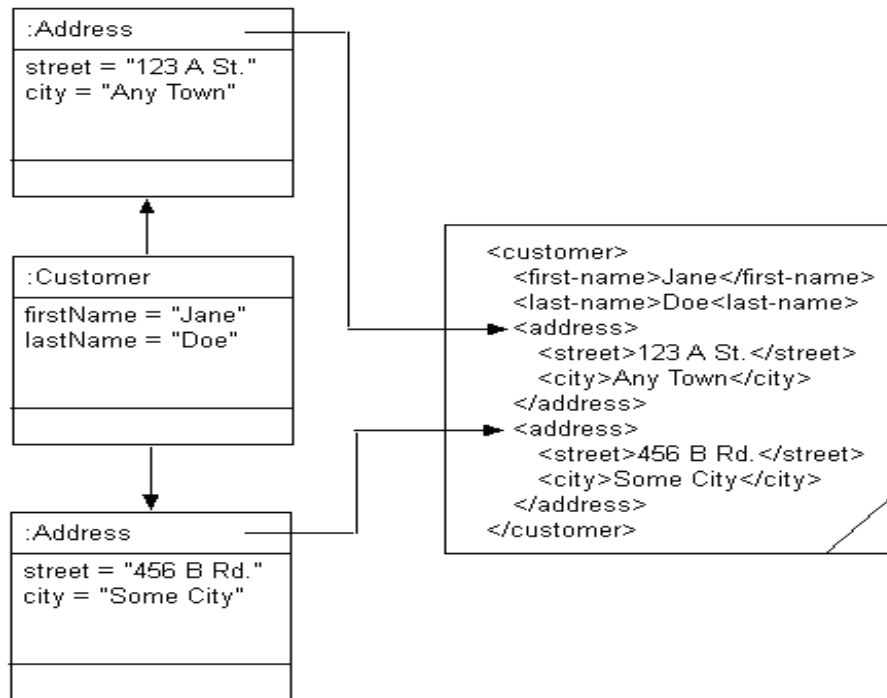


```

        <xsd:element name="city" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>

```

**Figure 53–26 XML Composite Object Mapping to Elements by Position**



**Example 53–47 Java for XML Composite Object Mapping to Elements by Position**

```

XMLCompositeObjectMapping billingAddressMapping = new XMLCompositeObjectMapping();
billinAddressMapping.setAttributeName("billingAddress");
billinAddressMapping.setXPath("address[1]");
billinAddressMapping.setReferenceClass(Address.class);

```

```

XMLCompositeObjectMapping shippingAddressMapping = new XMLCompositeObjectMapping();
shippingAddressMapping.setAttributeName("shippingAddress");
shippingAddressMapping.setXPath("address[2]");
shippingAddressMapping.setReferenceClass(Address.class);

```

## 53.6 XML Composite Collection Mapping

Use XML composite collection mappings to represent one-to-many relationships. Composite collection XML mappings can reference any class that has a `TopLink` descriptor. The attribute in the object mapped must implement either the Java Collection interface (for example, `Vector` or `HashSet`) or `Map` interface (for example, `Hashtable` or `TreeMap`). The `XMLCompositeCollectionMapping` class allows a reference to the mapped class and the indexing type for that class.

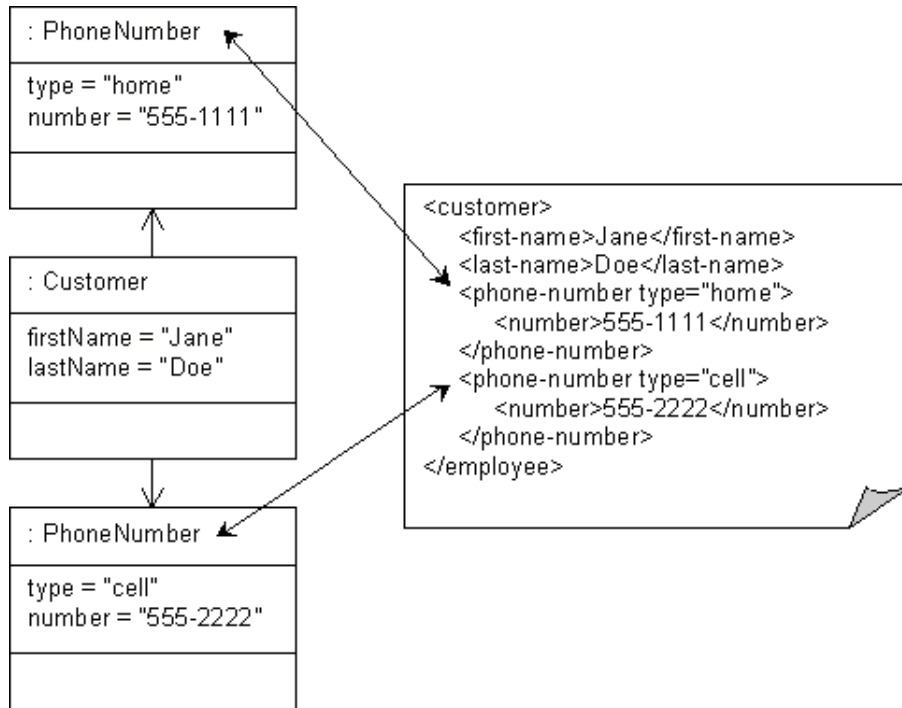
Given the XML schema in [Example 53–48](#), [Figure 53–27](#) illustrates an XML composite collection mapping to different elements by position in a corresponding XML document. [Example 53–49](#) shows how to configure this mapping in Java for a

Collection attribute, and [Example 53–50](#) shows how to configure this mapping in Java for a Map attribute.

**Example 53–48 Schema for XML Composite Collection Mapping**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="customer" type="customer-type"/>
  <xsd:complexType name="customer-type">
    <xsd:sequence>
      <xsd:element name="first-name" type="xsd:string"/>
      <xsd:element name="last-name" type="xsd:string"/>
      <xsd:element name="phone-number">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="number" type="xsd:string"/>
          </xsd:sequence>
          <xsd:attribute name="type" type="xsd:string"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

**Figure 53–27 XML Composite Collection Mapping**



**Example 53–49 Java for XML Composite Collection Mapping for a Collection Attribute**

```
XMLCompositeCollectionMapping phoneNumbersMapping = new XMLCompositeCollectionMapping();
phoneNumbersMapping.setAttributeName("phoneNumbers");
phoneNumbersMapping.setXPath("phone-number");
phoneNumbersMapping.setReferenceClass(PhoneNumber.class);
```

**Example 53–50 Java for XML Composite Collection Mapping for a Map Attribute**

```
XMLCompositeCollectionMapping phoneNumbersMapping = new XMLCompositeCollectionMapping();
phoneNumbersMapping.setAttributeName("phoneNumbers");
```

```
phoneNumbersMapping.setXPath("phone-number");
phoneNumbersMapping.setReferenceClass(PhoneNumber.class);
phoneNumbersMapping.useMapClass(HashMap.class, "getType");
```

See [Chapter 58, "Configuring an XML Composite Collection Mapping"](#) for more information.

## 53.7 XML Any Object Mapping

The XML any object mapping is similar to the composite object XML mapping (see [Section 53.5, "XML Composite Object Mapping"](#)) except that the reference object may be of any type (including `String`). This type does not need to be related to any other particular type through inheritance or a common interface.

The corresponding object attribute value can be an instance of any object with a `Descriptor`, a `java.lang.Object`, a `java.lang.String`, a primitive object (such as `java.lang.Integer`), or a user defined type generic enough for all possible application values.

This mapping is useful with the following XML schema constructs:

- any
- choice
- substitution groups

Referenced objects can specify a default root element on their descriptor (see [Section 16.2.12, "Default Root Element"](#)).

---



---

**Note:** The undefined document root element of a referenced object is ignored during marshalling with an any collection mapping and object mapping.

---



---

Given the XML schema in [Example 53–51, Figure 53–28](#) illustrates the Java classes used in this example. A single XML any object mapping is used to map `Customer` attribute `contactMethod`. This attribute must be generic enough to reference all possible values: in this example, instances of `Address`, `PhoneNumber`, and `String`.

### **Example 53–51** Schema for XML Any Object Mapping

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="customer" type="customer-type"/>
  <xsd:complexType name="customer-type">
    <xsd:sequence>
      <xsd:element name="contact-method" type="xsd:anyType"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="address">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="street" type="xsd:string"/>
        <xsd:element name="city" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="phone-number" type="xsd:string"/>
</xsd:schema>
```

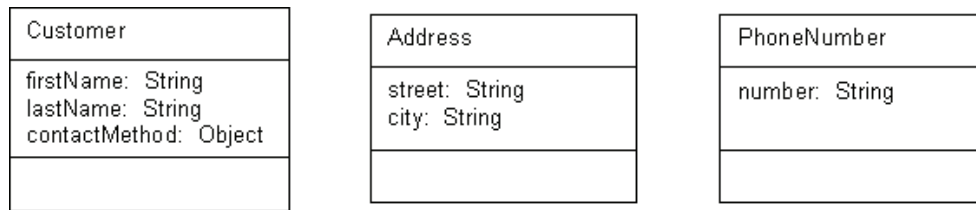
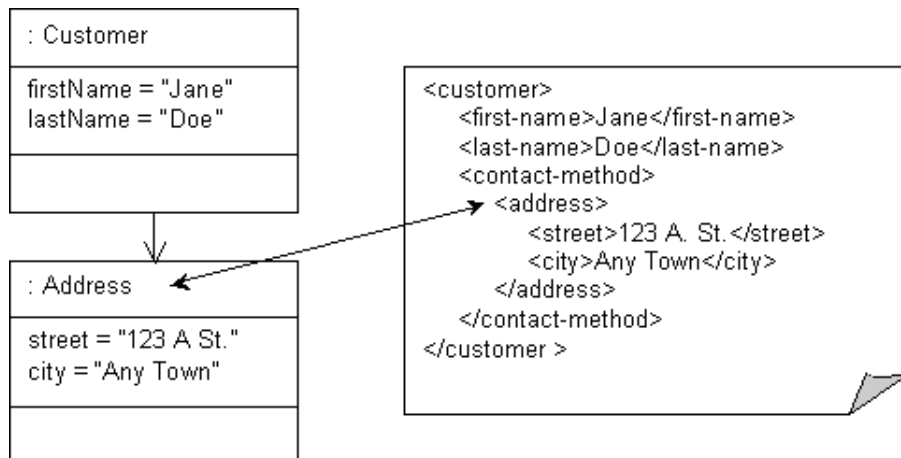
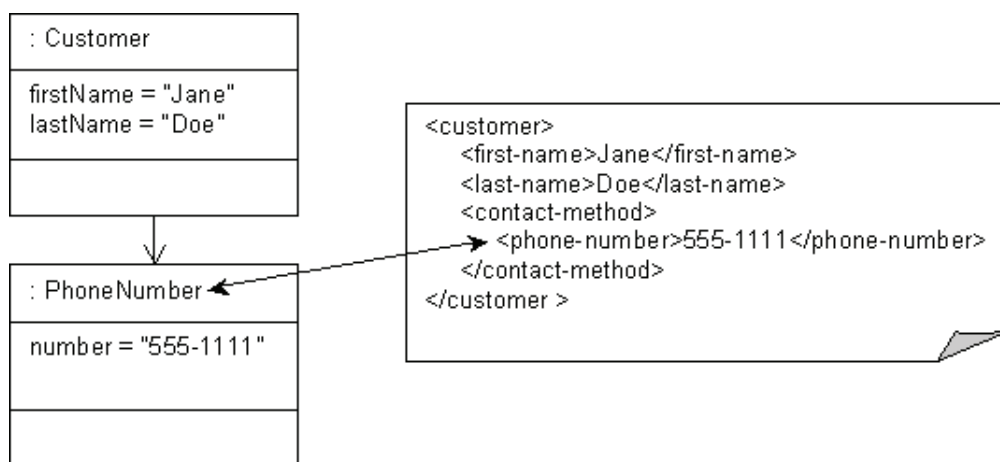
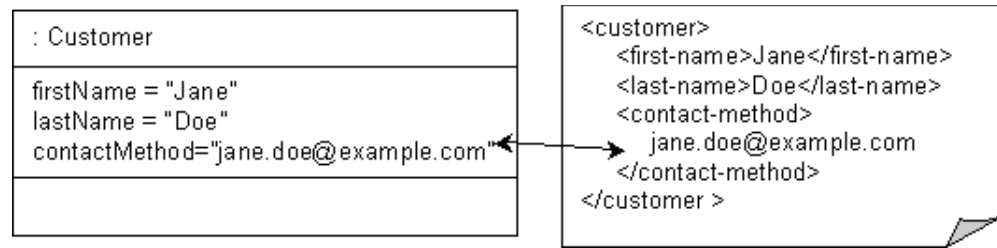
**Figure 53–28 Java Classes for XML Any Object Mapping**

Figure 53–29, Figure 53–30, and Figure 53–31 illustrate how the XML any object mapping maps to an Address, PhoneNumber, and String (respectively) in XML documents that conform to the schema in Example 53–51.

**Figure 53–29 XML Any Object Mapping to Address Type****Figure 53–30 XML Any Object Mapping to PhoneNumber Type**

**Figure 53–31 XML Any Object Mapping to String Type**

Example 53–52 shows how to configure this mapping in Java.

**Example 53–52 Java for XML Any Object Mapping**

```

XMLAnyObjectMapping contactMethodMapping = new XMLAnyObjectMapping();
contactMethodMapping.setAttributeName("contactMethod");
contactMethodMapping.setXPath("contact-method");

```

For more information about TopLink XML mapping support for `xs:any` and `xs:anyType`, see [Section 53.2.5, "xs:any and xs:anyType Support"](#).

See [Chapter 59, "Configuring an XML Any Object Mapping"](#) for more information.

## 53.8 XML Any Collection Mapping

The XML any collection mapping is similar to the composite collection XML mapping (see [Section 53.6, "XML Composite Collection Mapping"](#)), except that the referenced objects may be of different types (including `String`). These types need not be related to each other through inheritance or a common interface.

The corresponding object attribute value can be an instance of any object with a `Descriptor`, a `java.lang.Object`, a `java.lang.String`, a primitive object (such as `java.lang.Integer`), or a user-defined type generic enough for all possible application values.

This mapping is useful with the following XML schema constructs:

- any
- choice
- substitution groups

Each of the referenced objects (except `String`) must specify a default root element on their descriptor (see [Section 16.2.12, "Default Root Element"](#)).

Given the XML schema in [Example 53–53](#), [Figure 53–32](#) illustrates the Java classes used in this example. A single XML any collection mapping is used to map `Customer` attribute `contactMethods`. This attribute must be generic enough to reference all possible values: in this example, instances of `Address`, `PhoneNumber`, and `String`.

**Example 53–53 Schema for XML Any Collection Mapping**

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="customer" type="customer-type"/>
  <xsd:complexType name="customer-type">
    <xsd:sequence>
      <xsd:element name="contact-methods" type="xsd:anyType"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="address">
    <xsd:complexType>

```

```

<xsd:sequence>
  <xsd:element name="street" type="xsd:string"/>
  <xsd:element name="city" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="phone-number" type="xsd:string"/>
</xsd:schema>

```

**Figure 53–32 Java Classes for XML Any Collection Mapping**

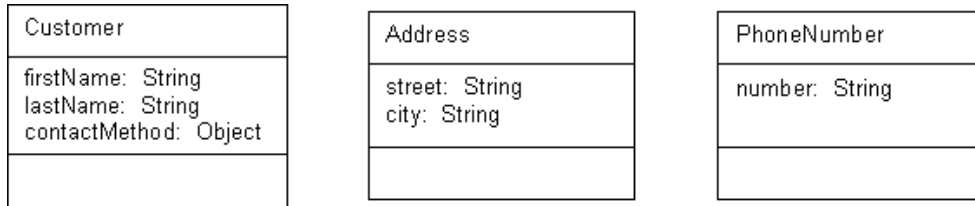
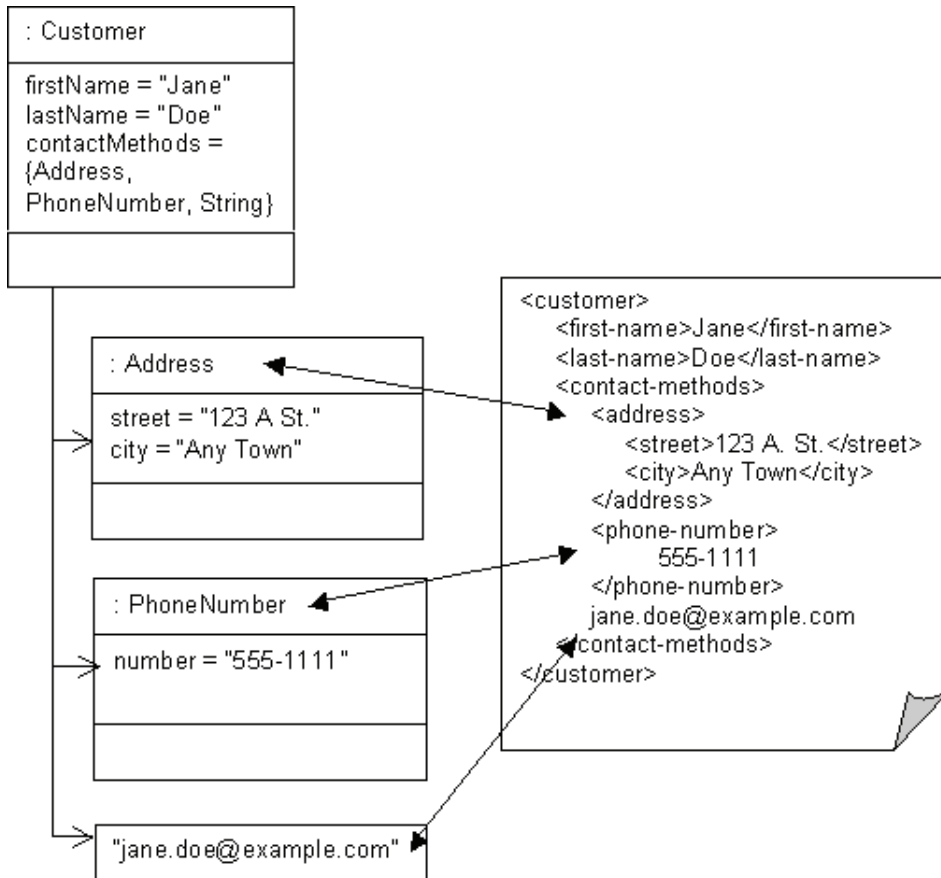


Figure 53–33 illustrate how the XML any collection mapping maps to a collection of Address, PhoneNumber, and String objects in an XML document that conforms to the schema in Example 53–53.

**Figure 53–33 XML Any Collection Mapping**



Example 53–54 shows how to configure this mapping in Java.

**Example 53–54 Java for XML Any Collection Mapping**

```
XMLAnyCollectionMapping contactMethodsMapping = new XMLAnyCollectionMapping();
contactMethodsMapping.setAttributeName("contactMethods");
contactMethodsMapping.setXPath("contact-methods");
```

For more information about TopLink XML mapping support for `xs:any` and `xs:anyType`, see [Section 53.2.5, "xs:any and xs:anyType Support"](#).

See [Chapter 60, "Configuring an XML Any Collection Mapping"](#) for more information.

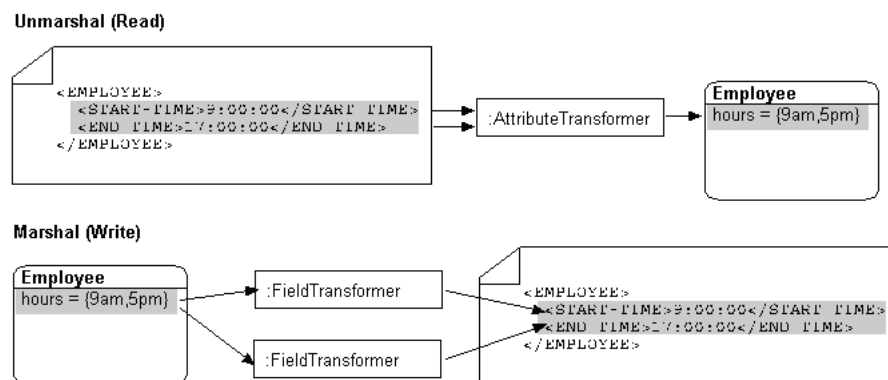
## 53.9 XML Transformation Mapping

You can use an XML transformation mapping to create a custom mapping where one or more XML nodes can be used to create the object to be stored in a Java class's attribute. To handle the custom requirements at marshal (write) and unmarshal (read) time, a transformation mapping takes instances of `oracle.toplink.mappings.transformers` (such as `AttributeTransformer` and `FieldTransformer`) that you provide. This provides a nonintrusive solution that avoids the need for your domain objects to implement special interfaces for this purpose.

As [Figure 53–34](#) illustrates, you configure the transformation mapping with an `oracle.toplink.mappings.transformers.AttributeTransformer` instance to perform the XML instance-to-Java attribute transformation at unmarshal time. In this example, the `AttributeTransformer` combines two XML text nodes into a single Java object.

Similarly, you also configure the transformation mapping with one or more `oracle.toplink.mappings.transformers.FieldTransformer` instances to perform the Java attribute-to-XML instance transformation at marshal time. In this example, each `FieldTransformer` is responsible for mapping one of the Java object values to an XML text node.

**Figure 53–34 XML Transformation Mappings**



See [Chapter 61, "Configuring an XML Transformation Mapping"](#) for more information.

## 53.10 XML Object Reference Mapping

The `oracle.toplink.ox.mappings.XMLObjectReferenceMapping` is a key on source-based aggregate mapping. It allows you to use one-to-one mappings to map a given element in an XML document to another element in that same XML document using one or more keys.

Use this mapping when several objects reference the same instance of another object. In this case, one and only one of the mappings is to be a composite—the remaining mappings must be reference mappings. These references will be created based on one or more key values.

XML object reference mapping is fully supported in the deployment XML.

With this mapping, TopLink provides support for composite keys, as well as for foreign key grouping elements.

---

**Note:** You should group together elements mapped to keys. Also, TopLink supports grouping elements that wrap all of the keys (not the ones that wrap each individual key).

---

The `XMLObjectReferenceMapping` captures the following information:

- Attribute name.
- Reference class.
- Map of source and target key pairs, such as XPath values (see [Section 53.2.3, "XPath Support"](#)) in the following format:

```
[ "project-id/text()", "@id" ]
```

Use the `addSourceToTargetKeyFieldAssociation` method to add a source and target XPath pair to the map.

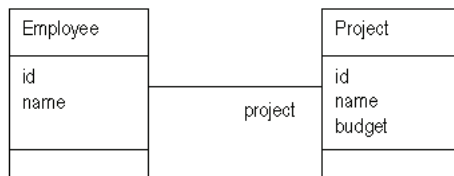
- List of source keys to maintain order.

See [Chapter 62, "Configuring an XML Object Reference Mapping"](#) for more information.

### 53.10.1 Mapping Using a Single Key

[Figure 53–35](#), [Example 53–55](#), [Example 53–56](#) and [Example 53–57](#) demonstrate how to map one element to another using a single key.

**Figure 53–35 Class Diagram**



**Example 53–55 Using Single Key - Instance Document**

```

...
<employee id="12">
  <name>Joe Brown</name>
  <project-id>99</project-id>
</employee>
<project id="99">
  <name>Big Project</name>
  <budget>100,000</budget>
</project>
...
  
```



[Example 53–56](#) shows how to create an `XMLObjectReferenceMapping`, set a single key on source, and then add the mapping to the descriptor.

**Example 53–56 Using Single Key - Project Class - Employee Descriptor**

```
...
XMLObjectReferenceMapping emp = new XMLObjectReferenceMapping();
emp.setAttributeName("project");
emp.setReferenceClass(Project.class);
emp.addSourceToTargetKeyFieldAssociation("project-id/text()", "@id");
empDescriptor.addMapping(emp);
...
```

[Example 53–57](#) shows how to define the primary key field on the descriptor.

**Example 53–57 Using Single Key - Project Class - Project Descriptor**

```
...
XMLDescriptor prjDescriptor = new XMLDescriptor();
prjDescriptor.setJavaClass(Project.class);
prjDescriptor.addPrimaryKeyField("@id");
...
```

## 53.10.2 Mapping Using a Composite Key

[Figure 53–35](#), [Example 53–58](#), [Example 53–59](#) and [Example 53–60](#) demonstrate how to map one element to another using a composite key.

**Example 53–58 Using Composite Key - Instance Document**

```
...
<employee id="12">
  <name>Joe Brown</name>
  <prj-name>Big Project</prj-name>
  <prj-budget>100,000</prj-budget>
</employee>
<project id="99">
  <name>Big Project</name>
  <budget>100,000</budget>
</project>
...
```

[Example 53–59](#) shows how to create an `XMLObjectReferenceMapping`, set a composite key on source, and then add the mapping to the descriptor.

**Example 53–59 Using Composite Key - Project Class - Employee Descriptor**

```
...
XMLObjectReferenceMapping emp = new XMLObjectReferenceMapping();
emp.setAttributeName("project");
emp.setReferenceClass(Project.class);
emp.addSourceToTargetKeyFieldAssociation("prj-name/text()", "name/text()");
emp.addSourceToTargetKeyFieldAssociation("prj-budget/text()", "budget/text()");
empDescriptor.addMapping(emp);
...
```

[Example 53–60](#) shows how to define a composite primary key field on the descriptor.

**Example 53–60 Using Composite Key - Project Class - Project Descriptor**

```
...
XMLDescriptor prjDescriptor = new XMLDescriptor();
prjDescriptor.setJavaClass(Project.class);
```

```
prjDescriptor.addPrimaryKeyField("name");
prjDescriptor.addPrimaryKeyField("budget");
...
```

### 53.10.3 Mapping Using JAXB

The JAXB generator generates a deployment descriptor based on annotations and default values.

---

---

**Note:** The JAXB specification states that for the @XmlID annotation, the following restrictions apply:

- The field type must be String.
  - The use of composite keys is not allowed.
- 
- 

Figure 53–35, Example 53–61, Example 53–62, Example 53–63 and Example 53–64 demonstrate how to map one XML element to another using JAXB annotations.

#### **Example 53–61 Using JAXB - Object Model**

```
public class Employee {

    @XmlAttribute(name="id")
    public String id;
    public String name;

    @XmlElement(name="project-id")
    @XmlIDREF
    public Project project;
    ...
}

public class Project {

    @XmlElement(name="project-id")
    @XmlID
    public String id;

    public String name;
    public String budget;
    ...
}
```

#### **Example 53–62 Using JAXB - Instance Document**

```
...
<employee id="12">
  <name>Joe Brown</name>
  <project-id>99</project-id>
</employee>
<project project-id="99">
  <name>Big Project</name>
  <budget>100,000</budget>
</project>
...
```

Example 53–63 shows how to create an XMLObjectReferenceMapping, set a key on source, and then add the mapping to the descriptor.

**Example 53–63 Using JAXB - Project Class - Employee Descriptor**

```

...
XMLObjectReferenceMapping emp = new XMLObjectReferenceMapping();
emp.setAttributeName("project");
emp.setReferenceClass(Project.class);
emp.addSourceToTargetKeyFieldAssociation("project-id/text()", "@pid");
empDescriptor.addMapping(emp);
...

```

Example 53–64 shows how to define the primary key field on the descriptor.

**Example 53–64 Using JAXB - Project Class - Project Descriptor**

```

...
XMLDescriptor prjDescriptor = new XMLDescriptor();
prjDescriptor.setJavaClass(Project.class);
prjDescriptor.addPrimaryKeyField("@pid");
...

```

For more information, see the following:

- JAXB 2.0 Specification at <http://jcp.org/aboutJava/communityprocess/pfd/jsr222/index.html>
- Chapter 47, "Introduction to XML Projects"

## 53.11 XML Collection Reference Mapping

The `oracle.toplink.ox.mappings.XMLCollectionReferenceMapping` is a key on source-based aggregate mapping. It allows you to use one-to-many mappings to map a given element in an XML document to another element in that same XML document using one or more keys.

With this mapping, TopLink provides support for foreign key grouping elements.

---

**Note:** You should group together elements mapped to keys. Also, TopLink supports grouping elements that wrap all of the keys (not the ones that wrap each individual key).

---

The `XMLCollectionReferenceMapping` captures the following information:

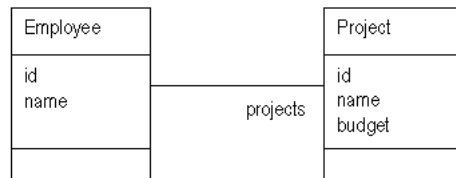
- Attribute name.
- Reference class.
- Map of source and target key pairs, such as XPath values (see Section 53.2.3, "XPath Support") in the following format:

```
["project-id/text()", "@id"]
```

Use the `addSourceToTargetKeyFieldAssociation` method to add a source and target XPath pair to the map.

- List of source keys to maintain order.

Figure 53–36, Example 53–65, Example 53–66 and Example 53–67 demonstrate how to map one element to another using a single key.

**Figure 53–36 Class Diagram****Example 53–65 Using a Single Key - Instance Document**

```

...
<employee id="12">
  <name>Joe Brown</name>
  <project-id>99</project-id>
  <project-id>199</project-id>
</employee>
<project id="99">
  <name>Big Project</name>
  <budget>100,000</budget>
</project>
<project id="199">
  <name>Bigger Project</name>
  <budget>100,000,000</budget>
</project>
...

```

[Example 53–66](#) shows how to create an `XMLCollectionReferenceMapping`, set a single key on source, and then add the mapping to the descriptor.

**Example 53–66 Using a Single Key - Project Class - Employee Descriptor**

```

...
XMLCollectionReferenceMapping prj = new XMLCollectionReferenceMapping();
prj.useCollectionClass(ArrayList.class);
prj.setAttributeName("projects");
prj.setReferenceClass(Project.class);
prj.addSourceToTargetKeyFieldAssociation("project-id/text()", "@id");
empDescriptor.addMapping(prj);
...

```

[Example 53–67](#) shows how to define the primary key field on the descriptor.

**Example 53–67 Using a Single Key - Project Class - Project Descriptor**

```

...
XMLDescriptor prjDescriptor = new XMLDescriptor();
prjDescriptor.setJavaClass(Project.class);
prjDescriptor.addPrimaryKeyField("@id");
...

```

[Figure 53–36](#), [Example 53–68](#), [Example 53–69](#) and [Example 53–70](#) demonstrate how to map one element to another using a single key as a space-separated list.

**Example 53–68 Using a Single Key as Space-Separated List - Instance Document**

```

...
<employee id="12" project-ids="99 199">
  <name>Joe Brown</name>
</employee>
<project id="99">
  <name>Big Project</name>

```

```

        <budget>100,000</budget>
    </project>
    <project id="199">
        <name>Bigger Project</name>
        <budget>100,000,000</budget>
    </project>
    ...

```

[Example 53–69](#) shows how to create an `XMLCollectionReferenceMapping`, set a single key on source as a space-separated list, and then add the mapping to the descriptor.

**Example 53–69 Using a Single Key as Space-Separated List - Project Class - Employee Descriptor**

```

...
XMLCollectionReferenceMapping prj = new XMLCollectionReferenceMapping();
prj.useCollectionClass(ArrayList.class);
prj.setAttributeName("projects");
prj.setReferenceClass(Project.class);
prj.addSourceToTargetKeyFieldAssociation("project-ids", "@id");
empDescriptor.addMapping(prj);
...

```

[Example 53–70](#) shows how to define the primary key field on the descriptor.

**Example 53–70 Using a Single Key as Space-Separated List - Project Class - Project Descriptor**

```

...
XMLDescriptor prjDescriptor = new XMLDescriptor();
prjDescriptor.setJavaClass(Project.class);
prjDescriptor.addPrimaryKeyField("@id");
...

```

See [Chapter 63, "Configuring an XML Collection Reference Mapping"](#) for more information.

## 53.12 XML Binary Data Mapping

The `oracle.toplink.ox.mappings.XMLBinaryDataMapping` is a direct mapping (see [Section 53.3, "XML Direct Mapping"](#)) that you use for handling binary data: it maps binary data in the object model to XML. This allows you to enable writing of binary data directly as inline binary data (base64 BLOB), or passing through as a MtOM or SwaRef attachment. For more information, see [Section 12.16, "Optimizing Storage and Retrieval of Binary Data in XML"](#).

The `XMLBinaryDataMapping` also lets you make callbacks to an attachment marshaller and unmarshaller (see [Section 12.16.1, "How to Use an Attachment Marshaller and Unmarshaller"](#)), as well as set XPath (see [Section 53.2.3, "XPath Support"](#)).

[Example 53–71](#) shows how to create an `XMLBinaryDataMapping`, set some of its properties, and then add the mapping to a descriptor.

**Example 53–71 Creating an XML Binary Data Mapping**

```

XMLBinaryDataMapping addressMapping = new XMLBinaryDataMapping();
addressMapping.setXPath("address");
addressMapping.setShouldInlineBinaryData(true);
descriptor.addMapping(addressMapping);
...

```

See [Chapter 64, "Configuring an XML Binary Data Mapping"](#) for more information.

## 53.13 XML Binary Data Collection Mapping

The `oracle.toplink.ox.mappings.XMLBinaryDataCollectionMapping` is very similar to the `XMLBinaryDataMapping` (see [Section 53.12, "XML Binary Data Mapping"](#)), except that it maps a collection of binary data in the object model to XML.

For more information, see the following:

- [Section 12.16, "Optimizing Storage and Retrieval of Binary Data in XML"](#)
- [Section 53.12, "XML Binary Data Mapping"](#)

[Example 53–72](#) shows how to create an `XMLBinaryDataCollectionMapping`, set some of its properties, and then add the mapping to a descriptor.

### **Example 53–72 Creating an XML Binary Data Collection Mapping**

```
XMLBinaryDataCollectionMapping addressMapping =
    new XMLBinaryDataCollectionMapping();
addressMapping.setCollectionContentType(type);
addressMapping.setXPath("address");
addressMapping.setShouldInlineBinaryData(true);
descriptor.addMapping(addressMapping);
...
```

For more information, see the following:

- [Chapter 65, "Configuring an XML Binary Data Collection Mapping"](#)
- [Section 53.4, "XML Composite Direct Collection Mapping"](#)

## 53.14 XML Fragment Mapping

The `oracle.toplink.ox.mappings.XMLFragmentMapping` is a direct mapping (see [Section 53.3, "XML Direct Mapping"](#)) that provides you a means to keep a part of an XML tree as a node.

The `XMLFragmentMapping` also lets you set the XPath (see [Section 53.2.3, "XPath Support"](#)).

[Example 53–73](#) shows how to create an `XMLFragmentMapping`, set some of its properties, and then add the mapping to a descriptor.

### **Example 53–73 Creating an XML Fragment Mapping**

```
XMLFragmentMapping addressMapping = new XMLFragmentMapping();
addressMapping.setXPath("address");
descriptor.addMapping(addressMapping);
...
```

See [Chapter 66, "Configuring an XML Fragment Mapping"](#) for more information.

## 53.15 XML Fragment Collection Mapping

The `oracle.toplink.ox.mappings.XMLFragmentCollectionMapping` is similar to the `XMLFragmentMapping` (see [Section 53.14, "XML Fragment Mapping"](#)), except that it allows you to keep a part of an XML tree as a collection of nodes.

[Example 53–74](#) shows how to create an `XMLFragmentMapping`, set the XPath (see [Section 53.2.3, "XPath Support"](#)), and then add the mapping to a descriptor.

**Example 53–74 Creating an XML Fragment Collection Mapping**

```
XMLFragmentCollectionMapping addressMapping =
    new XMLFragmentCollectionMapping();
addressMapping.setXPath("address");
descriptor.addMapping(addressMapping);
...
```

See [Chapter 67, "Configuring an XML Fragment Collection Mapping"](#) for more information.

## 53.16 XML Choice Object Mapping

The `oracle.toplink.ox.mappings.XMLChoiceObjectMapping` lets you map a single attribute to a number of different elements in an XML document.

Unlike other TopLink XML mappings, instead of setting a single XPath, you use the `addChoiceElement` method to specify an XPath as well as the type associated with this XPath, as follows:

```
xmlChoiceObjectMapping.addChoiceElement("mystring/text()", String.class);
xmlChoiceObjectMapping.addChoiceElement("myaddress", Address.class);
```

When any of these elements are encountered in the XML document, they are read in and set in the object as correct types.

Use this mapping to map to single choices or substitution groups (see [Section 53.2.10, "Substitution Groups"](#)) in an XML schema.

[Example 53–75](#) shows how to create an `XMLChoiceObjectMapping`, set the XPath (see [Section 53.2.3, "XPath Support"](#)), and then add the mapping to a descriptor.

**Example 53–75 Creating an XML Choice Object Mapping**

```
XMLChoiceObjectMapping addressMapping = new XMLChoiceObjectMapping();
addressMapping.setXPath("address", Address.class);
descriptor.addMapping(addressMapping);
...
```

See [Chapter 68, "Configuring an XML Choice Object Mapping"](#) for more information.

## 53.17 XML Choice Collection Mapping

The `oracle.toplink.ox.mappings.XMLChoiceCollectionMapping` is similar to `XMLChoiceObjectMapping` (see [Section 53.16, "XML Choice Object Mapping"](#)), except that you use it to handle reading and writing of XML documents containing a collection of choice or substitution group (see [Section 53.2.10, "Substitution Groups"](#)) elements.

[Example 53–76](#) shows how to create an `XMLChoiceCollectionMapping`, set the XPath, and then add the mapping to a descriptor.

**Example 53–76 Creating an XML Choice Collection Mapping**

```
XMLChoiceCollectionMapping addressMapping = new XMLChoiceCollectionMapping();
addressMapping.setXPath("address", Address.class);
descriptor.addMapping(addressMapping);
...
```

See [Chapter 69, "Configuring an XML Choice Collection Mapping"](#) for more information.

## 53.18 XML Any Attribute Mapping

The `oracle.toplink.ox.mappings.XMLAnyAttributeMapping` is a database mapping that you can use to map to an attribute in an object to any XML attributes contained on a specific element in the XML document. The attribute in the object will contain a map of attribute values keyed on a qualified name (`javax.xml.namespace.QName`). If one or more of the attributes found on the specified element is already mapped to another attribute in the object, TopLink will ignore that attribute during the unmarshal operation.

The `XMLAnyAttributeMapping` lets you set the XPath (see [Section 53.2.3, "XPath Support"](#)), however, this operation is optional for this type of mapping: if you do not set the XPath, the mapping will look for any attribute children directly owned by the current element.

[Example 53-77](#) shows how to create an `XMLAnyAttributeMapping`, set the XPath, and then add the mapping to a descriptor.

### ***Example 53-77 Creating an XML Any Attribute Mapping***

```
XMLAnyAttributeMapping addressMapping = new XMLAnyAttributeMapping();
addressMapping.setXPath("address", Address.class);
descriptor.addMapping(addressMapping);
...
```

See [Chapter 70, "Configuring an XML Any Attribute Mapping"](#) for more information.



---



---

## Configuring an XML Mapping

This chapter describes how to configure an XML mapping.

This chapter includes the following sections:

- [Introduction to XML Mapping Configuration](#)
- [Configuring Common XML Mapping Options](#)
- [Configuring Reference Descriptor](#)
- [Configuring Maps to Wildcard](#)
- [Configuring Source to Target Key Field Association](#)
- [Configuring Reference Class](#)
- [Configuring the Use of Inline Binary Data](#)
- [Configuring the Use of SwaRef Type](#)
- [Configuring the Choice Element](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 54.1 Introduction to XML Mapping Configuration

[Table 54–1](#) lists the types of XML mappings that you can configure and provides a cross-reference to the type-specific chapter that lists the configurable options supported by that type.

**Table 54–1** *Configuring XML Mappings*

Mapping Type	See Also...
XML direct mapping (see <a href="#">Section 53.3, "XML Direct Mapping"</a> )	<a href="#">Chapter 55, "Configuring an XML Direct Mapping"</a>
XML composite direct collection mapping (see <a href="#">Section 53.4, "XML Composite Direct Collection Mapping"</a> )	<a href="#">Chapter 56, "Configuring an XML Composite Direct Collection Mapping"</a>
XML composite object mapping (see <a href="#">Section 53.5, "XML Composite Object Mapping"</a> )	<a href="#">Chapter 57, "Configuring an XML Composite Object Mapping"</a>

**Table 54–1 (Cont.) Configuring XML Mappings**

<b>Mapping Type</b>	<b>See Also...</b>
XML composite collection mapping (see Section 53.6, "XML Composite Collection Mapping")	Chapter 58, "Configuring an XML Composite Collection Mapping"
XML any object mapping (see Section 53.7, "XML Any Object Mapping")	Chapter 59, "Configuring an XML Any Object Mapping"
XML any collection mapping (see Section 53.8, "XML Any Collection Mapping")	Chapter 60, "Configuring an XML Any Collection Mapping"
XML transformation mapping (see Section 53.9, "XML Transformation Mapping")	Chapter 61, "Configuring an XML Transformation Mapping"
XML object reference mapping (see Section 53.10, "XML Object Reference Mapping")	Chapter 62, "Configuring an XML Object Reference Mapping"
XML collection reference mapping (see Section 53.10, "XML Object Reference Mapping")	Chapter 62, "Configuring an XML Object Reference Mapping"
XML binary data mapping (see Section 53.10, "XML Object Reference Mapping")	Chapter 62, "Configuring an XML Object Reference Mapping"
XML binary data collection mapping (see Section 53.10, "XML Object Reference Mapping")	Chapter 62, "Configuring an XML Object Reference Mapping"
XML fragment mapping (see Section 53.10, "XML Object Reference Mapping")	Chapter 62, "Configuring an XML Object Reference Mapping"
XML fragment collection mapping (see Section 53.10, "XML Object Reference Mapping")	Chapter 62, "Configuring an XML Object Reference Mapping"
XML choice object mapping (see Section 53.10, "XML Object Reference Mapping")	Chapter 62, "Configuring an XML Object Reference Mapping"
XML choice collection mapping (see Section 53.10, "XML Object Reference Mapping")	Chapter 62, "Configuring an XML Object Reference Mapping"
XML any attribute mapping (see Section 53.10, "XML Object Reference Mapping")	Chapter 62, "Configuring an XML Object Reference Mapping"

For more information, see the following:

- Chapter 17, "Introduction to Mappings"
- Chapter 53, "Introduction to XML Mappings"

## 54.2 Configuring Common XML Mapping Options

Table 54–2 lists the configurable options shared by two or more XML mapping types. In addition to the configurable options described here, you must also configure the options described for the specific [XML Mapping Types](#), as shown in Table 54–1.

**Table 54–2 Common Options for XML Mappings**

Option	Oracle JDeveloper	TopLink Workbench	Java
Section 121.4, "Configuring XPath"	✓	✓	✓
Section 54.3, "Configuring Reference Descriptor"	✓	✓	✓
Section 121.14, "Configuring Container Policy"	✓	✓	✓
Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"	✓	✓	✓
Section 121.2, "Configuring Read-Only Mappings"	✓	✓	✓
Section 54.4, "Configuring Maps to Wildcard"	✓	✓	
Section 121.9, "Configuring a Serialized Object Converter"	✓	✓	✓
Section 121.10, "Configuring a Type Conversion Converter"	✓	✓	✓
Section 121.11, "Configuring an Object Type Converter"	✓	✓	✓
Section 121.12, "Configuring a Simple Type Translator"	✓	✓	✓
Section 121.19, "Configuring the Use of a Single Node"	✓	✓	✓
Section 121.20, "Configuring the Use of CDATA"			✓
Section 54.5, "Configuring Source to Target Key Field Association"	✓		✓
Section 54.6, "Configuring Reference Class"	✓		✓
Section 54.7, "Configuring the Use of Inline Binary Data"	✓		✓
Section 54.8, "Configuring the Use of SwaRef Type"	✓		✓
Section 54.9, "Configuring the Choice Element"	✓		✓

## 54.3 Configuring Reference Descriptor

For XML attributes that reference other descriptors (instead of a schema element), you may select a specific reference descriptor. If you do not specify a reference descriptor, TopLink uses the `xsi:type` attribute to determine the reference class object.

In TopLink versions prior to 11g (11.1.1.0), the reference class was required.

Table 54–3 summarizes which XML mappings support reference descriptor configuration.

**Table 54–3 XML Mapping Support for Reference Descriptor Configuration**

XML Mapping	How to Use Oracle JDeveloper	How to Configure a Reference Descriptor Using TopLink Workbench	How to Use Java
XML direct mapping (see Section 53.3, "XML Direct Mapping")			
XML composite direct collection mapping (see Section 53.4, "XML Composite Direct Collection Mapping")			
XML composite object mapping (see Section 53.5, "XML Composite Object Mapping")	✓	✓	✓
XML composite collection mapping (see Section 53.6, "XML Composite Collection Mapping")	✓	✓	✓

**Table 54–3 (Cont.) XML Mapping Support for Reference Descriptor Configuration**

<b>XML Mapping</b>	<b>How to Use Oracle JDeveloper</b>	<b>How to Configure a Reference Descriptor Using TopLink Workbench</b>	<b>How to Use Java</b>
XML any object mapping (see Section 53.7, "XML Any Object Mapping")			
XML any collection mapping (see Section 53.8, "XML Any Collection Mapping")			
XML transformation mapping (see Section 53.9, "XML Transformation Mapping")			
XML object reference mapping (see Section 53.10, "XML Object Reference Mapping")			
XML object collection mapping (see Section 53.11, "XML Collection Reference Mapping")			
XML binary data mapping (see Section 53.12, "XML Binary Data Mapping")			
XML binary data collection mapping (see Section 53.13, "XML Binary Data Collection Mapping")			
XML fragment mapping (see Section 53.14, "XML Fragment Mapping")			
XML fragment collection mapping (see Section 53.15, "XML Fragment Collection Mapping")			
XML choice object mapping (see Section 53.16, "XML Choice Object Mapping")			
XML choice collection mapping (see Section 53.17, "XML Choice Collection Mapping")			
XML any attribute mapping (see Section 53.18, "XML Any Attribute Mapping")			

### 54.3.1 How to Configure a Reference Descriptor Using TopLink Workbench

To specify a reference descriptor for an XML mapping that references another descriptor (instead of a schema element), use this procedure.

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

**Figure 54–1 General Tab, Reference Descriptor Field**

If this XML attribute refers to another descriptor (instead of a schema element), use the **Reference Descriptor** field to select a descriptor in the project.

## 54.4 Configuring Maps to Wildcard

This attribute applies only to Oracle JDeveloper TopLink Editor and TopLink Workbench. Use this option to solve "No XPath specified" problems (see [Section 5.3.5, "How to Use the Problems Window"](#)) for an XML mapping that does not need an XPath (see [Section 121.4, "Configuring XPath"](#)) for it maps to a wildcard.

If the XML mapping is owned by an `anyType` descriptor (see [Section 52.3, "Configuring for Complex Type of anyType"](#)), it cannot map to a wildcard, and you must specify an XPath.

[Table 54–4](#) summarizes which XML mappings support maps to wildcard configuration.

**Table 54–4 XML Mapping Support for Maps to Wildcard Configuration**

XML Mapping	How to Use Oracle JDeveloper	How to Configure Maps to Wildcard Using TopLink Workbench	How to Use Java
XML direct mapping (see <a href="#">Section 53.3, "XML Direct Mapping"</a> )			
XML composite direct collection mapping (see <a href="#">Section 53.4, "XML Composite Direct Collection Mapping"</a> )			
XML composite object mapping (see <a href="#">Section 53.5, "XML Composite Object Mapping"</a> )			
XML composite collection mapping (see <a href="#">Section 53.6, "XML Composite Collection Mapping"</a> )			
XML any object mapping (see <a href="#">Section 53.7, "XML Any Object Mapping"</a> )	✓	✓	
XML any collection mapping (see <a href="#">Section 53.8, "XML Any Collection Mapping"</a> )	✓	✓	

**Table 54–4 (Cont.) XML Mapping Support for Maps to Wildcard Configuration**

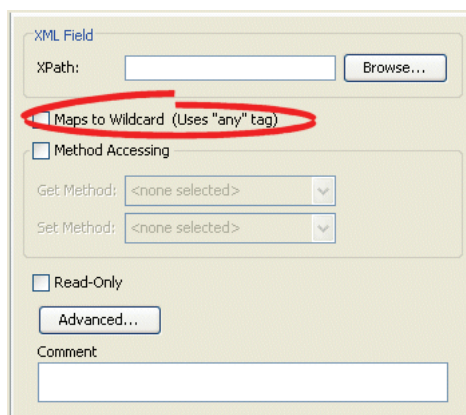
<b>XML Mapping</b>	<b>How to Use Oracle JDeveloper</b>	<b>How to Configure Maps to Wildcard Using TopLink Workbench</b>	<b>How to Use Java</b>
XML transformation mapping (see Section 53.9, "XML Transformation Mapping")			
XML object reference mapping (see Section 53.10, "XML Object Reference Mapping")			
XML object collection mapping (see Section 53.11, "XML Collection Reference Mapping")			
XML binary data mapping (see Section 53.12, "XML Binary Data Mapping")			
XML binary data collection mapping (see Section 53.13, "XML Binary Data Collection Mapping")			
XML fragment mapping (see Section 53.14, "XML Fragment Mapping")			
XML fragment collection mapping (see Section 53.15, "XML Fragment Collection Mapping")			
XML choice object mapping (see Section 53.16, "XML Choice Object Mapping")			
XML choice collection mapping (see Section 53.17, "XML Choice Collection Mapping")			
XML any attribute mapping (see Section 53.18, "XML Any Attribute Mapping")			

### 54.4.1 How to Configure Maps to Wildcard Using TopLink Workbench

To specify a map a schema element using the `xs:any` declaration, use this procedure.

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.

**Figure 54–2 Mapping Tab, Maps to Wildcard Option**



If the XML mapping is not owned by an anyType descriptor (see [Section 52.3, "Configuring for Complex Type of anyType"](#)) and maps to a wildcard, then you do not need to specify an XPath (see [Section 121.4, "Configuring XPath"](#)). Select the **Maps to Wildcard (uses "any" tag)** option to clear the missing XPath neediness message.

If the XML mapping is owned by an anyType descriptor, it cannot map to a wildcard and you must specify an XPath. Deselect the **Maps to Wildcard (Uses "any" tag)** option and ensure that you specify an XPath.

## 54.5 Configuring Source to Target Key Field Association

This option is applicable to key on source-based mappings. Use this option to add a source and target XPath pair to the map of such key pairs.

[Table 54–5](#) summarizes which XML mappings support source to target key field association configuration.

**Table 54–5 XML Mapping Support for Source to Target Key Field Association Configuration**

XML Mapping	How to Use Oracle JDeveloper	How to Use TopLink Workbench	How to Configure Source to Target Key Field Association Using Java
XML direct mapping (see <a href="#">Section 53.3, "XML Direct Mapping"</a> )			
XML composite direct collection mapping (see <a href="#">Section 53.4, "XML Composite Direct Collection Mapping"</a> )			
XML composite object mapping (see <a href="#">Section 53.5, "XML Composite Object Mapping"</a> )			
XML composite collection mapping (see <a href="#">Section 53.6, "XML Composite Collection Mapping"</a> )			
XML any object mapping (see <a href="#">Section 53.7, "XML Any Object Mapping"</a> )			
XML any collection mapping (see <a href="#">Section 53.8, "XML Any Collection Mapping"</a> )			
XML transformation mapping (see <a href="#">Section 53.9, "XML Transformation Mapping"</a> )			
XML object reference mapping (see <a href="#">Section 53.10, "XML Object Reference Mapping"</a> )	✓		✓
XML object collection mapping (see <a href="#">Section 53.11, "XML Collection Reference Mapping"</a> )	✓		✓
XML binary data mapping (see <a href="#">Section 53.12, "XML Binary Data Mapping"</a> )			

**Table 54–5 (Cont.) XML Mapping Support for Source to Target Key Field Association Configuration**

XML Mapping	How to Use Oracle JDeveloper	How to Use TopLink Workbench	How to Configure Source to Target Key Field Association Using Java
XML binary data collection mapping (see Section 53.13, "XML Binary Data Collection Mapping")			
XML fragment mapping (see Section 53.14, "XML Fragment Mapping")			
XML fragment collection mapping (see Section 53.15, "XML Fragment Collection Mapping")			
XML choice object mapping (see Section 53.16, "XML Choice Object Mapping")			
XML choice collection mapping (see Section 53.17, "XML Choice Collection Mapping")			
XML any attribute mapping (see Section 53.18, "XML Any Attribute Mapping")			

### 54.5.1 How to Configure Source to Target Key Field Association Using Java

To configure the source to target key field association for your mapping, use the `XMLObjectReferenceMapping.addSourceToTargetKeyFieldAssociation` method to add a specified source and target XPath pair to the map.

## 54.6 Configuring Reference Class

This option is applicable to key on source-based mappings.

Use this option to define the reference class, whose instances your XML object reference mapping will store in the domain objects.

Table 54–6 summarizes which XML mappings support source to target key field association configuration.

**Table 54–6 XML Mapping Support for Reference Class Configuration**

XML Mapping	How to Use Oracle JDeveloper	How to Use TopLink Workbench	How to Configure Reference Class Using Java
XML direct mapping (see Section 53.3, "XML Direct Mapping")			
XML composite direct collection mapping (see Section 53.4, "XML Composite Direct Collection Mapping")			



**Table 54–6 (Cont.) XML Mapping Support for Reference Class Configuration**

<b>XML Mapping</b>	<b>How to Use Oracle JDeveloper</b>	<b>How to Use TopLink Workbench</b>	<b>How to Configure Reference Class Using Java</b>
XML composite object mapping (see Section 53.5, "XML Composite Object Mapping")			
XML composite collection mapping (see Section 53.6, "XML Composite Collection Mapping")			
XML any object mapping (see Section 53.7, "XML Any Object Mapping")			
XML any collection mapping (see Section 53.8, "XML Any Collection Mapping")			
XML transformation mapping (see Section 53.9, "XML Transformation Mapping")			
XML object reference mapping (see Section 53.10, "XML Object Reference Mapping")	✓		✓
XML object collection mapping (see Section 53.11, "XML Collection Reference Mapping")	✓		✓
XML binary data mapping (see Section 53.12, "XML Binary Data Mapping")			
XML binary data collection mapping (see Section 53.13, "XML Binary Data Collection Mapping")			
XML fragment mapping (see Section 53.14, "XML Fragment Mapping")			
XML fragment collection mapping (see Section 53.15, "XML Fragment Collection Mapping")			
XML choice object mapping (see Section 53.16, "XML Choice Object Mapping")			
XML choice collection mapping (see Section 53.17, "XML Choice Collection Mapping")			
XML any attribute mapping (see Section 53.18, "XML Any Attribute Mapping")			

### 54.6.1 How to Configure Reference Class Using Java

To configure a reference class for your mapping, use the `AggregateMapping` method `setReferenceClass`.

## 54.7 Configuring the Use of Inline Binary Data

This option is applicable to binary data mappings.

Use this option to define whether or not there should always be inline binary data for this mapping.

Table 54–7 summarizes which XML mappings support the use of inline binary data configuration.

**Table 54–7 XML Mapping Support for the Use of Inline Binary Data Configuration**

XML Mapping	How to Use Oracle JDeveloper	How to Use TopLink Workbench	How to Configure the Use of Inline Binary Data Using Java
XML direct mapping (see Section 53.3, "XML Direct Mapping")			
XML composite direct collection mapping (see Section 53.4, "XML Composite Direct Collection Mapping")			
XML composite object mapping (see Section 53.5, "XML Composite Object Mapping")			
XML composite collection mapping (see Section 53.6, "XML Composite Collection Mapping")			
XML any object mapping (see Section 53.7, "XML Any Object Mapping")			
XML any collection mapping (see Section 53.8, "XML Any Collection Mapping")			
XML transformation mapping (see Section 53.9, "XML Transformation Mapping")			
XML object reference mapping (see Section 53.10, "XML Object Reference Mapping")			
XML object collection mapping (see Section 53.11, "XML Collection Reference Mapping")			
XML binary data mapping (see Section 53.12, "XML Binary Data Mapping")	✓		✓
XML binary data collection mapping (see Section 53.13, "XML Binary Data Collection Mapping")	✓		✓
XML fragment mapping (see Section 53.14, "XML Fragment Mapping")			
XML fragment collection mapping (see Section 53.15, "XML Fragment Collection Mapping")			

**Table 54–7 (Cont.) XML Mapping Support for the Use of Inline Binary Data Configuration**

XML Mapping	How to Use Oracle JDeveloper	How to Use TopLink Workbench	How to Configure the Use of Inline Binary Data Using Java
XML choice object mapping (see <a href="#">Section 53.16, "XML Choice Object Mapping"</a> )			
XML choice collection mapping (see <a href="#">Section 53.17, "XML Choice Collection Mapping"</a> )			
XML any attribute mapping (see <a href="#">Section 53.18, "XML Any Attribute Mapping"</a> )			

### 54.7.1 How to Configure the Use of Inline Binary Data Using Java

To configure the use of inline binary data for your mapping, use the `XMLBinaryDataMapping` or `XMLBinaryDataCollectionMapping` method `setShouldInlineBinaryData`. If you set it to `true`, you disable consideration for attachment handling for this mapping and indicate that you only want inline data.

## 54.8 Configuring the Use of SwaRef Type

This option is applicable to binary data mappings.

Use this option to specify that the target node of this mapping is of type `xs:swaRef`.

[Table 54–8](#) summarizes which XML mappings support the use of SwaRef type configuration.

**Table 54–8 XML Mapping Support for the Use of SwaRef Type Configuration**

XML Mapping	How to Use Oracle JDeveloper	How to Use TopLink Workbench	How to Configure the Use of SwaRef Type Using Java
XML direct mapping (see <a href="#">Section 53.3, "XML Direct Mapping"</a> )			
XML composite direct collection mapping (see <a href="#">Section 53.4, "XML Composite Direct Collection Mapping"</a> )			
XML composite object mapping (see <a href="#">Section 53.5, "XML Composite Object Mapping"</a> )			
XML composite collection mapping (see <a href="#">Section 53.6, "XML Composite Collection Mapping"</a> )			
XML any object mapping (see <a href="#">Section 53.7, "XML Any Object Mapping"</a> )			

**Table 54–8 (Cont.) XML Mapping Support for the Use of SwaRef Type Configuration**

XML Mapping	How to Use Oracle JDeveloper	How to Use TopLink Workbench	How to Configure the Use of SwaRef Type Using Java
XML any collection mapping (see Section 53.8, "XML Any Collection Mapping")			
XML transformation mapping (see Section 53.9, "XML Transformation Mapping")			
XML object reference mapping (see Section 53.10, "XML Object Reference Mapping")			
XML object collection mapping (see Section 53.11, "XML Collection Reference Mapping")			
XML binary data mapping (see Section 53.12, "XML Binary Data Mapping")	✓		✓
XML binary data collection mapping (see Section 53.13, "XML Binary Data Collection Mapping")	✓		✓
XML fragment mapping (see Section 53.14, "XML Fragment Mapping")			
XML fragment collection mapping (see Section 53.15, "XML Fragment Collection Mapping")			
XML choice object mapping (see Section 53.16, "XML Choice Object Mapping")			
XML choice collection mapping (see Section 53.17, "XML Choice Collection Mapping")			
XML any attribute mapping (see Section 53.18, "XML Any Attribute Mapping")			

### 54.8.1 How to Configure the Use of SwaRef Type Using Java

To configure the use of SwaRef type for your mapping, use the `XMLBinaryDataMapping` or `XMLBinaryDataCollectionMapping` method `setSwaRef`. If you set it to `true`, you indicate that the target node of this mapping is of type `xs:swaref`.

## 54.9 Configuring the Choice Element

This option is applicable to choice mappings.

Use this option to specify an XPath and the type associated with this XPath.

Table 54–9 summarizes which XML mappings support the choice element configuration.

**Table 54–9 XML Mapping Support for the Choice Element Configuration**

<b>XML Mapping</b>	<b>How to Use Oracle JDeveloper</b>	<b>How to Use TopLink Workbench</b>	<b>How to Configure the Choice Element Using Java</b>
XML direct mapping (see Section 53.3, "XML Direct Mapping")			
XML composite direct collection mapping (see Section 53.4, "XML Composite Direct Collection Mapping")			
XML composite object mapping (see Section 53.5, "XML Composite Object Mapping")			
XML composite collection mapping (see Section 53.6, "XML Composite Collection Mapping")			
XML any object mapping (see Section 53.7, "XML Any Object Mapping")			
XML any collection mapping (see Section 53.8, "XML Any Collection Mapping")			
XML transformation mapping (see Section 53.9, "XML Transformation Mapping")			
XML object reference mapping (see Section 53.10, "XML Object Reference Mapping")			
XML object collection mapping (see Section 53.11, "XML Collection Reference Mapping")			
XML binary data mapping (see Section 53.12, "XML Binary Data Mapping")			
XML binary data collection mapping (see Section 53.13, "XML Binary Data Collection Mapping")			
XML fragment mapping (see Section 53.14, "XML Fragment Mapping")			
XML fragment collection mapping (see Section 53.15, "XML Fragment Collection Mapping")			
XML choice object mapping (see Section 53.16, "XML Choice Object Mapping")	✓		✓
XML choice collection mapping (see Section 53.17, "XML Choice Collection Mapping")	✓		✓
XML any attribute mapping (see Section 53.18, "XML Any Attribute Mapping")			

### 54.9.1 How to Configure the Choice Element Using Java

Use the following `XMLChoiceObjectMapping` or `XMLChoiceCollectionMapping` methods to add choice element:

- `addChoiceElement(String xpath, Class elementType)`
- `addChoiceElement(String xpath, String elementTypeName)`

## Configuring an XML Direct Mapping

This chapter describes the various components that you must configure in order to use an XML direct mapping.

This chapter includes the following section:

- [Introduction to XML Direct Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 55.1 Introduction to XML Direct Mapping Configuration

Table 55–1 lists the configurable options for an XML direct mapping.

**Table 55–1 Configurable Options for XML Direct Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
XPath (see <a href="#">Section 121.4, "Configuring XPath"</a> )	✓	✓	✓
Simple type translator (see <a href="#">Section 121.12, "Configuring a Simple Type Translator"</a> )	✓	✓	✓
Use of a single node (see <a href="#">Section 121.19, "Configuring the Use of a Single Node"</a> )			✓
Use of CDATA (see <a href="#">Section 121.20, "Configuring the Use of CDATA"</a> )			✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Default null value (see <a href="#">Section 121.5, "Configuring a Default Null Value at the Mapping Level"</a> )	✓	✓	✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	✓
Serialized object converter (see <a href="#">Section 121.9, "Configuring a Serialized Object Converter"</a> )	✓	✓	✓
Type conversion converter (see <a href="#">Section 121.10, "Configuring a Type Conversion Converter"</a> )	✓	✓	✓
Object type converter (see <a href="#">Section 121.11, "Configuring an Object Type Converter"</a> )	✓	✓	

**Table 55–1 (Cont.) Configurable Options for XML Direct Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
JAXB typesafe enumeration converter (see <a href="#">Section 121.13</a> , " <a href="#">Configuring a JAXB Typesafe Enumeration Converter</a> ")			✓

For more information, see the following:

- [Section 53.3](#), "XML Direct Mapping"
- [Chapter 54](#), "Configuring an XML Mapping"



## Configuring an XML Composite Direct Collection Mapping

This chapter describes the various components that you must configure in order to use an XML composite direct collection mapping.

This chapter includes the following section:

- [Introduction to XML Composite Direct Collection Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 56.1 Introduction to XML Composite Direct Collection Mapping Configuration

[Table 56–1](#) lists the configurable options for an XML direct collection mapping.

**Table 56–1** Configurable Options for XML Direct Collection Mapping

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
XPath (see <a href="#">Section 121.4, "Configuring XPath"</a> )	✓	✓	✓
Simple type translator (see <a href="#">Section 121.12, "Configuring a Simple Type Translator"</a> )	✓	✓	✓
Use of a single node (see <a href="#">Section 121.19, "Configuring the Use of a Single Node"</a> )			✓
Use of CDATA (see <a href="#">Section 121.20, "Configuring the Use of CDATA"</a> )			✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Container policy (see <a href="#">Section 121.14, "Configuring Container Policy"</a> )	✓	✓	✓
Comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	
Serialized object converter (see <a href="#">Section 121.9, "Configuring a Serialized Object Converter"</a> )	✓	✓	✓
Type conversion converter (see <a href="#">Section 121.10, "Configuring a Type Conversion Converter"</a> )	✓	✓	✓

**Table 56–1 (Cont.) Configurable Options for XML Direct Collection Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Object type converter (see Section 121.11, "Configuring an Object Type Converter")	✓	✓	
JAXB typesafe enumeration converter (see Section 121.13, "Configuring a JAXB Typesafe Enumeration Converter")			✓

For more information, see the following:

- Section 53.4, "XML Composite Direct Collection Mapping"
- Chapter 54, "Configuring an XML Mapping"

## Configuring an XML Composite Object Mapping

This chapter describes the various components that you must configure in order to use an XML composite object mapping.

This chapter includes the following section:

- [Introduction to XML Composite Object Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 57.1 Introduction to XML Composite Object Mapping Configuration

[Table 57–1](#) lists the configurable options for an XML composite object mapping.

**Table 57–1** Configurable Options for XML Composite Object Mapping

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
XPath (see <a href="#">Section 121.4, "Configuring XPath"</a> )	✓	✓	✓
Reference descriptor (see <a href="#">Section 54.3, "Configuring Reference Descriptor"</a> )	✓	✓	✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	

For more information, see the following:

- [Section 53.5, "XML Composite Object Mapping"](#)
- [Chapter 54, "Configuring an XML Mapping"](#)



## Configuring an XML Composite Collection Mapping

This chapter describes the various components that you must configure in order to use an XML composite collection mapping.

This chapter includes the following section:

- [Introduction to XML Composite Collection Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 58.1 Introduction to XML Composite Collection Mapping Configuration

[Table 58–1](#) lists the configurable options for an XML composite collection mapping.

**Table 58–1** Configurable Options for XML Composite Collection Mapping

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
XPath (see <a href="#">Section 121.4, "Configuring XPath"</a> )	✓	✓	✓
Reference descriptor (see <a href="#">Section 54.3, "Configuring Reference Descriptor"</a> )	✓	✓	✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Container policy (see <a href="#">Section 121.14, "Configuring Container Policy"</a> )	✓	✓	✓
Comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	

For more information, see the following:

- [Section 53.6, "XML Composite Collection Mapping"](#)
- [Chapter 54, "Configuring an XML Mapping"](#)



## Configuring an XML Any Object Mapping

This chapter describes the various components that you must configure in order to use an XML any object mapping.

This chapter includes the following section:

- [Introduction to XML Any Object Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 59.1 Introduction to XML Any Object Mapping Configuration

[Table 59–1](#) lists the configurable options for an XML any object mapping.

**Table 59–1 Configurable Options for XML Any Object Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
XPath (see <a href="#">Section 121.4, "Configuring XPath"</a> )	✓	✓	✓
Maps to wildcard (see <a href="#">Section 54.4, "Configuring Maps to Wildcard"</a> )	✓	✓	
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	

For more information, see the following:

- [Section 53.7, "XML Any Object Mapping"](#)
- [Chapter 54, "Configuring an XML Mapping"](#)





## Configuring an XML Any Collection Mapping

This chapter describes the various components that you must configure in order to use an XML any collection mapping.

This chapter includes the following section:

- [Introduction to XML Any Collection Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 60.1 Introduction to XML Any Collection Mapping Configuration

[Table 60–1](#) lists the configurable options for an XML any collection mapping.

**Table 60–1** Configurable Options for XML Any Collection Mapping

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
XPath (see <a href="#">Section 121.4, "Configuring XPath"</a> )	✓	✓	✓
Maps to wildcard (see <a href="#">Section 54.4, "Configuring Maps to Wildcard"</a> )	✓	✓	
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Container policy (see <a href="#">Section 121.14, "Configuring Container Policy"</a> )	✓	✓	✓
Comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	

For more information, see the following:

- [Section 53.8, "XML Any Collection Mapping"](#)
- [Chapter 54, "Configuring an XML Mapping"](#)



## Configuring an XML Transformation Mapping

This chapter describes the various components that you must configure in order to use an XML transformation mapping.

This chapter includes the following section:

- [Introduction to XML Transformation Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 61.1 Introduction to XML Transformation Mapping Configuration

[Table 61–1](#) lists the configurable options for a XML transformation mapping.

**Table 61–1** Configurable Options for XML Transformation Mapping

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
<a href="#">Attribute transformer</a> (see <a href="#">Section 121.15, "Configuring Attribute Transformer"</a> )	✓	✓	✓
<a href="#">Field transformer associations</a> (see <a href="#">Section 121.16, "Configuring Field Transformer Associations"</a> )	✓	✓	✓
<a href="#">Method or direct field access</a> (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
<a href="#">Read-only</a> (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
<a href="#">Mutable mappings</a> (see <a href="#">Section 121.17, "Configuring Mutable Mappings"</a> )	✓	✓	✓
<a href="#">Comments</a> (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	
<a href="#">Indirection (lazy loading)</a> (see <a href="#">Section 121.3, "Configuring Indirection (Lazy Loading)"</a> )			✓

For more information, see the following:

- [Section 53.9, "XML Transformation Mapping"](#)
- [Chapter 54, "Configuring an XML Mapping"](#)



## Configuring an XML Object Reference Mapping

This chapter describes the various components that you must configure in order to use an XML object reference mapping.

This chapter includes the following sections:

- [Introduction to XML Object Reference Mapping](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 62.1 Introduction to XML Object Reference Mapping

[Table 62–1](#) lists the configurable options for an XML object reference mapping.

**Table 62–1 Configurable Options for XML Object Reference Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Source to target key field association (see <a href="#">Section 54.5, "Configuring Source to Target Key Field Association"</a> )	✓		✓
Reference class (see <a href="#">Section 54.6, "Configuring Reference Class"</a> )	✓		✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	

For more information, see the following:

- [Section 53.10, "XML Object Reference Mapping"](#)
- [Chapter 54, "Configuring an XML Mapping"](#)



## Configuring an XML Collection Reference Mapping

This chapter describes the various components that you must configure in order to use an XML collection reference mapping.

This chapter includes the following section:

- [Introduction to XML Collection Reference Mapping](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 63.1 Introduction to XML Collection Reference Mapping

[Table 63–1](#) lists the configurable options for an XML collection reference mapping.

**Table 63–1** Configurable Options for XML Collection Reference Mapping

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Source to target key field association (see <a href="#">Section 54.5, "Configuring Source to Target Key Field Association"</a> )	✓		✓
Reference class (see <a href="#">Section 54.6, "Configuring Reference Class"</a> )	✓		✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Container policy (see <a href="#">Section 121.14, "Configuring Container Policy"</a> )	✓	✓	✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	

For more information, see the following:

- [Section 53.11, "XML Collection Reference Mapping"](#)
- [Chapter 54, "Configuring an XML Mapping"](#)





## Configuring an XML Binary Data Mapping

This chapter describes the various components that you must configure in order to use an XML binary data mapping.

This chapter includes the following section:

- [Introduction to XML Binary Data Mapping](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 64.1 Introduction to XML Binary Data Mapping

[Table 64–1](#) lists the configurable options for an XML binary data mapping.

**Table 64–1 Configurable Options for XML Binary Data Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
XPath (see <a href="#">Section 121.4, "Configuring XPath"</a> )	✓	✓	✓
Simple type translator (see <a href="#">Section 121.12, "Configuring a Simple Type Translator"</a> )	✓	✓	✓
Use of a single node (see <a href="#">Section 121.19, "Configuring the Use of a Single Node"</a> )			✓
Use of CDATA (see <a href="#">Section 121.20, "Configuring the Use of CDATA"</a> )			✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Default null value (see <a href="#">Section 121.5, "Configuring a Default Null Value at the Mapping Level"</a> )	✓	✓	✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	✓
Serialized object converter (see <a href="#">Section 121.9, "Configuring a Serialized Object Converter"</a> )	✓	✓	✓
Type conversion converter (see <a href="#">Section 121.10, "Configuring a Type Conversion Converter"</a> )	✓	✓	✓
Object type converter (see <a href="#">Section 121.11, "Configuring an Object Type Converter"</a> )	✓	✓	

**Table 64–1 (Cont.) Configurable Options for XML Binary Data Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
JAXB typesafe enumeration converter (see <a href="#">Section 121.13</a> , "Configuring a JAXB Typesafe Enumeration Converter")			✓
Use of inline binary data (see <a href="#">Section 54.7</a> , "Configuring the Use of Inline Binary Data")	✓		✓
Use of SwaRef (see <a href="#">Section 54.8</a> , "Configuring the Use of SwaRef Type")	✓		✓

For more information, see the following:

- [Section 53.12](#), "XML Binary Data Mapping"
- [Chapter 54](#), "Configuring an XML Mapping"

## Configuring an XML Binary Data Collection Mapping

This chapter describes the various components that you must configure in order to use an XML binary data collection mapping.

This chapter includes the following section:

- [Introduction to XML Binary Data Collection Mapping](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 65.1 Introduction to XML Binary Data Collection Mapping

[Table 65–1](#) lists the configurable options for an XML binary data collection mapping.

**Table 65–1 Configurable Options for XML Binary Data Collection Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
XPath (see <a href="#">Section 121.4, "Configuring XPath"</a> )	✓	✓	✓
Simple type translator (see <a href="#">Section 121.12, "Configuring a Simple Type Translator"</a> )	✓	✓	✓
Use of a single node (see <a href="#">Section 121.19, "Configuring the Use of a Single Node"</a> )	✓	✓	✓
Use of CDATA (see <a href="#">Section 121.20, "Configuring the Use of CDATA"</a> )			✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Default null value (see <a href="#">Section 121.5, "Configuring a Default Null Value at the Mapping Level"</a> )	✓	✓	✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	✓
Serialized object converter (see <a href="#">Section 121.9, "Configuring a Serialized Object Converter"</a> )	✓	✓	✓
Type conversion converter (see <a href="#">Section 121.10, "Configuring a Type Conversion Converter"</a> )	✓	✓	✓

**Table 65–1 (Cont.) Configurable Options for XML Binary Data Collection Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Object type converter (see Section 121.11, "Configuring an Object Type Converter")	✓	✓	
JAXB typesafe enumeration converter (see Section 121.13, "Configuring a JAXB Typesafe Enumeration Converter")			✓
Container policy (see Section 121.14, "Configuring Container Policy")	✓	✓	✓
Use of inline binary data (see Section 54.7, "Configuring the Use of Inline Binary Data")	✓		✓
Use of SwaRef (see Section 54.8, "Configuring the Use of SwaRef Type")	✓		✓

For more information, see the following:

- Section 53.13, "XML Binary Data Collection Mapping"
- Chapter 54, "Configuring an XML Mapping"

## Configuring an XML Fragment Mapping

This chapter describes the various components that you must configure in order to use an XML fragment mapping.

This chapter includes the following section:

- [Introduction to XML Fragment Mapping](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 66.1 Introduction to XML Fragment Mapping

Table 66–1 lists the configurable options for an XML fragment mapping.

**Table 66–1 Configurable Options for XML Fragment Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
XPath (see <a href="#">Section 121.4, "Configuring XPath"</a> )	✓	✓	✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Default null value (see <a href="#">Section 121.5, "Configuring a Default Null Value at the Mapping Level"</a> )	✓	✓	✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	✓
Serialized object converter (see <a href="#">Section 121.9, "Configuring a Serialized Object Converter"</a> )	✓	✓	✓
Type conversion converter (see <a href="#">Section 121.10, "Configuring a Type Conversion Converter"</a> )	✓	✓	✓
Object type converter (see <a href="#">Section 121.11, "Configuring an Object Type Converter"</a> )	✓	✓	
JAXB typesafe enumeration converter (see <a href="#">Section 121.13, "Configuring a JAXB Typesafe Enumeration Converter"</a> )			✓

For more information, see the following:

- [Section 53.14, "XML Fragment Mapping"](#)
- [Chapter 54, "Configuring an XML Mapping"](#)



## Configuring an XML Fragment Collection Mapping

This chapter describes the various components that you must configure in order to use an XML fragment collection mapping.

This chapter includes the following section:

- [Introduction to XML Fragment Collection Mapping](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 67.1 Introduction to XML Fragment Collection Mapping

[Table 67–1](#) lists the configurable options for an XML fragment collection mapping.

**Table 67–1 Configurable Options for XML Fragment Collection Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
XPath (see <a href="#">Section 121.4, "Configuring XPath"</a> )	✓	✓	✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Default null value (see <a href="#">Section 121.5, "Configuring a Default Null Value at the Mapping Level"</a> )	✓	✓	✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	✓
Serialized object converter (see <a href="#">Section 121.9, "Configuring a Serialized Object Converter"</a> )	✓	✓	✓
Type conversion converter (see <a href="#">Section 121.10, "Configuring a Type Conversion Converter"</a> )	✓	✓	✓
Object type converter (see <a href="#">Section 121.11, "Configuring an Object Type Converter"</a> )	✓	✓	
JAXB typesafe enumeration converter (see <a href="#">Section 121.13, "Configuring a JAXB Typesafe Enumeration Converter"</a> )			✓

For more information, see the following:

- [Section 53.15, "XML Fragment Collection Mapping"](#)

- [Chapter 54, "Configuring an XML Mapping"](#)



## Configuring an XML Choice Object Mapping

This chapter describes the various components that you must configure in order to use an XML choice object mapping.

This chapter includes the following sections:

- [Introduction to XML Choice Object Mapping](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 68.1 Introduction to XML Choice Object Mapping

[Table 68–1](#) lists the configurable options for an XML choice object mapping.

**Table 68–1 Configurable Options for XML Choice Object Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Choice element (see <a href="#">Section 54.9, "Configuring the Choice Element"</a> )	✓		✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	

For more information, see the following:

- [Section 53.16, "XML Choice Object Mapping"](#)
- [Chapter 54, "Configuring an XML Mapping"](#)



## Configuring an XML Choice Collection Mapping

This chapter describes the various components that you must configure in order to use an XML choice collection mapping.

This chapter includes the following section:

- [Introduction to XML Choice Collection Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 69.1 Introduction to XML Choice Collection Mapping Configuration

[Table 69–1](#) lists the configurable options for an XML choice collection mapping.

**Table 69–1** Configurable Options for XML Choice Collection Mapping

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
XPath (see <a href="#">Section 121.4, "Configuring XPath"</a> )	✓	✓	✓
Choice element (see <a href="#">Section 54.9, "Configuring the Choice Element"</a> )	✓		✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Container policy (see <a href="#">Section 121.14, "Configuring Container Policy"</a> )	✓	✓	✓
Comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	

For more information, see the following:

- [Section 53.17, "XML Choice Collection Mapping"](#)
- [Chapter 54, "Configuring an XML Mapping"](#)



## Configuring an XML Any Attribute Mapping

This chapter describes the various components that you must configure in order to use an XML any attribute mapping.

This chapter includes the following section:

- [Introduction to XML Any Attribute Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 70.1 Introduction to XML Any Attribute Mapping Configuration

[Table 70–1](#) lists the configurable options for an XML any attribute mapping.

**Table 70–1 Configurable Options for XML Any Attribute Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
XPath (see <a href="#">Section 121.4, "Configuring XPath"</a> )	✓	✓	✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Container policy (see <a href="#">Section 121.14, "Configuring Container Policy"</a> )	✓	✓	✓
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	

For more information, see the following:

- [Section 53.18, "XML Any Attribute Mapping"](#)
- [Chapter 54, "Configuring an XML Mapping"](#)



# Part XVII

---

## EIS Projects

This part describes EIS projects and contains the following chapters:

- [Chapter 71, "Introduction to EIS Projects"](#)  
This chapter introduces EIS project concepts.
- [Chapter 72, "Creating an EIS Project"](#)  
This chapter explains how to create EIS projects.
- [Chapter 73, "Configuring an EIS Project"](#)  
This chapter explains how to configure EIS projects.





---

---

## Introduction to EIS Projects

This chapter provides an overview of EIS projects and their components.

This chapter includes the following section:

- [EIS Project Concepts](#)

For information on project concepts and features common to more than one type of TopLink projects, see [Chapter 15, "Introduction to Projects"](#).

### 71.1 EIS Project Concepts

Use an EIS project for transactional persistence of Java objects to a *nonrelational* data source accessed using a Java EE Connector Architecture (JCA) adapter and EIS records.

JCA provides a Common Client Interface (CCI) API to access nonrelational EIS. This provides a similar interface to nonrelational data sources as JDBC provides for relational data sources. This API defines several types of nonrelational record types including mapped and indexed. XML has emerged as the standard format to exchange data, and most leading JCA adapter providers have extended the CCI API to define XML data records.

To use a JCA adapter with TopLink EIS, the JCA adapter must support the JCA CCI interface. At run time, your JCA adapter and the `java.connector.jar` file (that contains the `javax.resource.cci` and `javax.resource.spi` interfaces that TopLink EIS uses) must be on your application or application server classpath.

If you are using TopLink Workbench, you must add your JCA adapter to the TopLink Workbench classpath. By default, TopLink Workbench updates its classpath to include the Java 1.5.*n* `connector.jar` file from `<ORACLE_HOME>/lib/java/api`. If this version of the `connector.jar` file is incompatible with your environment, edit the `workbench.cmd` or `workbench.sh` file in `<TOPLINK_HOME>/bin` to change the path to this file. For more information, see [Section 5.2, "Configuring the TopLink Workbench Environment"](#).

EIS includes legacy data sources, enterprise applications, legacy applications, and other information systems. These systems include such sources as Customer Information Control System (CICS), Virtual Storage Access Method (VSAM), Information Management System (IMS), ADATABASE database, and flat files.

Oracle recommends using EIS projects to integrate TopLink with a legacy or nonrelational data source. Other methods of accessing EIS data sources include:

- Using a specialized JDBC driver that allows connecting to an EIS system as if it were a relational database. You could use a TopLink relational project with these drivers (see [Chapter 18, "Introduction to Relational Projects"](#)).

- Linking to or integrating with the EIS data from a relational database, such as Oracle Database.
- Using a proprietary API to access the EIS system. In this case it may be possible to wrap the API with a JCA CCI interface to allow usage with a TopLink EIS project.

TopLink provides support for mapping Java objects to EIS mapped, indexed, and XML records, through JCA, using the TopLink mappings described in [Part XIX, "EIS Mappings"](#).

You configure a TopLink EIS descriptor to use a particular EIS record format (see [Section 76.4, "Configuring Record Format"](#)). TopLink EIS mappings use their EIS descriptor's record format configuration to determine how to map their Java objects to EIS records.

If you use XML records, the TopLink runtime performs XML data conversion based on one or more XML schemas. In an EIS project that uses XML records, TopLink Workbench directly references schemas in the deployment XML, and exports mappings configured with respect to the schemas you specify. For information on how to use TopLink Workbench with XML schemas, see [Section 5.6, "Using XML Schemas"](#). For information on how TopLink supports XML namespaces, see [Section 15.4, "XML Namespaces Overview"](#).

[Table 71–1](#) describes the components of an EIS project.

**Table 71–1 EIS Project Components**

Component	Supported Types
Data Source	For more information, see the following: <ul style="list-style-type: none"> <li>■ <a href="#">Section 96.1.2.2, "EISLogin"</a></li> <li>■ <a href="#">Section 96.1.3.2, "EIS Platforms"</a></li> </ul>
Descriptors	For more information, see <a href="#">Section 74.1, "EIS Descriptor Concepts"</a> .
Mappings	For more information, see the following: <ul style="list-style-type: none"> <li>■ <a href="#">Part VIII, "Mappings"</a></li> <li>■ <a href="#">Part XIX, "EIS Mappings"</a></li> </ul>

You can create an EIS project with TopLink Workbench for use with EIS XML records (see [Section 72.2, "Creating an EIS Project with XML Records"](#)) or you can build an EIS project in Java for use with any supported EIS record type (see [Section 72.3, "Creating an EIS Project with Indexed or Mapped Records"](#)).

In an EIS project, your EIS interactions (see [Section 108.9.3, "Enterprise Information System \(EIS\) Interactions"](#)) can make full use of TopLink queries (see [Chapter 108, "Introduction to TopLink Queries"](#)). However, you cannot use TopLink expressions with EIS: in an EIS project, interactions replace expressions.

---

---

## Creating an EIS Project

This chapter describes the various components that you must configure in order to create an EIS project.

This chapter includes the following sections:

- [Introduction to EIS Project Creation](#)
- [Creating an EIS Project with XML Records](#)
- [Creating an EIS Project with Indexed or Mapped Records](#)

For information on how to create more than one type of TopLink projects, see [Chapter 116, "Creating a Project"](#).

### 72.1 Introduction to EIS Project Creation

You can create a project using Oracle JDeveloper TopLink Editor, TopLink Workbench, or Java code.

Oracle recommends using either Oracle JDeveloper or TopLink Workbench to create projects and generate deployment XML, or Java source versions of the project for use at run time. For more information on how to create a project using TopLink Workbench, see [Section 116.1.2, "How to Create a Project Using TopLink Workbench"](#). For more information on how to create a project using Oracle JDeveloper, see [Section 116.1.1, "How to Create a Project Using Oracle JDeveloper"](#). For information on how to create a project using Java, see [Section 116.1.3, "How to Create a Project Using Java"](#).

For more information, see [Chapter 71, "Introduction to EIS Projects"](#).

For an EIS project that uses a record type other than XML, you must use Java code. For more information, see the following:

- [Section 72.3, "Creating an EIS Project with Indexed or Mapped Records"](#)
- [Section 116.1.3, "How to Create a Project Using Java"](#)
- *Oracle Fusion Middleware Java API Reference for Oracle TopLink.*

### 72.2 Creating an EIS Project with XML Records

TopLink Workbench provides complete support for creating EIS projects that map Java objects to EIS XML records.

Using TopLink Workbench, you can create an EIS project for transactional persistence of Java objects to a non-relational data source accessed using a JCA adapter and EIS XML records.

The TopLink runtime performs XML data conversions based on one or more XML schemas. In an EIS project, TopLink Workbench does not directly reference schemas in the deployment XML, but instead exports mappings configured in accordance to specific schemas.

EIS queries use `XMLInteraction`. For more information, see [Section 109.8, "Using EIS Interactions"](#).

### **72.2.1 How to Create an EIS Project with XML Records Using Oracle JDeveloper**

Refer to [Section 116.1.1, "How to Create a Project Using Oracle JDeveloper"](#) for this information.

### **72.2.2 How to Create an EIS Project with XML Records Using TopLink Workbench**

Refer to [Section 116.1.2, "How to Create a Project Using TopLink Workbench"](#) for this information.

## **72.3 Creating an EIS Project with Indexed or Mapped Records**

TopLink Workbench does not currently support non-XML EIS projects. You must create such an EIS project in Java.

Using Java, you can create an EIS project for transactional persistence of Java objects to a nonrelational data source accessed using a JCA adapter and any supported EIS record type including indexed, mapped, or XML records.

If you use XML records, the TopLink runtime performs XML data conversion based on one or more XML schemas. When you create an EIS project in Java, you configure mappings with respect to these schemas, but the TopLink runtime does not directly reference them.

You can base queries on any supported EIS interaction: `IndexedInteraction`, `MappedInteraction` (including `QueryStringInteraction`), or `XMLInteraction` (including `XQueryInteraction`). For more information, see [Section 109.8, "Using EIS Interactions"](#).

### **72.3.1 How to Create an EIS Project with Indexed or Mapped Records Using Java**

Refer to [Section 116.1.3, "How to Create a Project Using Java"](#) for this information.

## Configuring an EIS Project

This chapter describes the various components that you must configure in order to use an EIS project.

This chapter includes the following sections:

- [Introduction to EIS Project Configuration](#)
- [Configuring EIS Data Source Platform at the Project Level](#)
- [Configuring EIS Connection Specification Options at the Project Level](#)

For information on how to configure TopLink project options common to two or more project types, see [Chapter 117, "Configuring a Project"](#).

### 73.1 Introduction to EIS Project Configuration

lists the configurable options for EIS projects.

**Table 73–1 Configurable Options for EIS Projects**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Project save location (see <a href="#">Section 117.2, "Configuring Project Save Location"</a> )		✓	
Persistence type (see <a href="#">Section 117.5, "Configuring Persistence Type"</a> )	✓	✓	
Project classpath (see <a href="#">Section 117.3, "Configuring Project Classpath"</a> )		✓	
Comments (see <a href="#">Section 117.14, "Configuring Project Comments"</a> )	✓	✓	
Method or direct field access (see <a href="#">Section 117.4, "Configuring Method or Direct Field Access at the Project Level"</a> )	✓	✓	
Default descriptor advanced properties (see <a href="#">Section 117.6, "Configuring Default Descriptor Advanced Properties"</a> )	✓	✓	
Existence checking (see <a href="#">Section 117.7, "Configuring Existence Checking at the Project Level"</a> )	✓	✓	✓
Project deployment XML options (see <a href="#">Section 117.8, "Configuring Project Deployment XML Options"</a> )		✓	
Model Java source code options (see <a href="#">Section 117.9, "Configuring Model Java Source Code Options"</a> )	✓	✓	
EIS data source platform (see <a href="#">Section 73.2, "Configuring EIS Data Source Platform at the Project Level"</a> )		✓	✓
EIS connection specification options (see <a href="#">Section 73.3, "Configuring EIS Connection Specification Options at the Project Level"</a> )		✓	✓

**Table 73–1 (Cont.) Configurable Options for EIS Projects**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
XML parser platform (see <a href="#">Section 8.2.2.1, "Configuring XML Parser Platform"</a> )			✓
Importing an XML schema (see <a href="#">Section 5.6.3, "How to Import an XML Schema"</a> )	✓	✓	
XML schema namespace (see <a href="#">Section 5.6.5, "How to Configure XML Schema Namespace"</a> )	✓	✓	✓
Cache type and size (see <a href="#">Section 117.10, "Configuring Cache Type and Size at the Project Level"</a> )	✓	✓	✓
Cache isolation (see <a href="#">Section 117.11, "Configuring Cache Isolation at the Project Level"</a> )	✓	✓	✓
Cache coordination change propagation (see <a href="#">Section 117.12, "Configuring Cache Coordination Change Propagation at the Project Level"</a> )	✓	✓	✓
Cache expiration (see <a href="#">Section 117.13, "Configuring Cache Expiration at the Project Level"</a> )	✓	✓	

For more information, see [Chapter 71, "Introduction to EIS Projects"](#).

## 73.2 Configuring EIS Data Source Platform at the Project Level

For each EIS project, you must specify one of the following JCA data source platforms that you will be using:

- Oracle AQ
- Attunity Connect
- IBM MQSeries
- JMS
- Sun Black Box
- XML File

This platform configuration is overridden by the session login, if configured.

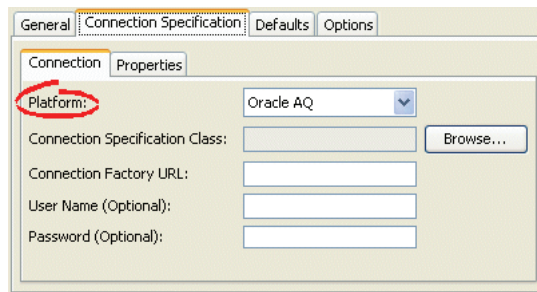
For more information, see the following:

- [Section 99.2, "Configuring an EIS Data Source Platform at the Session Level"](#)
- [Section 96.1.3, "Data Source Platform Types"](#)

### 73.2.1 How to Configure EIS Data Source Platform at the Project Level Using TopLink Workbench

To specify the data source platform of an EIS project, use this procedure:

1. Select an EIS project object in the **Navigator**.
2. Select the **Connection Specifications** tab in the **Editor**. The Connection Specifications tab appears.
3. Select the **Connection** tab. The Connection tab appears.

**Figure 73–1 Connection Tab, Platform Option**

Select the EIS platform for this project from the list of options. For more information, see [Section 96.1.3, "Data Source Platform Types"](#).

## 73.3 Configuring EIS Connection Specification Options at the Project Level

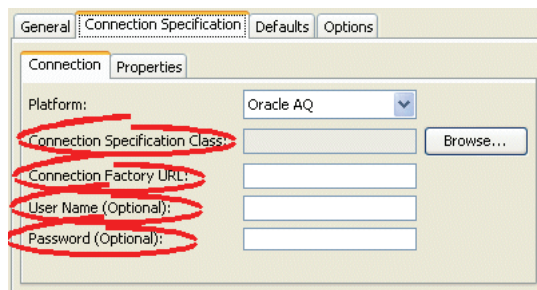
You can configure connection information at the project level for an EIS application. This information is stored in the `project.xml` file. The Oracle TopLink runtime uses this information as its deployment login: whenever your EIS application performs a persistence operation when deployed in a Java EE application server.

This connection configuration is overridden by the connection information at the session level, if configured. For more information about session level configuration, see [Section 99.3, "Configuring EIS Connection Specification Options at the Session Level"](#).

### 73.3.1 How to Configure EIS Connection Specification Options at the Project Level Using TopLink Workbench

To specify the connection information for an EIS project, use this procedure.

1. Select an EIS project object in the **Navigator**.
2. Select the **Connection Specifications** tab in the **Editor**. The Connection Specifications tab appears.
3. Select the **Connection** tab. The Connection tab appears.

**Figure 73–2 Connection Tab, Connection Specification Options**

Use this table to enter data in the following fields to configure the connection specification options:

Field	Description
<b>Connection Specification Class</b>	<p>Specify the appropriate connection specification class for the selected <b>Platform</b>. Click <b>Browse</b> to choose from all the classes in the TopLink class path. (example: if <b>Platform</b> is <code>oracle.toplink.eis.aq.AQPlatform</code>, use <code>oracle.toplink.eis.aq.AQEISConnectionSpec</code>).</p> <p>For more information on platform configuration, see <a href="#">Section 99.2, "Configuring an EIS Data Source Platform at the Session Level"</a>.</p>
<b>Connection Factory URL</b>	<p>Specify the appropriate connection factory URL (as a Java EE JNDI name) for the selected <b>Connection Specification Class</b> (example: <code>java:comp/env/eis/attunity</code>).</p>
<b>Username</b>	<p>Specify the name required to log in to the data source.</p>
<b>Password</b>	<p>Specify the password required to log in to the data source.</p> <p><b>Note:</b> When exporting Java source and deployment XML (see <a href="#">Section 116.3, "Exporting Project Information"</a>), TopLink Workbench writes the database password (if applicable) using JCE encryption (when using JDK 1.4). For information on how to specify password encryption options, see <a href="#">Section 97.3, "Configuring Password Encryption"</a>.</p>



# Part XVIII

---

## EIS Descriptors

This part contains general information about EIS descriptors, as well as detailed information on how to create and configure these descriptors.

This part contains the following chapters:

- [Chapter 74, "Introduction to EIS Descriptors"](#)  
This chapter introduces concepts to an EIS descriptor.
- [Chapter 75, "Creating an EIS Descriptor"](#)  
This chapter explains how to create descriptor options specific to an EIS descriptor.
- [Chapter 76, "Configuring an EIS Descriptor"](#)  
This chapter explains how to configure descriptor options specific to an EIS descriptor.



---

---

## Introduction to EIS Descriptors

This chapter introduces options specific to an EIS descriptor.

This chapter includes the following sections:

- [EIS Descriptor Concepts](#)
- [EIS Descriptors and Aggregation](#)
- [EIS Descriptors and Inheritance](#)

For information on descriptor concepts and features common to more than one type of TopLink descriptors, see [Chapter 16, "Introduction to Descriptors"](#).

### 74.1 EIS Descriptor Concepts

EIS descriptors describe Java objects that you map to an EIS data source by way of a JCA adapter.

Using EIS descriptors in an EIS project created with TopLink Workbench, you can configure EIS mappings (see [Section 77.1, "EIS Mapping Types"](#)) to XML records.

Using EIS descriptors in an EIS project that you create in Java, you can configure EIS mappings to any supported EIS record type: XML, mapped, or indexed.

See [Part XXVII, "Creation and Configuration of Descriptors"](#) for information on how to create and configure descriptors regardless of their type.

For information specific to creation and configuration of EIS descriptors, see the following:

- [Chapter 75, "Creating an EIS Descriptor"](#)
- [Chapter 76, "Configuring an EIS Descriptor"](#)

### 74.2 EIS Descriptors and Aggregation

When working with descriptors for a parent (source) and a child (target) objects, you have to accomplish the following:

- if the source object exists, then you must ensure that the target object also exists;
- if the source object is destroyed, then you must ensure that the target object is also destroyed.

For more information, see [Section 16.2.5, "Descriptors and Aggregation"](#).

In your EIS project, designate the descriptors for the source and target objects to reflect this relationship as [Root and Composite Descriptors in EIS Projects](#).

## 74.2.1 Root and Composite Descriptors in EIS Projects

In an EIS project, you can designate the descriptor as a composite (see [Section 75.2.1.2, "EIS Composite Descriptors"](#)).

The type of EIS mapping you wish to create will determine whether you configure an EIS descriptor as a composite or root (see [Section 77.2.6, "Composite and Reference EIS Mappings"](#)).

For more information, see [Section 76.6, "Configuring an EIS Descriptor as a Root or Composite Type"](#).

You cannot configure EJB information for an EIS descriptor designated as an composite (see [Section 16.2.3, "Descriptors and CMP and BMP"](#)).

You can configure inheritance for an EIS descriptor designated as a composite (see [Section 16.2.2, "Descriptors and Inheritance"](#)), however, in this case, *all* the descriptors in the inheritance tree must be composites. Composite and root descriptors cannot exist in the same inheritance tree.

## 74.3 EIS Descriptors and Inheritance

You can use descriptors to describe the inheritance relationships between classes in your EIS project. For more information, see the following:

- [Section 16.2.2, "Descriptors and Inheritance"](#)
- [Section 119.20, "Configuring Inheritance for a Child \(Branch or Leaf\) Class Descriptor"](#)
- [Section 119.21, "Configuring Inheritance for a Parent \(Root\) Descriptor"](#)
- [Section 119.22, "Configuring Inheritance Expressions for a Parent \(Root\) Class Descriptor"](#)
- [Section 119.23, "Configuring Inherited Attribute Mapping in a Subclass"](#)

### 74.3.1 Inheritance and Primary Keys in EIS Projects

For EIS projects, TopLink assumes that all of the classes in an inheritance hierarchy have the same primary key, as set in the root descriptor. Child descriptors associated with data source representations that have different primary keys must define the mapping between the root primary key and the local one.

For more information, see [Section 119.2, "Configuring Primary Keys"](#).

---

---

## Creating an EIS Descriptor

This chapter explains how to create descriptor options specific to an EIS descriptor.

This chapter includes the following sections:

- [Introduction to EIS Descriptor Creation](#)
- [Creating an EIS Descriptor](#)

For information on how to create more than one type of descriptors, see [Chapter 118](#), "Creating a Descriptor".

### 75.1 Introduction to EIS Descriptor Creation

After you create a descriptor, you must configure its various options (see [Chapter 119](#), "Configuring a Descriptor") and use it to define mappings.

For complete information on the various types of mapping that TopLink supports, see [Chapter 17](#), "Introduction to Mappings" and [Chapter 120](#), "Creating a Mapping".

For complete information on the various types of descriptor that TopLink supports, see [Section 16.1](#), "Descriptor Types".

For more information, see the following:

- [Chapter 16](#), "Introduction to Descriptors"
- [Chapter 74](#), "Introduction to EIS Descriptors"

### 75.2 Creating an EIS Descriptor

You can create an EIS descriptor using Oracle JDeveloper TopLink Editor, TopLink Workbench (see [Section 75.2.1](#), "How to Create an EIS Descriptor Using TopLink Workbench"), or Java code (see [Section 75.2.2](#), "How to Create an EIS Descriptor Using Java"). Oracle recommends that you use either Oracle JDeveloper or TopLink Workbench to create and manage your EIS descriptors.

#### 75.2.1 How to Create an EIS Descriptor Using TopLink Workbench

Using TopLink Workbench, you can create the following types of EIS descriptor in an EIS project:

- [EIS Root Descriptors](#)
- [EIS Composite Descriptors](#)

##### 75.2.1.1 EIS Root Descriptors



You can modify an EIS descriptor's behavior by configuring it as a root EIS descriptor (see [Section 76.6, "Configuring an EIS Descriptor as a Root or Composite Type"](#)). When you designate an EIS descriptor as a root, you tell the TopLink runtime that the EIS descriptor's reference class is a parent class: no other class will reference it by way of a composite object mapping or composite collection mapping. Using an EIS root descriptor, you can configure all supported mappings. You can also configure an EIS root descriptor with EIS interactions (see [Section 109.8, "Using EIS Interactions"](#)). However, if you configure the EIS root descriptor with a composite object mapping or composite collection mapping, the reference descriptor you define must be an EIS composite descriptor; it cannot be another EIS root descriptor.



### 75.2.1.2 EIS Composite Descriptors

By default, when you add a class to an EIS project (see [Section 117.3, "Configuring Project Classpath"](#)), TopLink Workbench creates an EIS descriptor for the class, and designates the EIS descriptor as a composite. When you designate an EIS descriptor as a composite, you tell the TopLink runtime that the EIS descriptor's reference class may be referenced by a composite object mapping or composite collection mapping. Using an EIS composite descriptor, you can configure all supported mappings. However, you cannot configure an EIS composite descriptor with EIS interactions: for this, you need an EIS root descriptor (see [Section 75.2.1.1, "EIS Root Descriptors"](#)).

## 75.2.2 How to Create an EIS Descriptor Using Java

[Example 75–1](#) shows how to create a relational descriptor using Java code.

### **Example 75–1** *Creating an EIS Descriptor in Java*

```
EISDescriptor descriptor = new EISDescriptor();  
descriptor.setJavaClass(YourClass.class);
```

To designate an EIS descriptor as a composite, use `ClassDescriptor` method `descriptorIsAggregate`.

## Configuring an EIS Descriptor

This chapter describes the various components that you must configure in order to use an EIS descriptor.

This chapter includes the following sections:

- [Introduction to EIS Descriptor Configuration](#)
- [Configuring Schema Context for an EIS Descriptor](#)
- [Configuring Default Root Element](#)
- [Configuring Record Format](#)
- [Configuring Custom EIS Interactions for Basic Persistence Operations](#)

For information on how to configure descriptor options common to two or more descriptor types, see [Chapter 119, "Configuring a Descriptor"](#).

### 76.1 Introduction to EIS Descriptor Configuration

[Table 76–1](#) lists the default configurable options for an EIS descriptor.

**Table 76–1** Configurable Options for EIS Descriptor

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
XML schema namespace (see <a href="#">Section 5.6.5, "How to Configure XML Schema Namespace"</a> )	✓	✓	✓
XML schema reference (see <a href="#">Section 5.6.4, "How to Configure an XML Schema Reference"</a> )	✓	✓	✓
Schema context (see <a href="#">Section 76.2, "Configuring Schema Context for an EIS Descriptor"</a> )	✓	✓	✓
Default root element (see <a href="#">Section 76.3, "Configuring Default Root Element"</a> )	✓	✓	
Primary keys (see <a href="#">Section 119.2, "Configuring Primary Keys"</a> ) <sup>1</sup>	✓	✓	✓
Read-only (see <a href="#">Section 119.3, "Configuring Read-Only Descriptors"</a> ) <sup>1</sup>	✓	✓	✓
Unit of work conforming (see <a href="#">Section 119.4, "Configuring Unit of Work Conforming at the Descriptor Level"</a> ) <sup>1</sup>	✓	✓	✓
Alias (see <a href="#">Section 119.5, "Configuring Descriptor Alias"</a> )	✓	✓	
Comments (see <a href="#">Section 119.6, "Configuring Descriptor Comments"</a> )	✓	✓	
Record format (see <a href="#">Section 76.4, "Configuring Record Format"</a> )			✓
Creating classes (see <a href="#">Section 5.7.1, "How to Create Classes"</a> )	✓	✓	

**Table 76–1 (Cont.) Configurable Options for EIS Descriptor**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Named queries (see Section 119.7, "Configuring Named Queries at the Descriptor Level") <sup>1</sup>	✓	✓	✓
Custom EIS interactions for basic persistence operations (see Section 76.5, "Configuring Custom EIS Interactions for Basic Persistence Operations") <sup>1</sup>		✓	✓
Cache refreshing (see Section 119.9, "Configuring Cache Refreshing") <sup>1</sup>	✓	✓	✓
Cache type and size (see Section 119.12, "Configuring Cache Type and Size at the Descriptor Level") <sup>1</sup>	✓	✓	✓
Cache isolation (see Section 119.13, "Configuring Cache Isolation at the Descriptor Level")	✓	✓	✓
Cache coordination change propagation (see Section 119.15, "Configuring Cache Coordination Change Propagation at the Descriptor Level")	✓	✓	✓
Cache expiration (see Section 119.16, "Configuring Cache Expiration at the Descriptor Level")	✓	✓	✓
Cache existence checking (see Section 119.17, "Configuring Cache Existence Checking at the Descriptor Level")	✓	✓	✓
EJB information (see Section 119.18, "Configuring a Descriptor with EJB CMP and BMP Information")	✓	✓	✓
EIS descriptor as a root or composite type (see Section 76.6, "Configuring an EIS Descriptor as a Root or Composite Type")		✓	✓
Inheritance for a child class descriptor (see Section 119.20, "Configuring Inheritance for a Child (Branch or Leaf) Class Descriptor")	✓	✓	✓
Inheritance for a parent class descriptor (see Section 119.21, "Configuring Inheritance for a Parent (Root) Descriptor")	✓	✓	✓
Inherited attribute mapping in a subclass (see Section 119.23, "Configuring Inherited Attribute Mapping in a Subclass")	✓	✓	✓
Domain object method as an event handler (see Section 119.24, "Configuring a Domain Object Method as an Event Handler")	✓	✓	✓
Descriptor event listener as an event handler (see Section 119.25, "Configuring a Descriptor Event Listener as an Event Handler")			✓
Locking policy (see Section 119.26, "Configuring Locking Policy") <sup>1</sup>	✓	✓	✓
Returning policy (see Section 119.27, "Configuring Returning Policy")	✓	✓	✓
Instantiation policy (see Section 119.28, "Configuring Instantiation Policy")	✓	✓	✓
Copy policy (see Section 119.29, "Configuring Copy Policy")	✓	✓	✓
Change policy (see Section 119.30, "Configuring Change Policy")			✓
Wrapper policy (see Section 119.32, "Configuring Wrapper Policy")			✓
Amendment methods (see Section 119.35, "Configuring Amendment Methods")	✓	✓	✓
Mapping (see Section 121, "Configuring a Mapping")	✓	✓	✓

<sup>1</sup> EIS root descriptors only (see Section 76.6, "Configuring an EIS Descriptor as a Root or Composite Type").

For more information, see Chapter 74, "Introduction to EIS Descriptors".



## 76.2 Configuring Schema Context for an EIS Descriptor

TopLink Workbench uses the schema context to associate the class that the EIS descriptor describes with a simple or complex type in one of the schemas associated with the EIS project (see [Section 5.6.4, "How to Configure an XML Schema Reference"](#)). This allows TopLink Workbench to display the appropriate attributes available for mapping in that context.

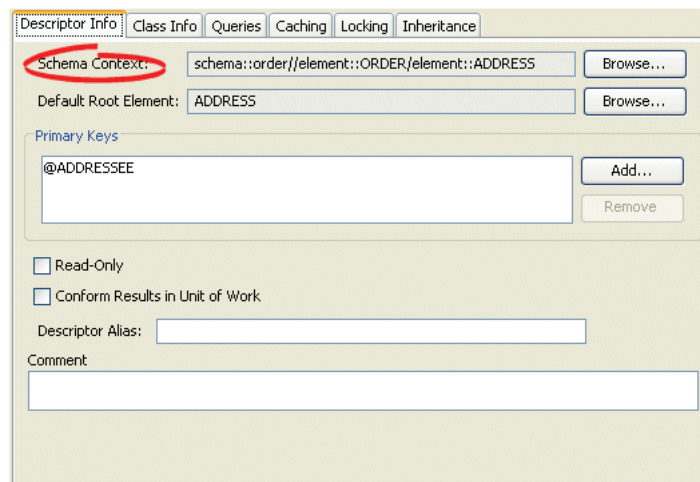
You must configure the schema context for an EIS root descriptor (see [Section 76.6, "Configuring an EIS Descriptor as a Root or Composite Type"](#)) only if you are using TopLink Workbench or Oracle JDeveloper TopLink Editor.

### 76.2.1 How to Configure Schema Context for an EIS Descriptor Using TopLink Workbench

To associate an EIS descriptor with a simple or complex type in this project's schema, use this procedure:

1. Select an EIS descriptor in the **Navigator**. Its properties appear in the Editor.
2. Click the **Descriptor Info** tab. The Descriptor Info tab appears.

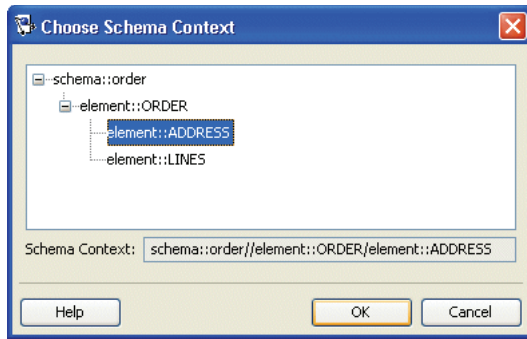
**Figure 76–1 Descriptor Info Tab, Schema Context Option**



Click **Browse** to select the schema element to associate with this descriptor. For more information, see [Section 76.2.1.1, "Choosing a Schema Context"](#).

#### 76.2.1.1 Choosing a Schema Context

Use the Choose Schema Context dialog box to select a specific schema element (such as when mapping an element).

**Figure 76–2 Choose Schema Context Dialog Box**

Select the schema element and click **OK**.

## 76.2.2 How to Configure Schema Context for an EIS Descriptor Using Java

For an EIS descriptor, the TopLink runtime does not need the schema context: the runtime can determine the schema context based on the mappings you configure on the descriptor. No further configuration is required.

## 76.3 Configuring Default Root Element

You must configure the default root element for an EIS root descriptor (see [Section 75.2.1.1, "EIS Root Descriptors"](#)) so that the TopLink runtime knows the data source data type associated with the class the descriptor describes. Descriptors used only in composite relationship mappings do not require a default root element.

---

---

**Note:** Although you select an element from your project's schema to configure this attribute, you are choosing the element's simple or complex type.

---

---

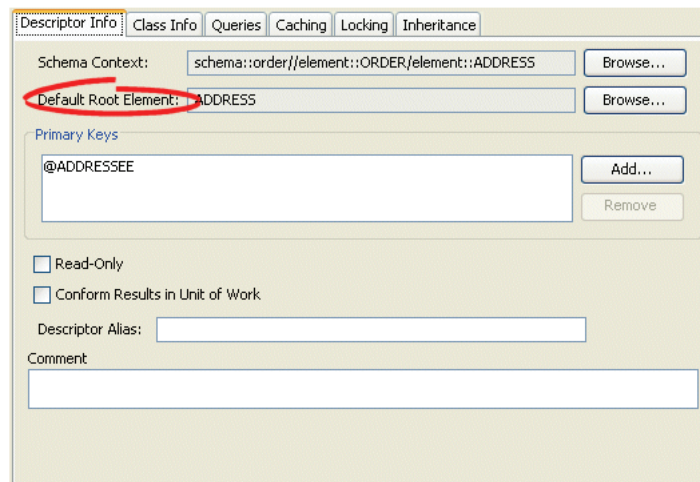
For more information, see [Section 16.2.12, "Default Root Element"](#).

### 76.3.1 How to Configure Default Root Element Using TopLink Workbench

When you create an EIS project using TopLink Workbench, you must use XML records. Consequently, you must configure a default root element so that TopLink Workbench knows what element to start with when persisting an instance of the class that the EIS descriptor describes.

To specify a schema element as the default root element for the descriptor, use this procedure:

1. Select a descriptor in the **Navigator**. Its properties appear in the Editor.
2. Click the **Descriptor Info** tab. The Descriptor Info tab appears.

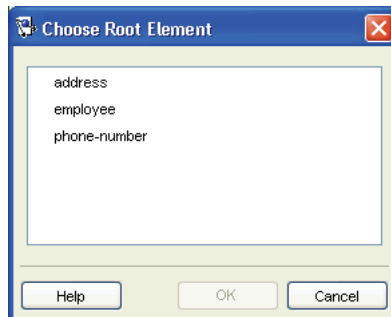
**Figure 76–3 Descriptor Info Tab, Default Root Element Option**

Use the **Default Root Element** option to select the root element for this descriptor.

Click **Browse** to select the schema element to identify as the root element. See [Section 76.3.1.1, "Choosing a Root Element"](#) for more information.

### 76.3.1.1 Choosing a Root Element

Use the Choose Root Element dialog box to select a specific root element.

**Figure 76–4 Choose Root Element Dialog Box**

Select the root element and click **OK**.

## 76.3.2 How to Configure Default Root Element Using Java

When you create an EIS project using Java code, use the EISDescriptor method `setDataTypeName` to specify the XML schema complex type name (if you are using XML records) or the JCA record name (if you are using indexed or mapped records) corresponding to the class that the EIS descriptor describes. For more information, see Oracle Fusion Middleware Java API Reference for Oracle TopLink.

## 76.4 Configuring Record Format

The EIS descriptor record format determines the EIS record type to which the descriptor's EIS mappings map.

When you create an EIS project using TopLink Workbench, TopLink configures all EIS descriptors with a record format of XML.

When you create an EIS project in Java, you can configure the EIS descriptor record type to any of the supported types, as [Table 76–2](#) shows.

**Table 76–2 EIS Record Formats**

EISDescriptor Method	EIS Record Type
<code>useMappedRecordFormat</code>	All EIS mappings owned by this descriptor map to EIS mapped records.
<code>useIndexedRecordFormat</code>	All EIS mappings owned by this descriptor map to EIS indexed records.
<code>useXMLRecordFormat</code>	<p>All EIS mappings owned by this descriptor map to EIS XML records.</p> <p>If you use the XML record format, you must specify one or more XML schemas in your EIS project (see <a href="#">Section 5.6.3, "How to Import an XML Schema"</a>). The TopLink runtime performs XML data conversion based on one or more XML schemas. In an EIS XML project, TopLink Workbench does not directly reference schemas in the deployment XML, but instead exports mappings configured with respect to the schemas you specify.</p> <p>For information on TopLink support for XML namespaces, see <a href="#">Section 15.2.7, "XML Namespaces"</a>.</p>

For more information, see [Section 77.2.1, "EIS Record Type"](#).

### 76.4.1 How to Configure Record Format Using Java

To configure the EIS record format for an EIS descriptor, use one of the `EISDescriptor` methods listed in [Table 76–2](#), as shown in [Example 76–1](#).

**Example 76–1 Configuring EISDescriptor Record Format**

```
EISDescriptor descriptor = new EISDescriptor();
descriptor.useIndexedRecordFormat();
```

## 76.5 Configuring Custom EIS Interactions for Basic Persistence Operations

You can use TopLink to define an interaction for each basic persistence operation (**insert**, **update**, **delete**, **read object**, **read all**, or **does exist**) so that when you query and modify your EIS-mapped objects, the TopLink runtime will use the appropriate EIS interaction instead of the default EIS interaction.

You can configure custom EIS interactions for basic persistence operations only for EIS descriptors designated as root descriptors ([Section 76.6, "Configuring an EIS Descriptor as a Root or Composite Type"](#)).

For CMP projects, the `ejb-jar.xml` file stores query lists. You can define the queries in the file and then read them into TopLink Workbench, or define them on the Queries tab and write them to the file. For more information, see [Section 19.7.1, "How to Write to the ejb-jar.xml File Using TopLink Workbench"](#) and [Section 19.7.2, "How to Read from the ejb-jar.xml File Using TopLink Workbench"](#).

Using TopLink Workbench, you can create `XMLInteraction` objects, in which there is a single query per interaction (see [Section 76.5.1, "How to Configure Custom EIS Interactions for Basic Persistence Operations Using TopLink Workbench"](#)).

Using Java, you can create any `EISInteraction` type. For some EIS projects, it is common for multiple interactions to be used in a single query. For example, one interaction—to enqueue a request, and another—to dequeue the response. Because TopLink Workbench does not support setting multiple interactions on a single query, you must use an amendment method to create and configure the interaction in Java (see [Section 76.5.2, "How to Configure Custom EIS Interactions for Basic Persistence Operations Using Java"](#)).

---

**Note:** In a one-to-one or one-to-many EIS mapping, you must also specify a selection interaction that TopLink uses to acquire target objects. You can use either the target object's read interaction (the default) or specify a separate selection interaction, if necessary. For more information, see [Section 78.4, "Configuring Selection Interaction"](#).

---

### 76.5.1 How to Configure Custom EIS Interactions for Basic Persistence Operations Using TopLink Workbench

To configure custom EIS interactions for basic persistence operations, use the following procedure:

1. In the **Navigator**, select an EIS root descriptor in a EIS project.
2. Click the **Queries** tab in the **Editor**. The Queries tab appears.
3. Click the **Custom Calls** tab. The Custom Calls tab appears.

**Figure 76–5** *Queries, Custom Calls Tab for EIS Calls*

Click the appropriate interaction type from the list (**Insert**, **Update**, **Delete**, **Read Object**, **Read All**, or **Does Exist**) and use the following table to enter data in each field

Field	Description
<b>Interaction Type</b>	Using TopLink Workbench, you can only use XML Interactions. You cannot change this field.
<b>Function Name</b>	The name of the EIS function that this call type (Read Object or Read All) invokes on the EIS.
<b>Input Record Name</b>	The name passed to the JCA adapter when creating the input record.
<b>Input Root Element</b>	The root element name to use for the input DOM.
<b>Input Arguments</b>	The query argument name to map to the interaction field or XPath nodes in the argument record.  For example, if you are using XML records, use this option to map input argument name to the XPath <code>name/first-name</code> .
<b>Output Arguments</b>	The result record field or XPath nodes to map to the correct nodes in the record used by the descriptor's mappings.  For example, if you are using XML records, use this option to map the output <code>fname</code> to <code>name/first-name</code> .  Output arguments are not required if the interaction returns an XML result that matches the descriptor's mappings.
<b>Input Result Path</b>	Use this option if the EIS interaction expects the interaction arguments to be nested in the XML record.  For example, specify <code>arguments</code> , if the arguments were to be nested under the root element <code>exec-find-order</code> , then under an <code>arguments</code> element.
<b>Output Result Path</b>	Use this option if the EIS interaction result record contains the XML data that maps to the objects in a nested structure.  For example, specify <code>order</code> , if the results were return under a root element <code>results</code> , then under an <code>order</code> element.
<b>Properties</b>	Any properties required by your EIS platform. For example, property name <code>operation</code> (from <code>AQPlatform.QUEUE_OPERATION</code> ) and property value <code>enqueue</code> (from <code>AQPlatform.ENQUEUE</code> ).

## 76.5.2 How to Configure Custom EIS Interactions for Basic Persistence Operations Using Java

Using Java, you can create any type of EIS interaction that TopLink supports (see [Section 109.8, "Using EIS Interactions"](#)).

For some EIS projects, it is common for multiple interactions to be used in a single query: for example, one interaction to enqueue a request and another to dequeue the response. Because TopLink Workbench does not support setting multiple interactions on a single query, you must use an amendment method to create and configure the interaction in Java, as [Example 76–2](#) shows.

### **Example 76–2** Creating an XML Interaction for an AQ Platform

```
public static void addXMLInteractions(ClassDescriptor descriptor) {
    // find order interaction
    XMLInteraction request = new XMLInteraction();
    request.setProperty(AQPlatform.QUEUE_OPERATION, AQPlatform.ENQUEUE);
    request.setProperty(AQPlatform.QUEUE, "ORDER_INBOUND_QUEUE");
    request.setProperty(AQPlatform.SCHEMA, "AQUER");
    request.setInputRootElementName("READ_ORDER");
}
```

```

request.addArgument("@id");

XMLInteraction response = new XMLInteraction();
response.setProperty(AQPlatform.QUEUE_OPERATION, AQPlatform.DEQUEUE);
response.setProperty(AQPlatform.QUEUE, "ORDER_OUTBOUND_QUEUE");
response.setProperty(AQPlatform.SCHEMA, "AUSER");

ReadObjectQuery query = new ReadObjectQuery();
query.addCall(request);
query.addCall(response);
descriptor.getQueryManager().setReadObjectQuery(query);

// place order interaction
XMLInteraction insert = new XMLInteraction();
insert.setProperty(AQPlatform.QUEUE_OPERATION, AQPlatform.ENQUEUE);
insert.setProperty(AQPlatform.QUEUE, "ORDER_INBOUND_QUEUE");
insert.setProperty(AQPlatform.SCHEMA, "AUSER");
insert.setInputRootElementName("INSERT_ORDER");

descriptor.getQueryManager().setInsertCall(insert);
}

```

## 76.6 Configuring an EIS Descriptor as a Root or Composite Type

You can designate an EIS descriptor as root (see [Section 75.2.1.1, "EIS Root Descriptors"](#)) or composite (see [Section 75.2.1.2, "EIS Composite Descriptors"](#)).

When you designate an EIS descriptor as a root, you tell the TopLink runtime that the EIS descriptor's reference class is a parent class—no other class will reference it by way of a composite object mapping or composite collection mapping. Using an EIS root descriptor, you can configure all supported mappings and you can configure the descriptor with EIS interactions (see [Section 109.8, "Using EIS Interactions"](#)). However, if you configure the EIS root descriptor with a composite object mapping or composite collection mapping, the reference descriptor you define must be an EIS composite descriptor; it cannot be another EIS root descriptor.

When you designate an EIS descriptor as a composite (the default), you tell the TopLink runtime that the EIS descriptor's reference class may be referenced by a composite object or composite collection mapping (see [Chapter 81, "Configuring an EIS Composite Object Mapping"](#) and [Chapter 82, "Configuring an EIS Composite Collection Mapping"](#)). Using an EIS composite descriptor, you can configure all supported mappings, but you cannot configure it with EIS interactions.

You can configure inheritance for a descriptor designated as a composite (see [Section 119.20, "Configuring Inheritance for a Child \(Branch or Leaf\) Class Descriptor"](#)), however, in this case, *all* the descriptors in the inheritance tree must be aggregates. Aggregate and class descriptors cannot exist in the same inheritance tree. For more information, see [Section 16.3.4, "Aggregate and Composite Descriptors and Inheritance"](#).

If you configure a descriptor as a composite using TopLink Workbench, you cannot configure the descriptor with EJB information (see [Section 119.18, "Configuring a Descriptor with EJB CMP and BMP Information"](#)).

For more information, see the following:

- [Section 50.1.1, "XML Descriptors and Aggregation"](#)
- [Section 77.2.6, "Composite and Reference EIS Mappings"](#)

## 76.6.1 How to Configure an EIS Descriptor as a Root or Composite Type Using TopLink Workbench

To configure an EIS descriptor as a root or composite EIS descriptor, use this procedure:

1. In the **Navigator**, select an EIS composite descriptor.
2. Click the **Root** or **Composite** descriptor button on the mapping toolbar.

You can also select the descriptor and choose **Selected > Descriptor Type > Root** or **Composite** from the menu or by right-clicking on the descriptor in the **Navigator** and selecting **Descriptor Type > Root** or **Composite** from the context menu.

## 76.6.2 How to Configure an EIS Descriptor as a Root or Composite Type Using Java

To configure an EIS descriptor as root or composite using Java, create a descriptor amendment method (see [Section 119.35, "Configuring Amendment Methods"](#)) and use the following `EISDescriptor` methods:

- To designate an EIS descriptor as a root descriptor, use `EISDescriptor` method `descriptorIsNormal`.
- To designate an EIS descriptor as a composite (nonroot) descriptor, use `EISDescriptor` method `descriptorIsAggregate`.



# Part XIX

---

## EIS Mappings

TopLink enterprise information system (EIS) mappings provide support for accessing legacy data sources and enterprise applications through Java EE Connector architecture (JCA) adapter. TopLink EIS mappings use the JCA Common Client Interface (CCI) to access the EIS through its resource adapter. This provides the ability to directly map from an existing Java object model to any transactional data source, such as mainframes with flat file/hierarchical data. An EIS mapping transforms object data members to the EIS record format defined by the object's descriptor.

This part contains the following chapters:

- [Chapter 77, "Introduction to EIS Mappings"](#)  
This chapter describes each of the different TopLink EIS mapping types and important EIS mapping concepts.
- [Chapter 78, "Configuring an EIS Mapping"](#)  
This chapter explains how to configure TopLink EIS mapping options common to two or more EIS mapping types.
- [Chapter 79, "Configuring an EIS Direct Mapping"](#)  
This chapter explains how to configure a direct EIS mapping.
- [Chapter 80, "Configuring an EIS Composite Direct Collection Mapping"](#)  
This chapter explains how to configure a direct collection EIS mapping.
- [Chapter 81, "Configuring an EIS Composite Object Mapping"](#)  
This chapter explains how to configure a composite object EIS mapping.
- [Chapter 82, "Configuring an EIS Composite Collection Mapping"](#)  
This chapter explains how to configure a composite collection EIS mapping.
- [Chapter 83, "Configuring an EIS One-to-One Mapping"](#)  
This chapter explains how to configure a one-to-one EIS mapping.
- [Chapter 84, "Configuring an EIS One-to-Many Mapping"](#)  
This chapter explains how to configure a one-to-many EIS mapping.
- [Chapter 85, "Configuring an EIS Transformation Mapping"](#)  
This chapter explains how to configure a transformation EIS mapping.



---

---

## Introduction to EIS Mappings

TopLink enterprise information system (EIS) mappings provide support for accessing legacy data sources and enterprise applications through Java EE Connector architecture (JCA) adapter. TopLink EIS mappings use the JCA Common Client Interface (CCI) to access the EIS through its resource adapter. This provides the ability to directly map from an existing Java object model to any transactional data source, such as mainframes with flat file/hierarchical data.

An EIS mapping transforms object data members to the EIS record format defined by the object's descriptor.

This chapter includes the following sections:

- [EIS Mapping Types](#)
- [EIS Mapping Concepts](#)
- [EIS Direct Mapping](#)
- [EIS Composite Direct Collection Mapping](#)
- [EIS Composite Object Mapping](#)
- [EIS Composite Collection Mapping](#)
- [EIS One-to-One Mapping](#)
- [EIS One-to-Many Mapping](#)
- [EIS Transformation Mapping](#)

For information on mapping concepts and features common to more than one type of TopLink mappings, see [Chapter 17, "Introduction to Mappings"](#).

### 77.1 EIS Mapping Types

TopLink supports the EIS mappings listed in [Table 77-1](#).

**Table 77-1 TopLink Object EIS Mapping Types**

EIS Mapping Type	Description	Oracle JDeveloper	TopLink Workbench	Java
Direct mapping (see <a href="#">Section 77.3, "EIS Direct Mapping"</a> )	Map a simple object attribute directly to an EIS record.	✓	✓	✓
Composite direct collection mapping (see <a href="#">Section 77.4, "EIS Composite Direct Collection Mapping"</a> )	Map a collection of Java attributes directly to an EIS record.	✓	✓	✓
Composite object mapping (see <a href="#">Section 77.5, "EIS Composite Object Mapping"</a> )	Map a Java object to an EIS record in a privately owned one-to-one relationship. Composite object mappings represent a relationship between two classes.	✓	✓	✓
Composite collection mapping (see <a href="#">Section 77.6, "EIS Composite Collection Mapping"</a> )	Map a Map or Collection of Java objects to an EIS record in a privately owned one-to-many relationship.	✓	✓	✓
One-to-one mapping (see <a href="#">Section 77.7, "EIS One-to-One Mapping"</a> )	Define a reference mapping that represents the relationship between a single source object and a single mapped persistent Java object.	✓	✓	✓
One-to-many mapping (see <a href="#">Section 77.8, "EIS One-to-Many Mapping"</a> )	Define a reference mapping that represents the relationship between a single source object and a collection of mapped persistent Java objects.	✓	✓	✓
Transformation mapping (see <a href="#">Section 77.9, "EIS Transformation Mapping"</a> )	Create custom mappings where one or more EIS record fields can be used to create the object to be stored in a Java class's attribute.	✓	✓	✓

## 77.2 EIS Mapping Concepts

This section describes concepts unique to TopLink EIS mappings, including the following:

- [EIS Record Type](#)
- [XPath Support](#)
- [xsd:list and xsd:union Support](#)
- [jaxb:class Support](#)
- [Typesafe Enumeration Support](#)
- [Composite and Reference EIS Mappings](#)
- [EIS Mapping Architecture](#)

### 77.2.1 EIS Record Type

TopLink supports the following JCA EIS record types:

- [Indexed Records](#)
- [Mapped Records](#)
- [XML Records](#)

You configure the record type at the EIS descriptor level (see [Section 76.4, "Configuring Record Format"](#)). EIS mappings use the record type of their EIS descriptor to determine how to map Java attributes. That is, you use the same EIS mapping regardless of the record type, with which you configure an EIS descriptor.

---

---

**Note:** Not all JCA adapters support all record types. Consult your JCA adapter documentation for details.

---

---

### 77.2.1.1 Indexed Records

The `javax.resource.cci.IndexedRecord` represents an ordered collection of record elements based on the `java.util.List` interface.

The TopLink runtime maps Java objects to indexed record elements or subrecords of an indexed record depending on the type of EIS mapping you use (see [Section 77.2.6, "Composite and Reference EIS Mappings"](#)).

### 77.2.1.2 Mapped Records

The `javax.resource.cci.MappedRecord` represents a key-value map-based collection of record elements based on the `java.util.Map` interface.

The TopLink runtime maps Java objects to mapped record elements or subrecords of a mapped record depending on the type of EIS mapping you use (see [Section 77.2.6, "Composite and Reference EIS Mappings"](#)).

### 77.2.1.3 XML Records

An XML record represents a `javax.resource.cci.Record` as an XML schema (XSD)-based XML document. Not all JCA adapters support XML records.

The TopLink runtime maps Java objects to XML documents according to your XSD and the behavior defined for XML mappings.

For more information, see [Chapter 53, "Introduction to XML Mappings"](#).

## 77.2.2 XPath Support

When using XML records, TopLink EIS mappings use XPath statements to efficiently map the attributes of a Java object to locations in an XML record. For more information about using XPath with XML mappings, see [Section 17.2.7, "Mappings and XPath"](#).

## 77.2.3 xsd:list and xsd:union Support

When using XML records, you can use EIS direct (see [Section 77.3, "EIS Direct Mapping"](#)) and composite direct collection (see [Section 77.4, "EIS Composite Direct Collection Mapping"](#)) mappings to map to `xsd:list` and `xsd:union` types in an XML record.

For more information, see [Section 17.2.8, "Mappings and xsd:list and xsd:union Types"](#).

## 77.2.4 jaxb:class Support

When using XML records, you can configure an EIS composite object mapping (see [Section 77.5, "EIS Composite Object Mapping"](#)) to accommodate `jaxb:class` customizations with the following XSD structures:

- `all`
- `sequence`
- `choice`
- `group`

For more information, see [Section 17.2.9, "Mappings and the javax:class Customization"](#).

## 77.2.5 Typesafe Enumeration Support

You can map a Java attribute to a typesafe enumeration using the `JAXBTypesafeEnumConverter` with an `EISDirectMapping` or `EISCompositeDirectCollectionMapping` with XML records.

For more information, see [Section 17.2.10, "Mappings and JAXB Typesafe Enumerations"](#).

## 77.2.6 Composite and Reference EIS Mappings

TopLink supports composite and reference EIS mappings. Although there is a source and target object in both mapping types, the TopLink runtime handles interactions with each differently. This section explains how.

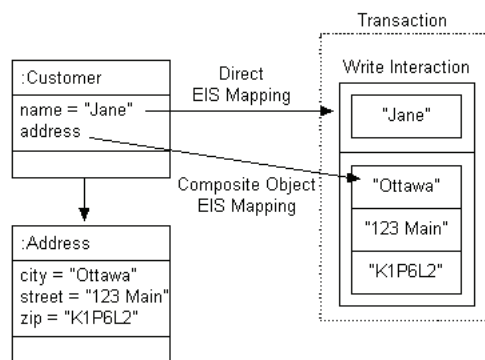
### 77.2.6.1 Composite EIS Mappings

In a composite EIS mapping ([Section 77.4, "EIS Composite Direct Collection Mapping"](#), [Section 77.5, "EIS Composite Object Mapping"](#), and [Section 77.6, "EIS Composite Collection Mapping"](#)), the source object contains (owns) the target object.

TopLink puts the attributes of the target (owned) object (or the owned collection of objects) into the source (owning) object's record as a subrecord. The target object needs not be a root object type (see [Section 76.6, "Configuring an EIS Descriptor as a Root or Composite Type"](#)): it needs not have interactions defined for it.

[Figure 77-1](#) illustrates a read interaction on an instance of the `Customer` class using indexed records. For the composite object EIS mapping defined for the `address` attribute, TopLink creates an `Address` subrecord in the `Customer` record.

**Figure 77-1 EIS Composite Mappings**



### 77.2.6.2 Reference EIS Mappings

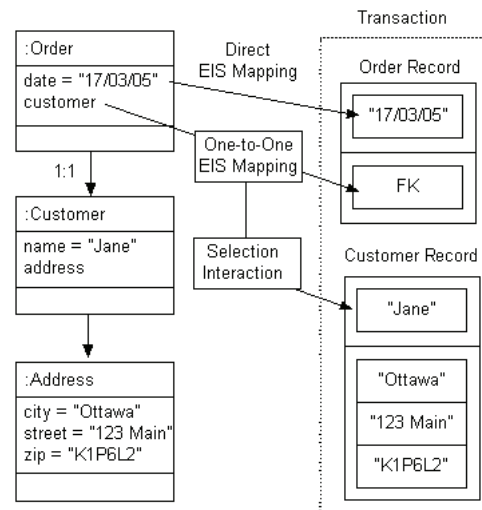
In a reference EIS mapping ([Section 77.7, "EIS One-to-One Mapping"](#) and [Section 77.8, "EIS One-to-Many Mapping"](#)), the source object contains only a foreign key (pointer) to the target object or, alternatively, the target object contains a foreign key to the source object (key on target).

TopLink puts the foreign key of the target object into the source object's record as a simple value. When an interaction is executed on the source object, TopLink uses the selection interaction that you define on its descriptor to retrieve the appropriate target object instance and creates a record for it in the source object's transaction. By default, the selection interaction is the target object's read interaction. If the read interaction is

not sufficient, you can define a separate selection interaction (see [Section 78.4, "Configuring Selection Interaction"](#)). Because both the source and target object use interactions, they must both be of a root object type (see [Section 76.6, "Configuring an EIS Descriptor as a Root or Composite Type"](#)).

[Figure 77–2](#) illustrates a read interaction on an instance of the `Order` class using indexed records. For the one-to-one EIS mapping defined for the `customer` attribute, TopLink puts the target `Customer` object's foreign key into the `Order` record as a simple value. TopLink then uses the selection interaction you configure on the `Order` descriptor to retrieve the appropriate instance of `Customer` and creates a record for it in the `Order` object's transaction.

**Figure 77–2 EIS Reference Mappings**

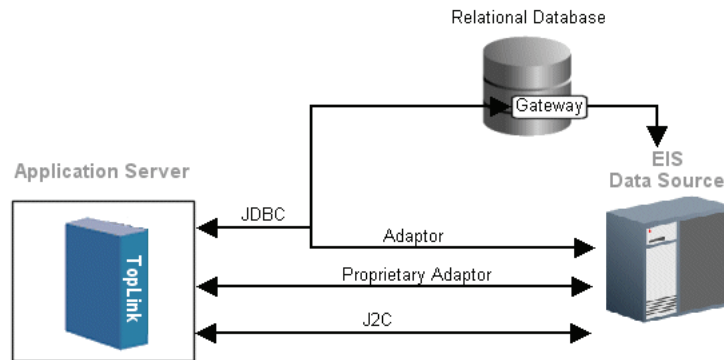


## 77.2.7 EIS Mapping Architecture

[Figure 77–3](#) illustrates the following possible TopLink EIS mapping architectures:

- JDBC database gateway (such as Oracle Database 10g)
- JDBC adapter
- Proprietary adapter (such as Oracle Interconnect)
- JCA

**Figure 77–3 Possible EIS Mapping Architectures**



The best solution may vary, depending on your specific EIS and infrastructure.

### 77.3 EIS Direct Mapping

An EIS direct mapping maps a simple object attribute directly to an EIS record according to its descriptor’s record type, as shown in [Table 77–2](#).

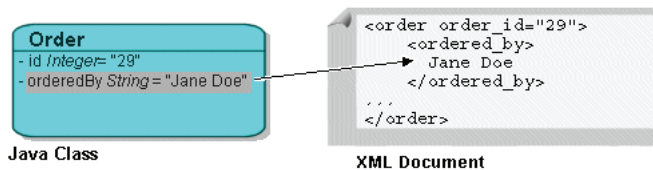
**Table 77–2 EIS Direct Mapping by EIS Record Type**

EIS Record Type	Mapping Behavior
Indexed	Maps directly to a field in the indexed record.
Mapped	Maps directly to a field in the mapped record.
XML	Maps directly to an attribute or text node in the XML record <sup>1</sup> .

<sup>1</sup> See also [Section 53.3, "XML Direct Mapping"](#).

[Figure 77–4](#) illustrates a direct EIS mapping between `Order` class attribute `orderedBy` and XML record attribute `ordered_by` within the `order` element.

**Figure 77–4 EIS Direct Mappings**



See [Chapter 79, "Configuring an EIS Direct Mapping"](#) for more information.

### 77.4 EIS Composite Direct Collection Mapping

An EIS composite direct collection mapping maps a collection of Java attributes directly to an EIS record according to its descriptor’s record type, as shown in [Table 77–3](#).



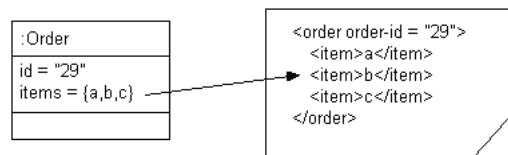
**Table 77-3 EIS Composite Direct Collection Mapping by EIS Record Type**

EIS Record Type	Mapping Behavior
Indexed	Maps directly to a subrecord in the indexed record <sup>1</sup> .
Mapped	Maps directly to a subrecord in the mapped record <sup>1</sup> .
XML	Maps directly to an attribute or text node in the XML record <sup>2</sup> .

<sup>1</sup> See also [Section 77.2.6.1, "Composite EIS Mappings"](#).

<sup>2</sup> See also [Section 53.4, "XML Composite Direct Collection Mapping"](#).

[Figure 77-5](#) illustrates a composite direct collection mapping between `Order` class attribute `items` and an XML record. The `Order` attribute `items` is a collection type (such as `Vector`). It is mapped to an XML record composed of an `order` element that contains a sequence of `item` elements.

**Figure 77-5 EIS Composite Direct Collection Mapping**

See [Chapter 80, "Configuring an EIS Composite Direct Collection Mapping"](#) for more information.

## 77.5 EIS Composite Object Mapping

An EIS composite object mapping maps a Java object to a privately owned one-to-one relationship in an EIS record according to its descriptor's record type, as shown in [Table 77-4](#).

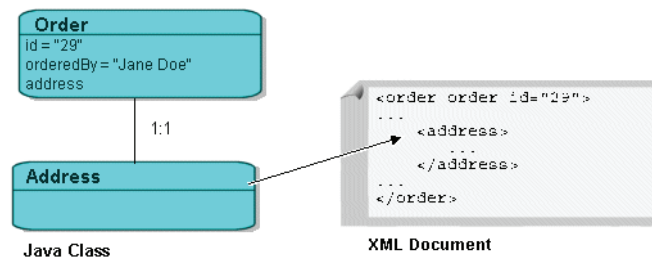
**Table 77-4 EIS Composite Object Mapping by EIS Record Type**

EIS Record Type	Mapping Behavior
Indexed	Maps directly to a subrecord in the indexed record <sup>1</sup> .
Mapped	Maps directly to a subrecord in the mapped record <sup>1</sup> .
XML	Maps directly to an attribute or text node in the XML record <sup>2</sup> .

<sup>1</sup> See also [Section 77.2.6.1, "Composite EIS Mappings"](#).

<sup>2</sup> See also [Section 53.5, "XML Composite Object Mapping"](#).

[Figure 77-6](#) illustrates a composite object EIS mapping between `Order` class attribute `address` and an XML record. `Order` attribute `address` is mapped to an XML record composed of an `order` element that contains an `address` element.

**Figure 77–6 EIS Composite Object Mappings**

You can use an EIS composite object mapping with a change policy (see [Section 119.30](#), "Configuring Change Policy").

See [Chapter 81](#), "Configuring an EIS Composite Object Mapping" for more information.

## 77.6 EIS Composite Collection Mapping

An EIS composite collection mapping maps a collection of Java objects to a privately owned one-to-many relationship in an EIS record according to its descriptor's record type, as shown in [Table 77–5](#). Composite collection mappings can reference any class that has a TopLink descriptor.

**Table 77–5 EIS Composite Collection Mapping by EIS Record Type**

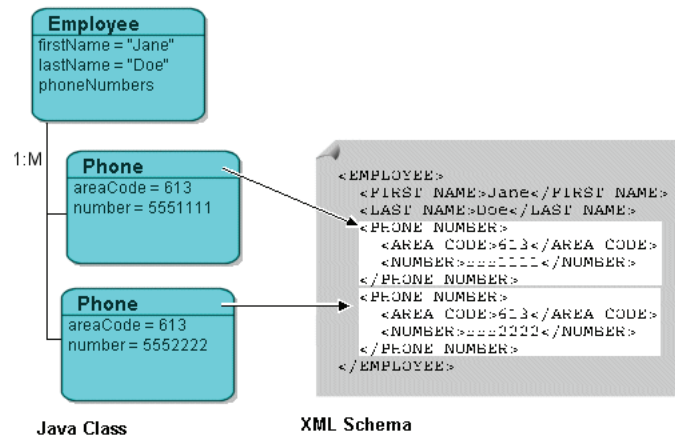
EIS Record Type	Mapping Behavior
Indexed	Maps directly to a subrecord in the indexed record <sup>1</sup> .
Mapped	Maps directly to a subrecord in the mapped record <sup>1</sup> .
XML	Maps directly to an attribute or text node in the XML record <sup>2</sup> .

<sup>1</sup> See also [Section 77.2.6.1](#), "Composite EIS Mappings".

<sup>2</sup> See also [Section 53.6](#), "XML Composite Collection Mapping".

[Figure 77–7](#) illustrates a composite collection EIS mapping between Phone class attribute `phoneNumbers` and an XML record. Employee attribute `phoneNumbers` is mapped to an XML record composed of an EMPLOYEE element that contains a sequence of PHONE\_NUMBER elements.

Figure 77–7 EIS Composite Collection Mappings



See [Chapter 82, "Configuring an EIS Composite Collection Mapping"](#) for more information.

## 77.7 EIS One-to-One Mapping

An EIS one-to-one mapping is a reference mapping that represents the relationship between a single source and target object. The source object usually contains a foreign key (pointer) to the target object (key on source). Alternatively, the target object may contain a foreign key to the source object (key on target). Because both the source and target object use interactions, they must both be of a root object type (see [Section 76.6, "Configuring an EIS Descriptor as a Root or Composite Type"](#))

[Table 77–6](#) summarizes the behavior of this mapping depending on the EIS record type you are using.

**Table 77–6 EIS One-to-One Mapping by EIS Record Type**

EIS Record Type	Mapping Behavior
Indexed	<p>A new indexed record is created for the target object<sup>1</sup>:</p> <ul style="list-style-type: none"> <li>With the Key on Source use case, the foreign key(s) is added to the record for the source object.</li> <li>With the Key on Target use case, the foreign key(s) is added to the record for the target object</li> </ul>
Mapped	<p>A new mapped record is created for the target object<sup>1</sup>:</p> <ul style="list-style-type: none"> <li>With the Key on Source use case, the foreign key(s) is added to the record for the source object.</li> <li>With the Key on Target use case, the foreign key(s) is added to the record for the target object</li> </ul>
XML	<p>A new XML record is created for the target object:</p> <ul style="list-style-type: none"> <li>With the Key on Source use case, the foreign key(s) is added to the record for the source object.</li> <li>With the Key on Target use case, the foreign key(s) is added to the record for the target object</li> </ul>

<sup>1</sup> See also [Section 77.2.6.2, "Reference EIS Mappings"](#).

This section describes the following:

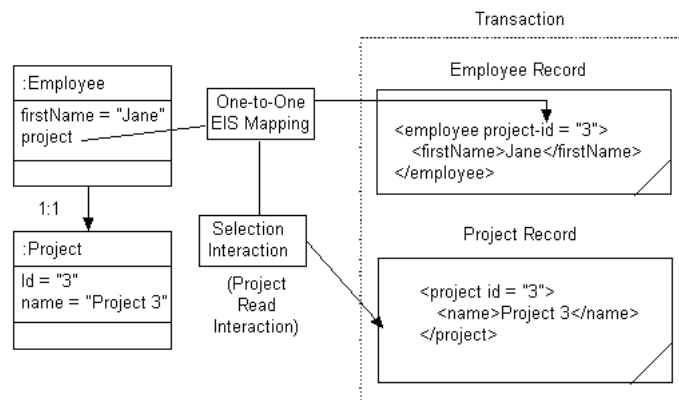
- EIS One-to-One Mappings with Key on Source
- EIS One-to-One Mappings with Key on Target

See [Chapter 83, "Configuring an EIS One-to-One Mapping"](#) for more information.

### 77.7.1 EIS One-to-One Mappings with Key on Source

[Figure 77–8](#) illustrates a EIS one-to-one mapping between the `Employee` class attribute `project` and the `Project` class using XML records in a **key on source** design.

**Figure 77–8 EIS One-to-One Mapping with Key on Source**



When a read interaction is executed on the `Employee` object, TopLink puts the target `Project` object's primary key into the `Employee` record as a simple value. TopLink then uses the selection interaction you configure on the `Employee` descriptor to retrieve the appropriate instance of `Project` and creates a record for it in the `Employee` object's transaction. In this example, you can designate the `Project` class's read interaction as the selection interaction.

The general procedure for creating and configuring this mapping is as follows:

1. Create a one-to-one EIS mapping on `Employee` attribute `project`.
2. Configure the reference descriptor as `Project` (see [Section 78.3, "Configuring Reference Descriptors"](#)).
3. Configure the source and target foreign keys (see [Section 83.2, "Configuring Foreign Key Pairs"](#)).

In this example:

- Source XML Field: `@project-id`
  - Target XML Field: `@id`
4. Configure the selection interaction (see [Section 78.4, "Configuring Selection Interaction"](#)).

In this example, you can designate the `Project` class's read interaction as the selection interaction.

Given the XSD shown in [Example 77–1](#), you can configure an EIS one-to-one mapping with key on source, as [Example 77–2](#) shows. In this case, the source object contains a foreign key reference to the target object. In the following example, the source object is

Employee and the target object is Project. Here, the Employee object has a Project that is referenced using the project's id.

**Example 77-1 XML Schema for EIS One-to-One Mapping with Key on Source**

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xsd:element name="employee" type="employee-type"/>
  <xsd:element name="project" type="project-type"/>
  <xsd:complexType name="employee-type">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="project">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="project-id" type="xsd:integer"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="project-type">
    <xsd:sequence>
      <xsd:element name="id" type="xsd:integer"/>
      <xsd:element name="leader" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

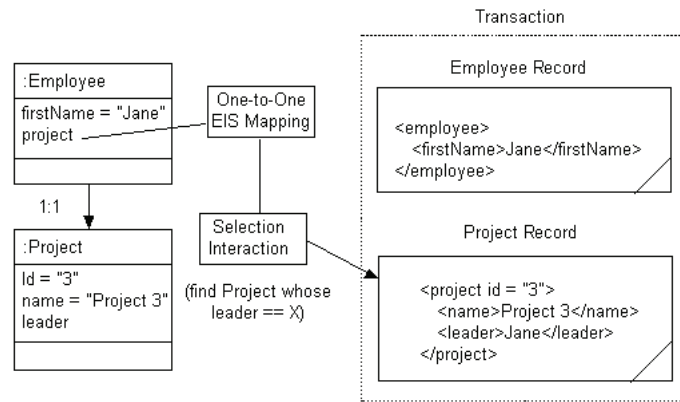
**Example 77-2 EIS One-to-One Mapping with Key On Source**

```
// Employee descriptor
EISDescriptor descriptor = new EISDescriptor();
descriptor.setJavaClass(Employee.class);
descriptor.setDataTypeName("employee");
descriptor.setPrimaryKeyFieldName("name/text()");

EISOneToOneMapping projectMapping = new EISOneToOneMapping();
projectMapping.setReferenceClass(Project.class);
projectMapping.setAttributeName("project");
projectMapping.dontUseIndirection();
projectMapping.addForeignKeyFieldName("project/project-id/text()", "id/text()");
```

## 77.7.2 EIS One-to-One Mappings with Key on Target

Figure 77-9 illustrates EIS one-to-one mapping between the Employee class attribute project and the Project class using XML records in a **key on target** design. You still configure a one-to-one EIS mapping between Employee and Project, but in this design, the Project attribute leader contains the foreign key of the Employee object.

**Figure 77–9 EIS One-to-One Mapping with Key on Target**

When a read interaction is executed on the `Employee` object, TopLink uses the selection interaction you configure on the `Employee` descriptor to retrieve the appropriate instance of `Project` and creates a record for it in the `Employee` object's transaction. In this example, the `Project` class's read interaction is unlikely to be sufficient: it is likely implemented to read based on `Project` attribute `Id`, not on `leader`. If this is the case, you must define a separate selection interaction on the `Employee` descriptor that does the following: finds the `Project`, whose `leader` equals `X`, where `X` is the value of `Employee` attribute `firstName`.

Note that in this configuration, `Project` attribute `leader` is not persisted. If you want this attribute persisted, you must configure a one-to-one EIS mapping from it to `Employee` attribute `firstName`.

The general procedure for creating and configuring this mapping is as follows:

1. Create a one-to-one EIS mapping on `Employee` attribute `project`.
2. Configure the reference descriptor as `Project` (see [Section 78.3, "Configuring Reference Descriptors"](#)).
3. Configure the source and target foreign keys (see [Section 83.2, "Configuring Foreign Key Pairs"](#)).

In this example:

- Source XML Field: `firstName/text()`
  - Target XML Field: `leader/text()`
4. Configure the selection interaction (see [Section 78.4, "Configuring Selection Interaction"](#)).

In this example, you must define a separate selection interaction on the `Employee` descriptor.

Given the XSD shown in [Example 77–3](#), you can configure an EIS one-to-one mapping with key on target, as [Example 77–4](#) shows. In this case, the target object contains a foreign key reference to the source object. In the following example, the source object is `Employee`, and the target object is `Project`. Here, a `Project` references its `leader` using the employee's name.

**Example 77–3 XML Schema for EIS One-to-One Mapping with Key on Target**

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
```

```

attributeFormDefault="unqualified">
<xsd:element name="employee" type="employee-type"/>
<xsd:element name="project" type="project-type"/>
<xsd:complexType name="employee-type">
  <xsd:sequence>
    <xsd:element name="name" type="xsd:string"/>
    <xsd:element name="project">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="project-id" type="xsd:integer"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="project-type">
  <xsd:sequence>
    <xsd:element name="id" type="xsd:integer"/>
    <xsd:element name="leader" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>

```

#### Example 77-4 EIS One-to-One Mapping with Key on Target

```

// Project descriptor
EISDescriptor descriptor = new EISDescriptor();
descriptor.setJavaClass(Project.class);
descriptor.setDataTypeName("project");
descriptor.setPrimaryKeyFieldName("id/text()");

EISOneToOneMapping leaderMapping = new EISOneToOneMapping();
leaderMapping.setReferenceClass(Employee.class);
leaderMapping.setAttributeName("leader");
leaderMapping.dontUseIndirection();
leaderMapping.addForeignKeyFieldName("leader/text()", "name/text()");

```

## 77.8 EIS One-to-Many Mapping

An EIS one-to-many mapping is a reference mapping that represents the relationship between a single source object and a collection of target objects. The source object usually contains a foreign key (pointer) to the target objects (key on source); alternatively, the target objects may contain a foreign key to the source object (key on target). Because both the source and target objects use interactions, they must all be of a root object type (see [Section 76.6, "Configuring an EIS Descriptor as a Root or Composite Type"](#)).

[Table 77-7](#) summarizes the behavior of this mapping depending on the EIS record type you are using.

**Table 77-7 EIS One-to-Many Mapping by EIS Record Type**

EIS Record Type	Mapping Behavior
Indexed	<p>A new indexed record is created for each target object<sup>1</sup>:</p> <ul style="list-style-type: none"> <li>■ With the Key on Source use case, the foreign key(s) is added to the record for the source object for each target object.</li> <li>■ With the Key on Target use case, the foreign key(s) is added to the record for the target object</li> </ul>

**Table 77-7 (Cont.) EIS One-to-Many Mapping by EIS Record Type**

EIS Record Type	Mapping Behavior
Mapped	<p>A new mapped record is created for each target object<sup>1</sup>:</p> <ul style="list-style-type: none"> <li>With the Key on Source use case, the foreign key(s) is added to the record for the source object.</li> <li>With the Key on Target use case, the foreign key(s) is added to the record for the target object</li> </ul>
XML	<p>.A new XML record is created for each target object:</p> <ul style="list-style-type: none"> <li>With the Key on Source use case, the foreign key(s) is added to the record for the source object for each target object.</li> <li>With the Key on Target use case, the foreign key(s) is added to the record for the target object</li> </ul>

<sup>1</sup> See also Section 77.2.6.2, "Reference EIS Mappings".

This section describes the following:

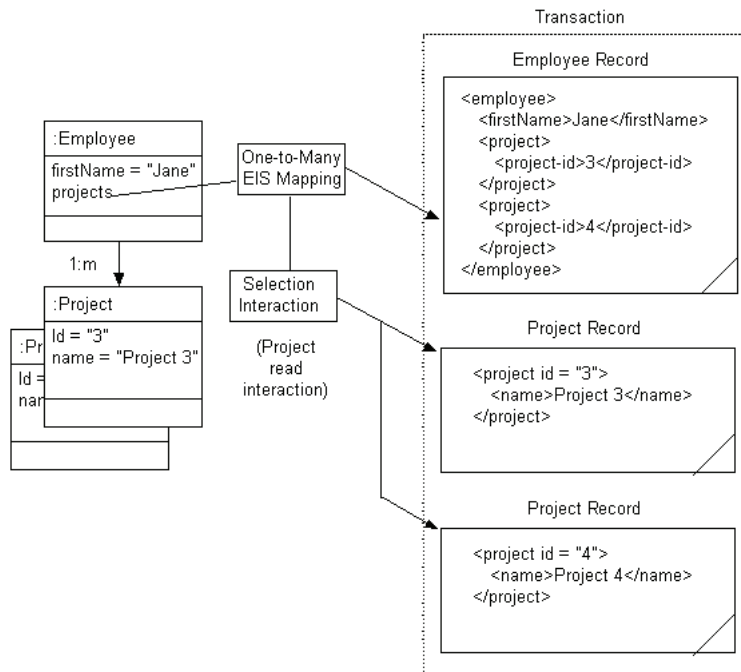
- EIS One-to-Many Mappings with Key on Source
- EIS One-to-Many Mappings with Key on Target

See Chapter 84, "Configuring an EIS One-to-Many Mapping" for more information.

### 77.8.1 EIS One-to-Many Mappings with Key on Source

Figure 77-10 illustrates an EIS one-to-many mapping between the Employee class attribute projects and multiple Project class instances using XML records in a key on source design.

**Figure 77-10 EIS One-to-Many Mapping with Key on Source**



When a read interaction is executed on the Employee object, TopLink puts each target Project object's foreign key into the Employee record as a subelement. If you



specify only one pair of source and target XML fields, by default, the foreign keys are not grouped in the `Employee` record. If you specify more than one pair of source and target XML fields, you must choose a grouping element (see [Section 78.3, "Configuring Reference Descriptors"](#)). [Figure 77-10](#) shows an `Employee` record with grouping element `Project`. `TopLink` then uses the selection interaction you configure on the `Employee` descriptor to retrieve the appropriate instances of `Project` and creates a record for each in the `Employee` object's transaction. In this example, you can designate the `Project` class's read interaction as the selection interaction.

The general procedure for creating and configuring this mapping is as follows:

1. Create a one-to-many EIS mapping on `Employee` attribute `project`.
2. Configure the reference descriptor as `Project` (see [Section 78.3, "Configuring Reference Descriptors"](#)).
3. Configure the source and target foreign keys (see [Section 84.2, "Configuring Foreign Key Pairs"](#)).

In this example:

- Source XML Field: `PROJECT`
  - Target XML Field: `@ID`
4. Configure the selection interaction (see [Section 78.4, "Configuring Selection Interaction"](#)).

In this example, you can designate the `Project` class's read interaction as the selection interaction.

Given the XSD shown in [Example 77-3](#), you can configure an EIS one-to-many mapping with key on source, as [Example 77-4](#) shows. In this case, the source object contains a foreign key reference to the target object. In the following example, the source object is `Employee`, and the target object is `Project`. Here, the `Employee` object has one or more `Project` instances that are referenced by `Project id`.

#### **Example 77-5 XML Schema for EIS One-to-Many Mapping with Key on Source**

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xsd:element name="employee" type="employee-type"/>
  <xsd:element name="project" type="project-type"/>
  <xsd:complexType name="employee-type">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="projects">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="project-id"
              type="xsd:integer" maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="project-type">
    <xsd:sequence>
      <xsd:element name="id" type="xsd:integer"/>
      <xsd:element name="leader" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

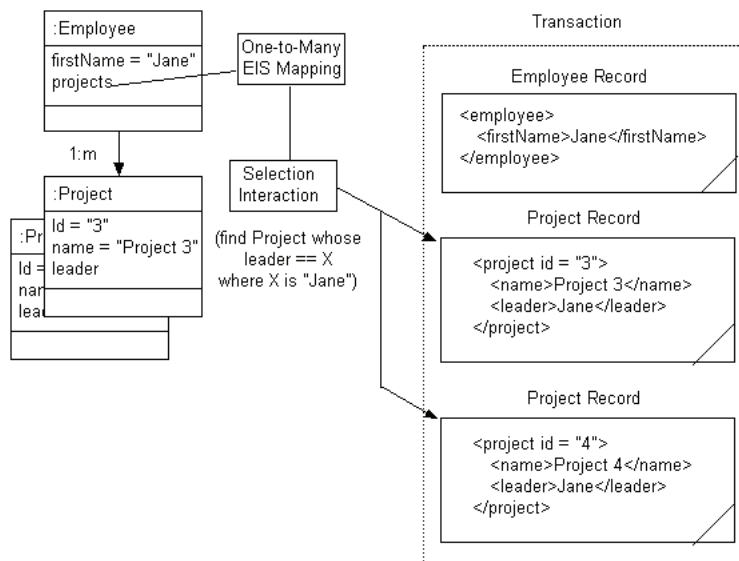
**Example 77-6 EIS One-to-Many Mapping with Key on Source**

```
// Employee descriptor
EISDescriptor descriptor = new EISDescriptor();
descriptor.setJavaClass(Employee.class);
descriptor.setDataTypeName("employee");
descriptor.setPrimaryKeyFieldName("name/text()");

EISOneToManyMapping projectMapping = new EISOneToManyMapping();
projectMapping.setReferenceClass(Project.class);
projectMapping.setAttributeName("projects");
projectMapping.setForeignKeyGroupingElement("projects");
projectMapping.setIsForeignKeyRelationship(true);
projectMapping.dontUseIndirection();
projectMapping.addForeignKeyFieldName("project-id/text()", "id/text()");
```

**77.8.2 EIS One-to-Many Mappings with Key on Target**

Figure 77-9 illustrates an EIS one-to-many mapping between the `Employee` class attribute `projects` and multiple `Project` class instances using XML records in a key on target design. You still configure a one-to-one EIS mapping between `Employee` and `Project` but in this design, the `Project` attribute `leader` contains the foreign key of the `Employee` object.

**Figure 77-11 EIS One-to-Many Mapping with Key on Target**

When a read interaction is executed on the `Employee` object, TopLink uses the selection interaction you configure on the `Employee` descriptor to retrieve the appropriate instances of `Project` and creates a record for each in the `Employee` object's transaction. In this example, the `Project` class's read interaction is unlikely to be sufficient: it is likely implemented to read based on `Project` attribute `Id`, not on `leader`. If this is the case, you must define a separate selection interaction on the `Employee` descriptor that does the following: finds the `Project`, whose `leader` equals `X`, where `X` is "Jane".

Note that in this configuration, `Project` attribute `leader` is not persisted. If you want this attribute persisted, you must configure a one-to-one EIS mapping from it to `Employee` attribute `firstName`.

The general procedure for creating and configuring this mapping is as follows:

1. Create a one-to-one EIS mapping on `Employee` attribute `project`.
2. Configure the reference descriptor as `Project` (see [Section 78.3, "Configuring Reference Descriptors"](#)).
3. Configure the source and target foreign keys (see [Section 83.2, "Configuring Foreign Key Pairs"](#)).

In this example, you select **Foreign Keys Located On Source** and specify one pair of source and target XML fields:

- Source XML Field:
- Target XML Field:

4. Configure the selection interaction (see [Section 78.4, "Configuring Selection Interaction"](#)).

In this example, you must define a separate selection interaction on the `Employee` descriptor.

Given the XSD shown in [Example 77–3](#), you can configure an EIS one-to-many mapping with key on target, as [Example 77–4](#) shows. In this case, the target object contains a foreign key reference to the source object. In the following example, the source object is `Employee`, and the target object is `Project`. Here, each `Project` references its `leader` using the employee's name.

#### **Example 77–7 XML Schema for EIS One-to-Many Mapping with Key on Target**

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xsd:element name="employee" type="employee-type"/>
  <xsd:element name="project" type="project-type"/>
  <xsd:complexType name="employee-type">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="projects">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="project-id"
              type="xsd:integer" maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="project-type">
    <xsd:sequence>
      <xsd:element name="id" type="xsd:integer"/>
      <xsd:element name="leader" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

#### **Example 77–8 EIS One-to-Many Mapping with Key on Target**

```
// Project descriptor
EISDescriptor descriptor = new EISDescriptor();
descriptor.setJavaClass(Project.class);
descriptor.setDataTypeName("project");
descriptor.setPrimaryKeyFieldName("id/text()");
```

```

EISOneToManyMapping leaderMapping = new EISOneToOneMapping();
leaderMapping.setReferenceClass(Employee.class);
leaderMapping.setAttributeName("leader");
leaderMapping.dontUseIndirection();
leaderMapping.addForeignKeyFieldName("leader/text()", "name/text()");
    
```

## 77.9 EIS Transformation Mapping

A transformation EIS mapping lets you create a custom mapping, where one or more fields in an EIS record can be used to create the object to be stored in a Java class's attribute.

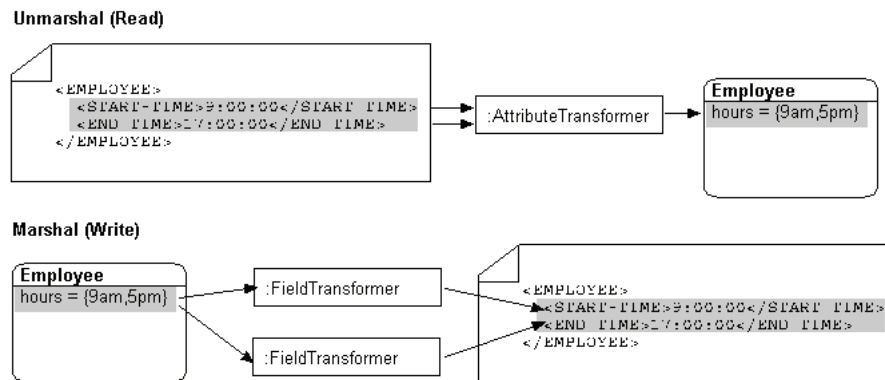
Table 77-8 summarizes the behavior of this mapping depending on the EIS record type you are using.

**Table 77-8 EIS Transformation Mapping by EIS Record Type**

EIS Record Type	Mapping Behavior
Indexed	.The field transformer adds data to the indexed record (you have access to the indexed record in the attribute transformer).
Mapped	.The field transformer adds data to the mapped record (you have access to the mapped record in the attribute transformer).
XML	.The field transformer adds data to the XML record (you have access to the XML record in the attribute transformer).

As Figure 77-12 illustrates, you configure the transformation mapping with an `oracle.toplink.mappings.transformers.AttributeTransformer` instance to perform the XML instance-to-Java attribute transformation at unmarshal time. In this example, the `AttributeTransformer` combines two XML text nodes into a single Java object.

**Figure 77-12 EIS Transformation Mappings**



See Chapter 85, "Configuring an EIS Transformation Mapping" for more information.

---



---

## Configuring an EIS Mapping

This chapter describes how to configure an EIS mapping.

This chapter includes the following sections:

- [Introduction to EIS Mapping Configuration](#)
- [Configuring Common EIS Mapping Options](#)
- [Configuring Reference Descriptors](#)
- [Configuring Selection Interaction](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 78.1 Introduction to EIS Mapping Configuration

[Table 78–1](#) lists the types of EIS mappings that you can configure and provides a cross-reference to the type-specific chapter that lists the configurable options supported by that type.

**Table 78–1** *Configuring EIS Mappings*

<b>If you are creating...</b>	<b>See Also...</b>
EIS direct mapping (see <a href="#">Section 77.3, "EIS Direct Mapping"</a> )	<a href="#">Chapter 79, "Configuring an EIS Direct Mapping"</a>
EIS composite direct collection mapping (see <a href="#">Section 77.4, "EIS Composite Direct Collection Mapping"</a> )	<a href="#">Chapter 80, "Configuring an EIS Composite Direct Collection Mapping"</a>
EIS composite object mapping (see <a href="#">Section 77.5, "EIS Composite Object Mapping"</a> )	<a href="#">Chapter 81, "Configuring an EIS Composite Object Mapping"</a>
EIS composite collection mapping (see <a href="#">Section 77.6, "EIS Composite Collection Mapping"</a> )	<a href="#">Chapter 82, "Configuring an EIS Composite Collection Mapping"</a>
EIS one-to-one mapping (see <a href="#">Section 77.7, "EIS One-to-One Mapping"</a> )	<a href="#">Chapter 83, "Configuring an EIS One-to-One Mapping"</a>
EIS one-to-many mapping (see <a href="#">Section 77.8, "EIS One-to-Many Mapping"</a> )	<a href="#">Chapter 84, "Configuring an EIS One-to-Many Mapping"</a>
EIS transformation mapping (see <a href="#">Section 77.9, "EIS Transformation Mapping"</a> )	<a href="#">Chapter 85, "Configuring an EIS Transformation Mapping"</a>

For more information, see the following:

- [Chapter 17, "Introduction to Mappings"](#)
- [Chapter 77, "Introduction to EIS Mappings"](#)

## 78.2 Configuring Common EIS Mapping Options

Table 78–2 lists the configurable options shared by two or more EIS mapping types. In addition to the configurable options described here, you must also configure the options described for the specific EIS mapping types (see [Section 77.1, "EIS Mapping Types"](#)), as shown in [Table 78–1](#).

**Table 78–2 Common Options for EIS Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Read-only (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Indirection (lazy loading) (see <a href="#">Section 121.3, "Configuring Indirection (Lazy Loading)"</a> )	✓	✓	✓
XPath (see <a href="#">Section 121.4, "Configuring XPath"</a> )	✓	✓	✓
Default null value (see <a href="#">Section 121.5, "Configuring a Default Null Value at the Mapping Level"</a> )	✓	✓	✓
Reference descriptors (see <a href="#">Section 78.3, "Configuring Reference Descriptors"</a> )	✓	✓	✓
Method or direct field access (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Private or independent relationships (see <a href="#">Section 121.7, "Configuring Private or Independent Relationships"</a> )	✓	✓	✓
Comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	
Serialized object converter (see <a href="#">Section 121.9, "Configuring a Serialized Object Converter"</a> )	✓	✓	✓
Type conversion converter (see <a href="#">Section 121.10, "Configuring a Type Conversion Converter"</a> )	✓	✓	✓
Object type converter (see <a href="#">Section 121.11, "Configuring an Object Type Converter"</a> )	✓	✓	✓
Simple type translator (see <a href="#">Section 121.12, "Configuring a Simple Type Translator"</a> )	✓	✓	✓
Container policy (see <a href="#">Section 121.14, "Configuring Container Policy"</a> )	✓	✓	✓
Selection interaction (see <a href="#">Section 78.4, "Configuring Selection Interaction"</a> )	✓	✓	✓
Use of a single node (see <a href="#">Section 121.19, "Configuring the Use of a Single Node"</a> )	✓	✓	✓
JAXB typesafe enumeration converter (see <a href="#">Section 121.13, "Configuring a JAXB Typesafe Enumeration Converter"</a> )			✓

## 78.3 Configuring Reference Descriptors

In EIS mappings that extend `oracle.toplink.mappings.ForeignReferenceMapping` or `oracle.toplink.mappings.AggregateMapping` class, attributes reference other TopLink descriptors—not the data source. You can select a descriptor in the current project, or a descriptor from some other project.

[Table 78–3](#) summarizes which EIS mappings support this option.

**Table 78–3 Mapping Support for Reference Descriptor**

Mapping	How to Use Oracle JDeveloper	How to Configure Reference Descriptors Using TopLink Workbench	How to Use Java
Direct mapping (see <a href="#">Section 77.3, "EIS Direct Mapping"</a> )			
Composite direct collection mapping (see <a href="#">Section 77.4, "EIS Composite Direct Collection Mapping"</a> )			
Composite object mapping (see <a href="#">Section 77.5, "EIS Composite Object Mapping"</a> )	✓	✓	✓
Composite collection mapping (see <a href="#">Section 77.6, "EIS Composite Collection Mapping"</a> )	✓	✓	✓
One-to-one mapping (see <a href="#">Section 77.7, "EIS One-to-One Mapping"</a> )	✓	✓	✓
One-to-many mapping (see <a href="#">Section 77.8, "EIS One-to-Many Mapping"</a> )	✓	✓	✓
Transformation mapping (see <a href="#">Section 77.9, "EIS Transformation Mapping"</a> )			

### 78.3.1 How to Configure Reference Descriptors Using TopLink Workbench

To specify a reference descriptor for an EIS mapping, use this procedure.

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

**Figure 78–1 General Tab, Reference Descriptor Field**

The screenshot shows the 'General' tab of the 'Table Reference' configuration window. The 'Reference Descriptor' field is highlighted with a red circle and contains the value 'Address (examples.servletjsp.model)'. Below this field are several options: 'Method Accessing' (unchecked), 'Read-Only' (unchecked), 'Private Owned' (unchecked), 'Batch Reading' (unchecked), 'Join Fetch' (set to 'None'), 'Use Indirection' (checked), and 'ValueHolder' (selected) with 'Proxy' (unselected). At the bottom, there is a 'Maintains Bidirectional Relationship' section with 'Relationship Partner' set to '<none selected>' and a 'Comment' text area.

Use the **Reference Descriptor** field to select the descriptor referenced by this relationship mapping.

---



---

**Note:** For one-to-one and one-to-many EIS mappings, the reference descriptor must be a root descriptor. See [Section 76.6, "Configuring an EIS Descriptor as a Root or Composite Type"](#).

---



---

You can specify a reference descriptor that is not in the current TopLink Workbench project. For example, to create a mapping to an `Employee` class that does not exist in the current project, do the following:

1. Add the `Employee` class to your current project. See [Section 116.2, "Working with Projects"](#).
2. Create the relationship mapping to the `Employee` descriptor.
3. Deactivate the `Employee` descriptor. See [Active and Inactive Descriptors](#).

When you generate the deployment XML for your project, the mapping to the `Employee` class will be included, but not the `Employee` class itself.

## 78.4 Configuring Selection Interaction

In EIS mappings that extend `oracle.toplink.mappings.ForeignReferenceMapping` class, TopLink uses a selection interaction to acquire the instance of the target object to which the mapping refers.

By default, TopLink uses the read interaction you define for the mapping's reference descriptor (see [Section 78.3, "Configuring Reference Descriptors"](#)). In most cases, this interaction is sufficient. If the reference descriptor's read interaction is not sufficient, you can define a separate interaction.

[Table 78-4](#) summarizes which EIS mappings support this option.

**Table 78-4 Mapping Support for Selection Interaction**

Mapping	How to Use Oracle JDeveloper	How to Configure Selection Interaction Using TopLink Workbench	How to Use Java
Direct mapping (see <a href="#">Section 77.3, "EIS Direct Mapping"</a> )			
Composite direct collection mapping (see <a href="#">Section 77.4, "EIS Composite Direct Collection Mapping"</a> )			
One-to-one mapping (see <a href="#">Section 77.7, "EIS One-to-One Mapping"</a> )			
One-to-many mapping (see <a href="#">Section 77.8, "EIS One-to-Many Mapping"</a> )			
Composite object mapping (see <a href="#">Section 77.5, "EIS Composite Object Mapping"</a> )	✓	✓	✓
Composite collection mapping (see <a href="#">Section 77.6, "EIS Composite Collection Mapping"</a> )	✓	✓	✓



**Table 78–4 (Cont.) Mapping Support for Selection Interaction**

Mapping	How to Use Oracle JDeveloper	How to Configure Selection Interaction Using TopLink Workbench	How to Use Java
Transformation mapping (see <a href="#">Section 77.9</a> , "EIS Transformation Mapping")			

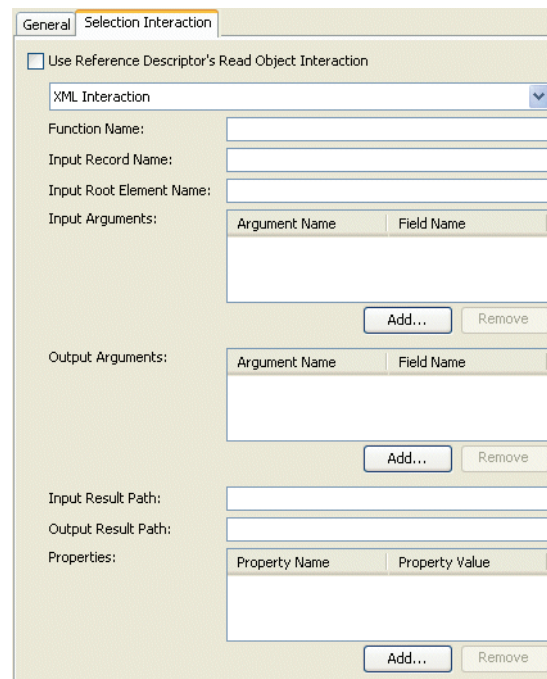
For more information about how TopLink uses the selection criteria, see [Section 77.2.6.2](#), "Reference EIS Mappings".

### 78.4.1 How to Configure Selection Interaction Using TopLink Workbench

To specify the selection interaction (such as Read Object) for the EIS mapping, use this procedure:

1. Select the one-to-many EIS mapping in the **Navigator**. Its properties appear in the Editor.
2. Click the **Selection Interaction** tab. The Selection Interaction tab appears.

**Figure 78–2 Selection Interaction Tab**



Use the following information to enter data in each field on the tab:

Field	Description
<b>Function Name</b>	The name of the EIS function that this call type (Read Object or Read All) invokes on the EIS.
<b>Input Record Name</b>	The name passed to the JCA adapter when creating the input record.

---

<b>Field</b>	<b>Description</b>
<b>Input Root Element Name</b>	The root element name to use for the input DOM.
<b>Input Arguments</b>	<p>The query argument name to map to the interaction field or XPath nodes in the argument record.</p> <p>For example, if you are using XML records, use this option to map input argument name to the XPath <code>name/first-name</code>.</p>
<b>Output Arguments</b>	<p>The result record field or XPath nodes to map to the correct nodes in the record used by the descriptor's mappings.</p> <p>For example, if you are using XML records, use this option to map the output <code>fname</code> to <code>name/first-name</code>.</p> <p>Output arguments are not required if the interaction returns an XML result that matches the descriptor's mappings.</p>
<b>Input Result Path</b>	<p>Use this option if the EIS interaction expects the interaction arguments to be nested in the XML record.</p> <p>For example, specify <code>arguments</code>, if the arguments were to be nested under the root element <code>exec-find-order</code>, then under an <code>arguments</code> element.</p>
<b>Output Result Path</b>	The name of the EIS function that this call type (Read Object or Read All) invokes on the EIS.
<b>Properties</b>	Any properties required by your EIS platform. For example, property name <code>operation</code> (from <code>AQPlatform.QUEUE_OPERATION</code> ) and property value <code>enqueue</code> (from <code>AQPlatform.ENQUEUE</code> ).

---

## Configuring an EIS Direct Mapping

This chapter describes the various components that you must configure in order to use an EIS direct mapping.

This chapter includes the following section:

- [Introduction to EIS Direct Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 79.1 Introduction to EIS Direct Mapping Configuration

[Table 79–1](#) lists the configurable options for an EIS direct mapping.

**Table 79–1** Configurable Options for EIS Direct Mapping

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
XPath (see <a href="#">Section 121.4, "Configuring XPath"</a> )	✓	✓	✓
Use of a single node (see <a href="#">Section 121.19, "Configuring the Use of a Single Node"</a> )			✓
Simple type translator (see <a href="#">Section 121.12, "Configuring a Simple Type Translator"</a> )	✓	✓	✓
Method or direct field access at the mapping level (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Default null value at the mapping level (see <a href="#">Section 121.5, "Configuring a Default Null Value at the Mapping Level"</a> )	✓	✓	✓
Read-only mappings (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Mapping comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	
Serialized object converter (see <a href="#">Section 121.9, "Configuring a Serialized Object Converter"</a> )	✓	✓	✓
Type conversion converter (see <a href="#">Section 121.10, "Configuring a Type Conversion Converter"</a> )	✓	✓	✓
Object type converter (see <a href="#">Section 121.11, "Configuring an Object Type Converter"</a> )	✓	✓	✓
JAXB typesafe enumerated converter (see <a href="#">Section 121.13, "Configuring a JAXB Typesafe Enumeration Converter"</a> )			✓

For more information, see the following:

- [Section 77.3, "EIS Direct Mapping"](#)
- [Chapter 78, "Configuring an EIS Mapping"](#)

## Configuring an EIS Composite Direct Collection Mapping

This chapter describes the various components that you must configure in order to use an EIS composite direct collection mapping.

This chapter includes the following section:

- [Introduction to EIS Composite Direct Collection Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 80.1 Introduction to EIS Composite Direct Collection Mapping Configuration

[Table 80–1](#) lists the configurable options for an EIS composite direct collection mapping.

**Table 80–1** Configurable Options for EIS Composite Direct Collection Mapping

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
XPath (see <a href="#">Section 121.4, "Configuring XPath"</a> )	✓	✓	✓
Simple type translator (see <a href="#">Section 121.12, "Configuring a Simple Type Translator"</a> )	✓	✓	✓
Use of a single node (see <a href="#">Section 121.19, "Configuring the Use of a Single Node"</a> )	✓	✓	✓
Method or direct field access at the mapping level (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Read-only mappings (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Container policy (see <a href="#">Section 121.14, "Configuring Container Policy"</a> )	✓	✓	✓
Mapping comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	
Serialized object converter (see <a href="#">Section 121.9, "Configuring a Serialized Object Converter"</a> )	✓	✓	✓
Type conversion converter (see <a href="#">Section 121.10, "Configuring a Type Conversion Converter"</a> )	✓	✓	✓

**Table 80–1 (Cont.) Configurable Options for EIS Composite Direct Collection Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Object type converter (see <a href="#">Section 121.11</a> , "Configuring an Object Type Converter")	✓	✓	✓
JAXB typesafe enumerated converter (see <a href="#">Section 121.13</a> , "Configuring a JAXB Typesafe Enumeration Converter")	✓	✓	

For more information, see the following:

- [Section 77.4](#), "EIS Composite Direct Collection Mapping"
- [Chapter 78](#), "Configuring an EIS Mapping"

## Configuring an EIS Composite Object Mapping

This chapter describes the various components that you must configure in order to use an EIS composite object mapping.

This chapter includes the following section:

- [Introduction to EIS Composite Object Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 81.1 Introduction to EIS Composite Object Mapping Configuration

[Table 81–1](#) lists the configurable options for an EIS composite object mapping.

**Table 81–1 Configurable Options for EIS Composite Object Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
XPath (see <a href="#">Section 121.4, "Configuring XPath"</a> )	✓	✓	✓
<a href="#">Section 78.3, "Configuring Reference Descriptors"</a>	✓	✓	✓
Method or direct field access at the mapping level (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Read-only mappings (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Mapping comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	

For more information, see the following:

- [Section 77.5, "EIS Composite Object Mapping"](#)
- [Chapter 78, "Configuring an EIS Mapping"](#)





## Configuring an EIS Composite Collection Mapping

This chapter describes the various components that you must configure in order to use an EIS composite collection mapping.

This chapter includes the following section:

- [Introduction to EIS Composite Collection Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 82.1 Introduction to EIS Composite Collection Mapping Configuration

[Table 82–1](#) lists the configurable options for an EIS composite collection mapping.

**Table 82–1 Configurable Options for EIS Composite Collection Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
XPath (see <a href="#">Section 121.4, "Configuring XPath"</a> )	✓	✓	✓
Reference descriptors (see <a href="#">Section 78.3, "Configuring Reference Descriptors"</a> )	✓	✓	✓
Method or direct field access at the mapping level (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Read-only mappings (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Configuring container policy (see <a href="#">Section 121.14, "Configuring Container Policy"</a> )	✓	✓	✓
Mapping comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	

For more information, see the following:

- [Section 77.6, "EIS Composite Collection Mapping"](#)
- [Chapter 78, "Configuring an EIS Mapping"](#)



## Configuring an EIS One-to-One Mapping

This chapter describes the various components that you must configure in order to use an EIS one-to-one mapping.

This chapter includes the following sections:

- [Introduction to EIS One-to-One Mapping Configuration](#)
- [Configuring Foreign Key Pairs](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 83.1 Introduction to EIS One-to-One Mapping Configuration

Table 83–1 lists the configurable options for an EIS one-to-one mapping.

**Table 83–1 Configurable Options for EIS One-to-One Mappings**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Reference descriptors (see <a href="#">Section 78.3, "Configuring Reference Descriptors"</a> )	✓	✓	✓
Foreign key pairs (see <a href="#">Section 83.2, "Configuring Foreign Key Pairs"</a> )	✓	✓	✓
Bidirectional relationship (see <a href="#">Section 121.18, "Configuring Bidirectional Relationship"</a> )	✓	✓	✓
Method or direct field access at the mapping level (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Read-only mappings (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Private or independent relationships (see <a href="#">Section 121.7, "Configuring Private or Independent Relationships"</a> )	✓	✓	✓
Indirection (lazy loading) (see <a href="#">Section 121.3, "Configuring Indirection (Lazy Loading)"</a> )	✓	✓	✓
Mapping comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	
Selection interaction (see <a href="#">Section 78.4, "Configuring Selection Interaction"</a> )	✓	✓	✓

For more information, see the following:

- [Section 77.7, "EIS One-to-One Mapping"](#)

- [Chapter 78, "Configuring an EIS Mapping"](#)

## 83.2 Configuring Foreign Key Pairs

In a one-to-one EIS mapping, you relate a source object attribute to a target object attribute by specifying one or more pairs of source and target object fields.

In a one-to-one EIS mapping with key on source (see [Section 77.7.1, "EIS One-to-One Mappings with Key on Source"](#)) using XML records, TopLink puts the target XML field value into the source object's record as a simple value.

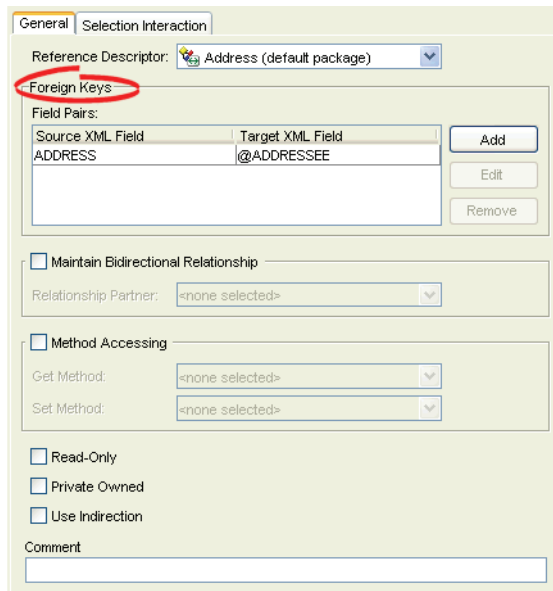
In a one-to-one EIS mapping with key on target (see [Section 77.7.2, "EIS One-to-One Mappings with Key on Target"](#)) using XML records, TopLink uses the source XML field value in the selection interaction to acquire the appropriate instance of target object.

### 83.2.1 How to Configure Foreign Key Pairs Using TopLink Workbench

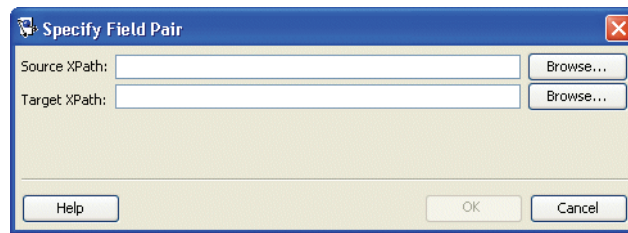
To specify the source and target XML field pairs for a one-to-one EIS mapping, use this procedure:

1. Select the one-to-one EIS mapping in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab opens.

**Figure 83–1** General Tab, Foreign Keys Field



3. Click **Add** in the Foreign Keys area to add a key pair. The Specify Field Pair dialog box appears.

**Figure 83–2 Specify Field Pair Dialog Box**

Click **Browse** to add a foreign key for the **Source XPath** and **Target XPath** fields.



## Configuring an EIS One-to-Many Mapping

This chapter describes the various components that you must configure in order to use an EIS one-to-many mapping.

This chapter includes the following sections:

- [Introduction to EIS One-to-Many Mapping Configuration](#)
- [Configuring Foreign Key Pairs](#)
- [Configuring Delete All Interactions](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 84.1 Introduction to EIS One-to-Many Mapping Configuration

Table 84–1 lists the configurable options for an EIS one-to-many mapping.

**Table 84–1 Configurable Options for EIS One-to-Many Mappings**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Reference descriptors (see <a href="#">Section 78.3, "Configuring Reference Descriptors"</a> )	✓	✓	✓
Foreign key pairs (see <a href="#">Section 84.2, "Configuring Foreign Key Pairs"</a> )	✓	✓	✓
Bidirectional relationship (see <a href="#">Section 121.18, "Configuring Bidirectional Relationship"</a> )	✓	✓	✓
Method or direct field access at the mapping level (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Read-only mappings (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Private or independent relationships (see <a href="#">Section 121.7, "Configuring Private or Independent Relationships"</a> )	✓	✓	✓
Indirection (lazy loading) (see <a href="#">Section 121.3, "Configuring Indirection (Lazy Loading)"</a> )	✓	✓	✓
Container policy (see <a href="#">Section 121.14, "Configuring Container Policy"</a> )	✓	✓	✓
Mapping comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	

**Table 84–1 (Cont.) Configurable Options for EIS One-to-Many Mappings**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Selection interaction (see <a href="#">Section 78.4, "Configuring Selection Interaction"</a> )	✓	✓	✓
Delete all interactions (see <a href="#">Section 84.3, "Configuring Delete All Interactions"</a> )	✓	✓	✓

For more information, see the following:

- [Section 77.8, "EIS One-to-Many Mapping"](#)
- [Chapter 78, "Configuring an EIS Mapping"](#)

## 84.2 Configuring Foreign Key Pairs

In a one-to-many EIS mapping, you relate a source object attribute to a target object attribute by specifying one or more pairs of source and target object fields.

In a one-to-many EIS mapping with key on source (see [Section 77.8.1, "EIS One-to-Many Mappings with Key on Source"](#)) using XML records, TopLink puts the target XML field value into the source object's record as a simple value. By default, these values are not grouped, as [Example 84–1](#) shows.

### **Example 84–1 Source Object XML Record without Grouping**

```
<employee>
  <name>Jane</name>
  <project-id>3</project-id>
  <project-id>4</project-id>
</employee>
```

If you specify more than one source and target XML field pair, you must specify a grouping element, as [Example 84–2](#) shows.

### **Example 84–2 Source Object XML Record with Grouping**

```
<employee>
  <name>Jane</name>
  <project>
    <project-id>3</project-id>
    <project-name>Project 3</project-name>
  </project>
  <project>
    <project-id>4</project-id>
    <project-name>Project 4</project-name>
  </project>
</employee>
```

In a one-to-one EIS mapping with key on target (see [Section 77.8.2, "EIS One-to-Many Mappings with Key on Target"](#)) using XML records, TopLink uses the source XML field value in the selection interaction to acquire the appropriate instances of target object.

### 84.2.1 How to Configure Foreign Key Pairs Using TopLink Workbench

To specify the source and target XML field pairs for a one-to-many EIS mapping, use this procedure:

1. Select the one-to-one EIS mapping in the **Navigator**. Its properties appear in the Editor.



2. Click the **General** tab. The General tab appears.

**Figure 84–1 Foreign Keys Field on General Tab**

Use the following information to complete the Foreign Keys fields on the **General** tab:

Field	Description
<b>Foreign Keys Located On Target</b>	Select if you are creating a one-to-many EIS mapping with key on target (see <a href="#">Section 77.8.2, "EIS One-to-Many Mappings with Key on Target"</a> ).
<b>Foreign Keys Located On Source</b>	Select if you are creating a one-to-many EIS mapping with key on source (see <a href="#">Section 77.8.1, "EIS One-to-Many Mappings with Key on Source"</a> ).
<b>Grouping Element</b>	Specify the element in which foreign key pairs are grouped in the source object's EIS record.  If you specify only one pair of source and target XML fields, this is optional.  If you specify more than one pair of source and target XML fields, this is required.
<b>Field Pairs</b>	Click <b>Add</b> to add a pair of source and target XML fields. Specify Field Pair dialog box opens. Click <b>Browse</b> to add a foreign key for the <b>Source XPath</b> and <b>Target XPath</b> fields.

## 84.3 Configuring Delete All Interactions

The TopLink query and expression framework supports delete all queries. If your JCA adapter provides access to an EIS Delete All function, you can configure a delete all interaction to support TopLink delete all queries.

### 84.3.1 How to Configure Delete All Interactions Using TopLink Workbench

To specify the DeleteAll interaction for an EIS one-to-many mapping, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **Delete All Interaction** tab. The Delete All Interaction tab appears.

**Figure 84–2 Delete All Interaction Tab**

Use the following information to enter data in each field on the Delete All Interaction tab:

Field	Description
<b>Function Name</b>	The name of the EIS function that this call type (Read Object or Read All) invokes on the EIS.
<b>Input Record Name</b>	The name passed to the JCA adapter when creating the input record.
<b>Input Root Element Name</b>	The root element name to use for the input DOM.
<b>Input Arguments</b>	The query argument name to map to the interaction field or XPath nodes in the argument record. For example, if you are using XML records, use this option to map input argument name to the XPath name/ <i>first-name</i> .

<b>Field</b>	<b>Description</b>
<b>Output Arguments</b>	<p>The result record field or XPath nodes to map to the correct nodes in the record used by the descriptor's mappings.</p> <p>For example, if you are using XML records, use this option to map the output <code>fname</code> to <code>name/first-name</code>.</p> <p>Output arguments are not required if the interaction returns an XML result that matches the descriptor's mappings.</p>
<b>Input Result Path</b>	<p>Use this option if the EIS interaction expects the interaction arguments to be nested in the XML record.</p> <p>For example, specify <code>arguments</code>, if the arguments were to be nested under the root element <code>exec-find-order</code>, then under an <code>arguments</code> element.</p>
<b>Output Result Path</b>	<p>The name of the EIS function that this call type (Read Object or Read All) invokes on the EIS.</p>
<b>Properties</b>	<p>Any properties required by your EIS platform. For example, property name <code>operation</code> (from <code>AQPlatform.QUEUE_OPERATION</code>) and property value <code>enqueue</code> (from <code>AQPlatform.ENQUEUE</code>).</p>



## Configuring an EIS Transformation Mapping

This chapter describes the various components that you must configure in order to use an EIS transformation mapping.

This chapter includes the following sections:

- [Introduction EIS Transformation Mapping Configuration](#)

For information on how to configure TopLink mappings options common to two or more mapping types, see [Chapter 121, "Configuring a Mapping"](#).

For information on how to create TopLink mappings, see [Chapter 120, "Creating a Mapping"](#).

### 85.1 Introduction EIS Transformation Mapping Configuration

[Table 85–1](#) lists the configurable options for an EIS transformation mapping.

**Table 85–1 Configurable Options for EIS Transformation Mapping**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Attribute transformer (see <a href="#">Section 121.15, "Configuring Attribute Transformer"</a> )	✓	✓	✓
Field transformer associations (see <a href="#">Section 121.16, "Configuring Field Transformer Associations"</a> )	✓	✓	✓
Method or direct field access at the mapping level (see <a href="#">Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"</a> )	✓	✓	✓
Read-only mappings (see <a href="#">Section 121.2, "Configuring Read-Only Mappings"</a> )	✓	✓	✓
Mutable mappings (see <a href="#">Section 121.17, "Configuring Mutable Mappings"</a> )	✓	✓	✓
Mapping comments (see <a href="#">Section 121.8, "Configuring Mapping Comments"</a> )	✓	✓	

For more information, see the following:

- [Section 77.9, "EIS Transformation Mapping"](#)
- [Chapter 78, "Configuring an EIS Mapping"](#)



# Part XX

---

## Using TopLink

This part describes how to associate a TopLink project with a particular instance of a data source and use it to manage persistence in your application. It contains the following chapters:

- [Chapter 86, "Introduction to Persistence Layer"](#)

This chapter provides an overview of how to use sessions, queries, and transactions in your application.





---

---

## Introduction to Persistence Layer

This chapter provides the conceptual overview of the persistence layer of a TopLink application.

This chapter includes the following section:

- [Persistence Layer Concepts](#)

### 86.1 Persistence Layer Concepts

The purpose of your application's persistence layer is to use a session (see [Section 86.1.1, "Sessions"](#)) at run time to associate mapping metadata (see [Section 2.4.3.1, "Mapping Metadata"](#)) and a data source (see [Section 86.1.2, "Data Access"](#)) in order to create, read, update, and delete persistent objects using the TopLink cache (see [Section 86.1.3, "Cache"](#)), queries and expressions (see [Section 86.1.4, "Queries and Expressions"](#)), as well as transactions (see [Section 86.1.5, "Transactions"](#)).

This section introduces the following persistence layer concepts:

- [Sessions](#)
- [Data Access](#)
- [Cache](#)
- [Queries and Expressions](#)
- [Transactions](#)

#### 86.1.1 Sessions

A session is the primary interface between the client application and the TopLink runtime, and represents the connection to the underlying data source.

For POJO projects, TopLink offers several different session types (see [Chapter 87, "Introduction to TopLink Sessions"](#)), each optimized for different design requirements and architectures. The most commonly used session is the server session—a session that clients access on the server through a client session. The server session provides a shared cache and shared connection resources.

For CMP projects, the TopLink runtime creates and uses a session internally, but your application does not acquire or use this session directly. Depending on the application server you use, you can specify some of the parameters for this internal session (see [Section 9.1.4, "JAVA-EE-CONTAINER-ejb-jar.xml File"](#)).

In JPA projects, sessions are used internally as follows:

- an `EntityManagerFactory` wraps a `ServerSession`;

- an `EntityManager` wraps a `UnitOfWork` and a `ClientSession`.

## 86.1.2 Data Access

The login (if any) associated with a session determines how the TopLink runtime connects to the project's data source.

A login includes details of data source access, such as authentication, use of connection pools, and use of external transaction controllers. A login (an instance of `Login` interface) owns a data source platform.

A platform includes options, such as binding, use of native SQL, use of batch writing, and sequencing, that are specific to a particular data source. For more information about platforms, see [Section 96.1.3, "Data Source Platform Types"](#).

For projects that do not persist to a data source, a login is not required. For projects that do persist to a data source, a login is always required.

For relational and For more information, see [Chapter 96, "Introduction to Data Access"](#)

## 86.1.3 Cache

By default, a TopLink session provides an object level cache that guarantees object identity and enhances performance by reducing the number of times the application needs to access the data source. TopLink provides a variety of cache options, including locking, refresh, invalidation, isolation, and coordination. Using cache coordination, you can configure TopLink to synchronize changes with other instances of the deployed application. You configure most cache options at the session level. You can also configure cache options on a per-query basis, or on a descriptor to apply to all queries on the reference class.

For more information, see [Chapter 102, "Introduction to Cache"](#)

## 86.1.4 Queries and Expressions

TopLink provides several object and data query types, and offers flexible options for query selection criteria, including the following:

- TopLink expressions
- JP QL
- SQL
- Stored procedures
- Query by example

With these options, you can build any type of query. Oracle recommends using predefined queries to define application queries. Predefined queries are held in the project metadata and referenced by name. This simplifies application development and encapsulates the queries to reduce maintenance costs.

Regardless of the architecture or persistent entity type, you are free to use any of the query options. Oracle JDeveloper TopLink Editor and TopLink Workbench provide the simplest way to define queries. Alternatively, you can build queries in code, using the TopLink API.

For more information, see the following:

- [Chapter 108, "Introduction to TopLink Queries"](#)
- [Chapter 110, "Introduction to TopLink Expressions"](#)

## 86.1.5 Transactions

TopLink provides the ability to write transactional code isolated from the underlying database and schema by using a **unit of work**.

The unit of work isolates changes in a transaction from other threads until it successfully commits the changes to the database. Unlike other transaction mechanisms, the unit of work automatically manages changes to the objects in the transaction, the order of the changes, and changes that might invalidate other TopLink caches. The unit of work manages these issues by calculating a minimal change set, ordering the database calls to comply with referential integrity rules and deadlock avoidance, and merging changed objects into the shared cache. In a clustered environment, the unit of work also synchronizes changes with the other servers in the coordinated cache.

If an application uses EJB entity beans, you do not access the unit of work API directly, but you still benefit from its features: the integration between the TopLink runtime and the Java EE container automatically uses the unit of work.

For more information, see [Chapter 113, "Introduction to TopLink Transactions"](#).



# Part XXI

---

## TopLink Sessions

This part describes the TopLink artifact used to associate a TopLink project with a particular instance of a data source. It contains the following chapters:

- [Chapter 87, "Introduction to TopLink Sessions"](#)  
This chapter describes each of the different TopLink session types and important session concepts.
- [Chapter 88, "Creating a Session"](#)  
This chapter contains procedures for creating TopLink sessions.
- [Chapter 89, "Configuring a Session"](#)  
This chapter explains how to configure TopLink session options common to two or more session types.
- [Chapter 90, "Acquiring and Using Sessions at Run Time"](#)  
This chapter explains how to acquire and use a TopLink session at runtime.
- [Chapter 91, "Configuring Server Sessions"](#)  
This chapter explains how to configure TopLink server and client sessions.
- [Chapter 92, "Configuring Exclusive Isolated Client Sessions for Virtual Private Database"](#)  
This chapter explains how to configure a TopLink isolated client session.
- [Chapter 93, "Configuring Historical Sessions"](#)  
This chapter explains how to configure a TopLink historical session.
- [Chapter 94, "Configuring Session Broker and Client Sessions"](#)  
This chapter explains how to configure TopLink session broker and client sessions.
- [Chapter 95, "Configuring Database Sessions"](#)  
This chapter explains how to configure a TopLink database session suitable for simple single-user, single-data source and prototyping applications.



---

---

## Introduction to TopLink Sessions

A TopLink session provides the primary access to the TopLink runtime. It is the means by which your application performs all persistence operations with the data source that contains persistent objects.

A session associates data source platform information, data source login information, and mapping metadata for a particular application. You can reuse mapping metadata in different applications by defining different sessions.

TopLink provides different session types, each optimized for different design requirements and data access strategies. You can combine different session types in the same application.

This chapter includes the following sections:

- [Session Types](#)
- [Session Concepts](#)
- [Server and Client Sessions](#)
- [Unit of Work Sessions](#)
- [Isolated Client Sessions](#)
- [Historical Sessions](#)
- [Session Broker and Client Sessions](#)
- [Database Sessions](#)
- [Remote Sessions](#)
- [Sessions and the Cache](#)
- [Session API](#)

### 87.1 Session Types

Table 87–1 lists the session types that you can use in a POJO TopLink application and classifies them as basic or advanced. See [Section 87.2.11, "Sessions and CMP"](#) for information on using Oracle TopLink with CMP.

**Table 87-1 TopLink Session Types**

Session Type	Description	Oracle JDeveloper	TopLink Workbench	Java
Server and client sessions (see <a href="#">Section 87.3</a> , "Server and Client Sessions")	Server sessions provide session management to a single data source (including shared object cache and connection pools) for multiple clients in a three-tier architecture using database or EIS platforms. This is the most flexible, scalable, and commonly used session. You acquire a client session from a server session at run time to provide access to a single data source for each client.	✓	✓	✓
Unit of work sessions (see <a href="#">Section 87.4</a> , "Unit of Work Sessions")	Acquired from any session type (directly, or by way of an external transaction controller) to transactionally modify objects.			✓
Isolation client sessions (see <a href="#">Section 87.5</a> , "Isolated Client Sessions")	A special type of client session that uses a session cache isolated from the shared object cache of its parent server session.			✓
Historical sessions (see <a href="#">Section 87.6</a> , "Historical Sessions")	A special type of client session that provides a read-only snapshot of object versions as of a specified time and uses a session cache isolated from the shared object cache of its parent server session.			✓
Session broker and client sessions (see <a href="#">Section 87.7</a> , "Session Broker and Client Sessions")	Provides session management to multiple data sources for multiple clients by aggregating two or more server sessions (can also be used with database sessions). You acquire a client session from a session broker at run-time to provide access to all the data sources managed by the session broker for each client.	✓	✓	✓
Database sessions (see <a href="#">Section 87.8</a> , "Database Sessions")	Provides session management to a single database for a single client suitable for simple or two-tiered applications. Oracle does not recommend this session type in three-tiered applications because it does not offer the same flexibility and scalability as the server session.	✓	✓	✓
Remote sessions (see <a href="#">Section 87.9</a> , "Remote Sessions")	A client-side session that communicates over RMI with a corresponding dedicated client session and shared server session. Remote sessions handle object identity and marshalling and unmarshalling between client-side and server-side.			✓

For more information, see the following:

- [Chapter 88, "Creating a Session"](#)
- [Chapter 89, "Configuring a Session"](#)
- [Chapter 90, "Acquiring and Using Sessions at Run Time"](#)

## 87.2 Session Concepts

This section describes concepts unique to TopLink sessions, including the following:

- [Session Architecture](#)
- [Session Configuration and the sessions.xml File](#)
- [Session Customization](#)
- [Acquiring a Session at Run Time with the Session Manager](#)



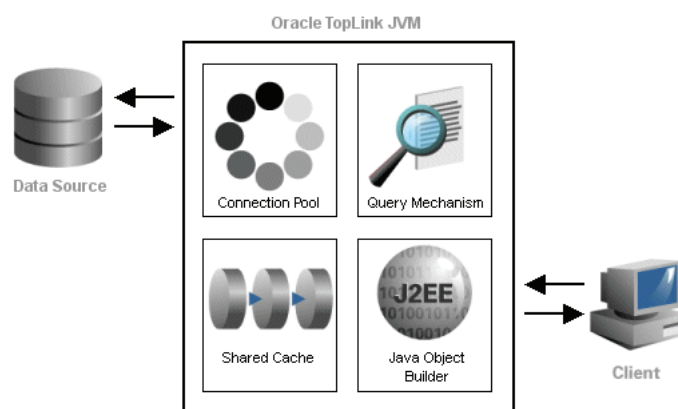
- Managing Session Events with the Session Event Manager
- Logging
- Profiler
- Integrity Checker
- Exception Handlers
- Registering Descriptors
- Sessions and CMP
- Sessions and Sequencing

## 87.2.1 Session Architecture

As Figure 87–1 illustrates, a session instance is composed of the following components:

- Object Cache
- Connection Pools
- Query Mechanism
- Java Object Builder

**Figure 87–1 Simple TopLink Session Architecture**



How these session components are implemented and how they interact depends on the type of session. For example, for server and client sessions, the server session provides a connection pool and shared object cache on behalf of all client sessions acquired from it.

### 87.2.1.1 Object Cache

TopLink sessions provide an object cache. This cache, known as the **session cache**, retains information about objects that are read from or written to the database, and is a key element for improving the performance of a TopLink application.

Typically, a server session's object cache is shared by all client sessions acquired from it. That is, for a Server session `myServerSession`, each client session acquired by calling server session method `acquireClientSession` shares the same object cache as `myServerSession`.

Isolated and historical sessions provide their own session cache isolated from the shared object cache of their parent server session. For more information, see [Section 87.5, "Isolated Client Sessions"](#) and [Section 87.6, "Historical Sessions"](#).

You can easily manage concurrent access to this shared cache by using a unit of work session acquired from any session. For more information, see [Section 87.4, "Unit of Work Sessions"](#).

For more information, see [Section 87.10, "Sessions and the Cache"](#).

### 87.2.1.2 Connection Pools

A **connection pool** is a collection of reusable connections to a single data source.

---

---

**Note:** To simultaneously access multiple databases from within a single session, use a session broker. For more information, see [Section 87.7, "Session Broker and Client Sessions"](#).

---

---

Because creating a data source connection is usually expensive, a properly configured connection pool significantly improves performance.

You can configure your session to use internal connection pools provided by TopLink or external connection pools provided by a JDBC driver or Java EE container. By default, TopLink uses internal connection pools.

Internal connection pools are usually used in non-EJB applications, or when an external transaction controller (JTA) is not used. If you configure your session to use internal connection pools, you can configure its default read and write connection pools. You can create special purpose connection pools for application-specific purposes (named connection pools) or exclusively for sequencing (sequence connection pool). For more information, see [Section 96.1.6.1, "Internal Connection Pools"](#).

External connection pools are usually used in EJB applications and when an external transaction controller (JTA) is used. For more information, see [Section 96.1.6.2, "External Connection Pools"](#).

For more information about data access configuration in general, see [Chapter 96, "Introduction to Data Access"](#).

### 87.2.1.3 Query Mechanism

At run time, your application uses a session to perform all persistence operations: creating, reading, updating, and deleting objects. You perform these operations using TopLink queries and expressions with the session query API.

For more information, see [Chapter 108, "Introduction to TopLink Queries"](#).

### 87.2.1.4 Java Object Builder

When you use object-level read queries, TopLink automatically builds Java objects from the data retrieved. When you use object-level write queries, TopLink automatically converts the affected Java objects into the appropriate data native to your data source.

## 87.2.2 Session Configuration and the sessions.xml File

TopLink provides two ways to configure your sessions: through Java code using the `Session` API, or using TopLink Workbench to build a session configuration file, the `sessions.xml` file.

In most cases, you configure sessions for the application using the `sessions.xml` file. This file is an Extensible Markup Language (XML) file that contains all sessions that are associated with the application. The `sessions.xml` file can contain any number of sessions and session types.

Oracle recommends that you use the `sessions.xml` file to deploy a TopLink application, because it provides the following advantages:

- It is easy to create and maintain in TopLink Workbench.
- It is easy to troubleshoot.
- It provides access to most session configuration options.
- It offers excellent flexibility, including the ability to modify deployed applications without recompiling.

For more information on creating a session in the `sessions.xml` file, see [Section 88.1, "Introduction to the Session Creation"](#).

## 87.2.3 Session Customization

You can customize a session at run time by specifying a session customizer—a Java class that implements the `oracle.toplink.tools.sessionconfiguration.SessionCustomizer` interface and provides a default (zero-argument) constructor.

Since the `sessions.xml` file is not used for CMP projects, use `oracle.toplink.ejb.cmp.DeploymentCustomization` interface as a customizer when creating a CMP project to specify your database login information.

You use a session customizer to customize a session at run time through code API similar to how you use an amendment method to customize a descriptor (see [Section 16.2.7, "Amendment and After-Load Methods"](#)).

For more information, see [Section 89.8, "Configuring a Session Customizer Class"](#).

## 87.2.4 Acquiring a Session at Run Time with the Session Manager

The TopLink session manager lets you build a series of sessions that are maintained under a singleton object called the session manager.

The session manager is a static utility class that loads TopLink sessions from the `sessions.xml` file (see [Section 87.2.2, "Session Configuration and the sessions.xml File"](#)), caches the sessions by name in memory, and provides a single access point for TopLink sessions.

At run time, TopLink will attempt to load the `sessions.xml` file from the two following default resource names: `sessions.xml` and `META-INF/sessions.xml`. Refer to [Chapter 10, "Packaging a TopLink Application"](#) for additional information.

The session manager supports the following session types:

- `ServerSession` (see [Section 87.3, "Server and Client Sessions"](#))
- `SessionBroker` (see [Section 87.7, "Session Broker and Client Sessions"](#))
- `DatabaseSession` (see [Section 87.8, "Database Sessions"](#))

The session manager has two main functions: it creates instances of these sessions and it ensures that only a single instance of each named session exists for any instance of a session manager.

The session manager instantiates sessions as follows:

1. The client application requests a session by name.
2. The session manager looks up the session name in the `sessions.xml` file. If the session name exists, the session manager instantiates the specified session; otherwise, it raises an exception.
3. After instantiation, the session remains viable until you shut down the application.

Once you have a session instance, you can use it to acquire additional types of sessions for special tasks. For example, you can acquire a unit of work from any session to perform transactional operations. You can acquire a client session from a server session to perform client operations in a three-tier architecture.

For more information, see [Chapter 90, "Acquiring and Using Sessions at Run Time"](#).

## 87.2.5 Managing Session Events with the Session Event Manager

Sessions raise **session events** for most session operations. Session events help you debug or coordinate the actions of multiple sessions.

The session event manager handles information about session events. Applications register session event listeners with the session event manager to receive session events.

For example, session event listeners play an important role in the configuration of isolated sessions (see [Chapter 92, "Configuring Exclusive Isolated Client Sessions for Virtual Private Database"](#)). In an isolated session, if the TopLink runtime raises a `SessionEvent.NoRowsModified` event, it is handled by your `SessionEventListener` (see [Section 92.4, "Using NoRowsModifiedSessionEvent Event Handler"](#)). This event listener is your opportunity to determine whether the update failure was due to a security violation (in which case you should not retry the operation) or due to an optimistic lock issue (in which case a retry may be appropriate). See [Section 87.2.6, "Logging"](#) for information on adding logging to your event listeners.

Another example is the use of session event listeners to configure proxy authentication in Oracle Database. (see [Section 98.8, "Configuring Oracle Database Proxy Authentication"](#)).

### 87.2.5.1 Session Event Manager Events

The session event manager supports the session events listed in the following tables:

- [Table 87–2, "Session Events"](#)
- [Table 87–3, "Unit of Work Events"](#)

**Table 87–2 Session Events**

Event	Description
MissingDescriptor	Raised if a descriptor is missing for a class being persisted. You can use this event to lazy register the descriptor or set of descriptors.
MoreRowsDetected	Raised when a <code>ReadObjectQuery</code> detects more than one row returned from the database. This event can indicate a possible error condition in your application.

**Table 87-2 (Cont.) Session Events**

<b>Event</b>	<b>Description</b>
NoRowsModified	Raised after update or delete SQL has been sent to the database and a row count of zero is returned.
OutputParametersDetected	Raised after a stored procedure call with output parameters executes. This event enables you to retrieve a result set and output parameters from a single stored procedure.
PostAcquireClientSession	Raised after a client Session is acquired
PostAcquireConnection	Raised after acquiring a connection
PostAcquireExclusiveConnection	Raised when a client Session, with isolated data, acquires an exclusive connection.
PostBeginTransaction	Raised after a database transaction starts
PostCommitTransaction	Raised after a database transaction commits
PostConnect	Raised after connecting to the database
PostExecuteQuery	Raised after the execution of every query on the session
PostLogin	Raised after the Session initializes and acquires connections
PostReleaseClientSession	Raised after releasing a client Session
PostRollbackTransaction	Raised after a database transaction rolls back
PreBeginTransaction	Raised before a database transaction starts
PreCommitTransaction	Raised before a database transaction commits
PreExecuteQuery	Raised before the execution of every query on the session
PreLogin	Raised before the Session initializes and acquires connections
PreReleaseClientSession	Raised before releasing a client Session
PreReleaseConnection	Raised before releasing a connection
PreReleaseExclusiveConnection	Raised before a client Session, with isolated data, releases its exclusive connection.
PreRollbackTransaction	Raised before a database transaction rolls back

**Table 87-3 Unit of Work Events**

<b>Event</b>	<b>Description</b>
PostAcquireUnitOfWork	Raised after a UnitOfWork is acquired
PostCalculateUnitOfWorkChangeSet	Raised after the commit has begun on the UnitOfWork and after the changes are calculated. The UnitOfWorkChangeSet, at this point, will contain change sets without the version fields updated and without identity field type primary keys. These will be updated after the insert, or update, of the object.
PostCommitUnitOfWork	Raised after a UnitOfWork commits
PostDistributedMergeUnitOfWorkChangeSet	Raised after a UnitOfWork change set has been merged when that change set has been received from a distributed session.
PostMergeUnitOfWorkChangeSet	Raised after a UnitOfWork change set has been merged.
PostReleaseUnitOfWork	Raised on a UnitOfWork after it is released.
PostResumeUnitOfWork	Raised on a UnitOfWork after it resumes.
PreCalculateUnitOfWorkChangeSet	Raised after the commit has begun on the UnitOfWork but before the changes are calculated.
PreCommitUnitOfWork	Raised before a UnitOfWork commits.
PreDistributedMergeUnitOfWorkChangeSet	Raised before a UnitOfWork change set has been merged when that change set has been received from a distributed session.

**Table 87-3 (Cont.) Unit of Work Events**

Event	Description
PreMergeUnitOfWorkChangeSet	Raised before a <code>UnitOfWork</code> change set has been merged.
PrepareUnitOfWork	Raised after the a <code>UnitOfWork</code> flushes its SQL, but before it commits its transaction.
PreReleaseUnitOfWork	Raised on a <code>UnitOfWork</code> before it is released.

### 87.2.5.2 Session Event Listeners

You can create session event listeners in two ways: either by implementing the `SessionEventListener` interface, or by extending the `SessionEventAdapter` class.

To register a `SessionEventListener` for session events, register it with a session using the `SessionEventManager` method `addListener`.

For more information, see [Section 89.10, "Configuring Session Event Listeners"](#).

## 87.2.6 Logging

You can configure a session to write run-time information to a TopLink log. This information includes status, diagnostic, SQL, and, when profiling is enabled, performance data (see [Section 12.3, "Measuring TopLink Performance with the TopLink Profiler"](#) or [Section 12.4, "Measuring TopLink Performance with the Oracle Dynamic Monitoring System \(DMS\)"](#)).

Logging options are configurable at the session level (see [Section 89.4, "Configuring Logging"](#)).

---

**Note:** To facilitate debugging, you can add logging to your listeners to only log the events that are of the interest to your application. Within the session context, use the following logging utility:

```
Session.getSessionLog().log(int level, String message)
```

Without the session context, use the following logging utility:

```
AbstractSessionLog.getLog().log(int level, String message)
```

Both the `getSessionLog` and `getLog` methods return a session log (an instance of a `SessionLog` interface) loaded with an accessor's log messages and SQL. Then the session log performs logging at the level that you specify.

For more information on session event listeners, see [Section 87.2.5.2, "Session Event Listeners"](#).

---

### 87.2.6.1 Log Types

TopLink supports the following types of logging:

- [TopLink Native Logging](#)
- [java.util Logging](#)
- [Server Logging](#)

#### 87.2.6.1.1 TopLink Native Logging

TopLink native logging is the default session log type. It is provided by `oracle.toplink.logging.DefaultSessionLog`. [Example 87-1](#) shows a typical TopLink native log message.

You can configure TopLink native logging options using TopLink Workbench (see [Section 89.4.1, "How to Configure Logging Using TopLink Workbench"](#)).

### **Example 87-1 Sample TopLink Log Message**

```
[TopLink Info]: DATE
TIME-DatabaseSession(12345)-Thread(12345)-TopLink, version: Oracle TopLink - 10g (Build
031203)
[TopLink Config]: DATE
TIME-DatabaseSession(12345)-Thread(12345)-Connection(12345)-
  connecting(DatabaseLogin(
    platform=>Oracle9Platform
    user name=> "username"
    datasource URL=> "jdbc:oracle:thin:@144.23.214.115:1521:toplink"
  ))
[TopLink Config]: DATE
TIME-DatabaseSession(12345)-Thread(12345)-Connection(12345)-
  Connected: jdbc:oracle:thin:@144.23.214.115:1521:toplink
  User: USERNAME
  Database: Oracle Version: Oracle9i Enterprise Edition - Production
  With the Partitioning, OLAP and Oracle Data Mining options
  JServer Release 9.2.0.3.0 - Production
  Driver: Oracle JDBC driver Version: 9.2.0.3.0
[TopLink Info]: DATE
TIME-DatabaseSession(12345)-Thread(12345)-loggingTestSession login
successful
```

### **87.2.6.1.2 java.util Logging**

This type of logging makes TopLink conform to the `java.util.logging` package. It is provided by `oracle.toplink.logging.JavaLog`. Logging options are configured in the `<JRE_HOME>/lib/logging.properties` file. Messages are written to any number of destinations based on this configuration. [Example 87-2](#) shows a typical `java.util.logging` log message.

For more information on using `java.util.logging` package, see [Section 89.4.5, "How to Configure a Session to use the java.util.logging Package"](#).

### **Example 87-2 Sample java.util.logging Log Messages**

```
Dec 9, 2003 2:05:05 PM oracle.toplink.loggingTestSession DatabaseSession(32603767) Thread(10)
INFO: TopLink, version: Oracle TopLink - 10g (10.0.3) Developer Preview (Build 031203)
Dec 9, 2003 2:05:07 PM oracle.toplink.loggingTestSession.connection DatabaseSession(32603767)
Connection(927929) Thread(10)
CONFIG: connecting(DatabaseLogin(
  platform=>Oracle9Platform
  user name=> "coredev8"
  datasource URL=> "jdbc:oracle:thin:@144.23.214.115:1521:toplink"
))
Dec 9, 2003 2:05:08 PM oracle.toplink.loggingTestSession.connection DatabaseSession(32603767)
Connection(927929) Thread(10)
CONFIG: Connected: jdbc:oracle:thin:@144.23.214.115:1521:toplink
  User: COREDEV8
  Database: Oracle Version: Oracle9i Enterprise Edition Release 9.2.0.3.0 - Production
  With the Partitioning, OLAP and Oracle Data Mining options
  JServer Release 9.2.0.3.0 - Production
  Driver: Oracle JDBC driver Version: 9.2.0.3.0
Dec 9, 2003 2:05:08 PM oracle.toplink.loggingTestSession DatabaseSession(32603767) Thread(10)
INFO: loggingTestSession login successful
```

### 87.2.6.1.3 Server Logging

Server logging is used to integrate TopLink logging with an application server log.

The TopLink runtime determines the server log type to use given the server platform you configure when you create your project (Section 116.1, "Introduction to the Project Creation").

For example, if your project uses the WebLogic platform, TopLink uses the `oracle.toplink.platform.server.wls.WlsLog`; if your project uses the OC4J platform, TopLink uses the `oracle.toplink.platform.server.oc4j.OjdlLog`.

For more information, see the following:

- Section 89.4.4, "How to Configure Logging in a Java EE Container"

### 87.2.6.2 Log Output

If you are using TopLink native logging, you can configure TopLink to write log messages to a file or to the console (see Section 89.4, "Configuring Logging").

If you are using `java.util.logging`, TopLink writes log messages to the destinations you configure in the `<JRE_HOME>/lib/logging.properties` file (see Section 89.4.5, "How to Configure a Session to use the `java.util.logging` Package").

If you are using server logging, TopLink writes log messages to the application server's log file (there is no separate TopLink log file in this case).

### 87.2.6.3 Log Level

You can control the amount and detail of log output by configuring the log level (in ascending order of information) in the following way:

- SEVERE—Logs exceptions indicating TopLink cannot continue, as well as any exceptions generated during login. This includes a stack trace.
- WARNING—Logs exceptions that do not force TopLink to stop, including all exceptions not logged with severe level. This does not include a stack trace.
- INFO (default)—Logs the login/logout per server session, including the user name. After acquiring the session, detailed information is logged.
- CONFIG—Logs only login, JDBC connection, and database information.
- FINE—Logs SQL (including thread information).
- FINER—Similar to warning. Includes stack trace.
- FINEST—Includes additional low level information
- ALL—Logs everything.

By default, TopLink logs at the `oracle.toplink.logging.SessionLog.INFO` level so that some information is logged by default.

At run time, set the log level using `Session` method `setLogLevel`, passing in one of the log level constants provided by `oracle.toplink.logging.SessionLog`.

### 87.2.6.4 Logging SQL

In a relational project, TopLink accesses the database using SQL strings that it generates internally. This feature enables applications to use the session methods or query objects without having to perform their own SQL translation.

If, for debugging purposes, you want to review a record of the SQL that is sent to the database, set the session log level to



`oracle.toplink.logging.SessionLog.FINE`—the session will log all executed SQL to the session log.

[Example 87-3](#) shows how to configure the log destination using the `setLog` method on the session.

### Example 87-3 Configuring the Log Destination

```
private static SessionEventListener buildListener() {
    return new SessionEventAdapter() {
        public void preLogin(SessionEvent event) {
            File file = new
                File("C:\\oracle\\904\\toplink\\examples\\jdev\\2-TierEmployee\\toplin
k.log");
            try {
                System.out.println("FILE: " + file.getAbsolutePath());
                FileWriter writer = new FileWriter(file);
                event.getSession().setLog(writer);
            } catch (IOException ioe) {
                ioe.printStackTrace();
                throw new RuntimeException("Failed to setup logging to: " +
file.getAbsolutePath());
            }
        }
    };
}
```

#### 87.2.6.5 Logging Chained Exceptions

The logging chained exception facility enables you to log causality when one exception causes another as part of the standard stack back-trace. Causal chains appear automatically in your logs.

#### 87.2.6.6 Logging Inside a Java EE Container

When you deploy a TopLink-enabled application to an application server or EJB container, TopLink CMP and EclipseLink JPA default to `ServerLog` with no log level so that TopLink uses the configuration in `j2ee-logging.xml`.

For more information, see [Section 89.4, "Configuring Logging"](#).

#### 87.2.6.7 Logging Outside of a Java EE Container

When you deploy a TopLink-enabled application outside of an EJB container, the logging defaults revert to `DefaultSessionLog` and `INFO` log level.

If you are using TopLink native logging (to a file) or the `java.util.logging` package outside of a Java EE container, you control logging using the `<JRE_HOME>/lib/logging.properties` file.

For more information, see the following:

- [Section 89.4, "Configuring Logging"](#)
- [Section 89.4.5, "How to Configure a Session to use the java.util.logging Package"](#)

## 87.2.7 Profiler

The TopLink session provides profiling API that lets you identify performance bottlenecks in your application (see [Section 89.6, "Configuring a Performance Profiler"](#)). When enabled, the profiler logs a summary of the performance statistics for every query that the application executes.

TopLink allows you to measure application performance using [TopLink Profiler](#) or [Oracle Dynamic Monitoring System \(DMS\)](#).

### 87.2.7.1 TopLink Profiler

The TopLink profiler is a high-level logging service. Instead of logging SQL statements, the profiler logs a summary of each query you execute. The summary includes a performance breakdown of the query that lets you identify performance bottlenecks. The profiler also provides a report summarizing the query performance for an entire session.

Access profiler reports and profiles through the **Profile** tab in the TopLink Web client, or create your own application or applet to view the profiler logs. For more information, see [Section 12.3.2, "How to Access the TopLink Profiler Results"](#).

For more information, see [Section 12.3, "Measuring TopLink Performance with the TopLink Profiler"](#).

### 87.2.7.2 Oracle Dynamic Monitoring System (DMS)

Oracle DMS is a library that enables application and system developers to use a variety of DMS sensors to measure and export customized performance metrics for specific software components (called nouns).

TopLink includes DMS instrumentation in essential objects to provide efficient monitoring of runtime data in TopLink enabled applications, including both Java EE and non-Java EE applications.

By enabling DMS profiling in a TopLink application, you can collect and easily access run-time data that can help you with application administration tasks and performance tuning.

You can easily access DMS data at run time using a management application that supports the Java Management Extensions (JMX) API (see [Section 12.4.2, "How to Access Oracle DMS Profiler Data Using JMX"](#)), or using any Web browser and the DMS Spy servlet (see [Section 12.4.3, "How to Access Oracle DMS Profiler Data Using the DMS Spy Servlet"](#)).

For more information, see [Section 12.4, "Measuring TopLink Performance with the Oracle Dynamic Monitoring System \(DMS\)"](#).

## 87.2.8 Integrity Checker

When you log into a session, TopLink initializes and validates the descriptors you registered with it. By configuring the integrity checker, you can customize this validation process.

For more information, see [Section 89.11, "Configuring the Integrity Checker"](#).

## 87.2.9 Exception Handlers

Exception handlers allow any exception that occurs in a session to be caught and processed. Exception handlers can be used for debugging purposes, or to resolve database timeouts or failures.

To use exception handlers, register an implementor of the `oracle.toplink.exceptions.ExceptionHandler` interface with the session (see [Section 89.7, "Configuring an Exception Handler"](#)).

If an exception occurs during a session operation, such as executing a query, the exception is passed to the exception handler. The exception handler can either rethrow

the exception, or handle the exception and retry the operation. When handling exceptions, ensure that the following conditions are met:

- If you are performing a write query and you are within a transaction, you should not retry the operation.
- If you are performing a read query, you may retry the operation, and, if successful, return the query result.

If your exception handler cannot proceed, you should throw an appropriate application-specific exception.

For more information on the types of exceptions that TopLink can throw, see [Appendix A, "Troubleshooting a TopLink Application"](#).

### 87.2.10 Registering Descriptors

You use a session to perform persistence operations on the objects described by TopLink mapping metadata represented as a TopLink project (see [Chapter 15, "Introduction to Projects"](#)). Each session must therefore be associated with the descriptors of at least one TopLink project. You associate descriptors with a session by registering them with the session.

The preferred way to register descriptors with a session is to use Oracle JDeveloper TopLink Editor or TopLink Workbench to configure the session with a mapping project (see [Section 89.2, "Configuring a Primary Mapping Project"](#) and [Section 89.5, "Configuring Multiple Mapping Projects"](#)).

### 87.2.11 Sessions and CMP

Although TopLink is an integral part of a Java EE application, in most cases the client does not interact with TopLink directly. Instead, TopLink features are invoked indirectly by way of EJB container callbacks.

In a CMP TopLink project, you do not explicitly create, configure, or acquire a session. The TopLink runtime creates, configures, acquires and uses a session itself internally. For more information, see [Section 2.9.3, "Creating Session Metadata"](#). Similarly, in a CMP TopLink project, how metadata is deployed depends on the EJB container and application server you use (see [Section 2.9.4, "Deploying Metadata"](#)).

### 87.2.12 Sessions and Sequencing

An essential part of maintaining object identity is managing the assignment of unique values to distinguish one instance from another. For more information, see [Section 15.2.6, "Projects and Sequencing"](#).

Sequencing options you configure in a `sessions.xml` (or `project.xml`) file determine the type of sequencing that TopLink uses.

In a CMP project, you do not configure a `sessions.xml` file directly: in this case you must configure the sequence type in the `project.xml` file (see [Section 20.3, "Configuring Sequencing at the Project Level"](#)).

In a POJO project, you can use session-level sequence configuration to override project-level sequence configuration, on a session-by-session basis, if required (see [Section 98.4, "Configuring Sequencing at the Session Level"](#)).

After configuring the sequence type at the session (or project) level, for each descriptor you must also configure sequencing options for that descriptor to use sequencing (see [Section 16.2.10, "Descriptors and Sequencing"](#)).

## 87.3 Server and Client Sessions

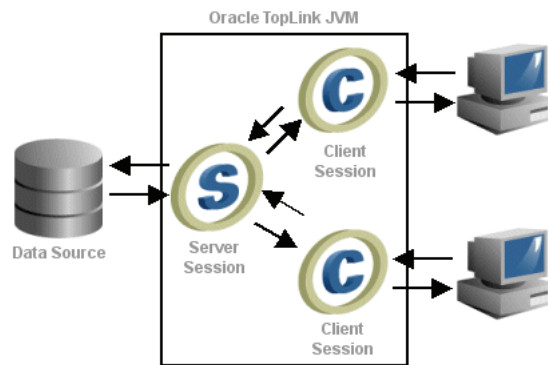
A server session manages the server side of client/server communications, providing shared resources, including a shared object cache and connection pools to a single data source.

A client session is a server-side communications mechanism that works together with the server session to provide the client/server connection. You acquire client sessions from a server session at run time as required. By default, a client session shares the session cache of its parent server session. Each client session serves one client. A client session communicates with the server session on behalf of the client application.

Each client session can have only one associated server session, but a server session can support any number of client sessions.

As [Figure 87–2](#) illustrates, together, the client session and server session provide a three-tier architecture that you can scale easily, by adding more client sessions. A server session is the most common TopLink session type because it supports this three-tier architecture that is common in enterprise applications. Because of this scalability, Oracle recommends that you use the three-tier architecture to build your TopLink applications.

**Figure 87–2 Typical TopLink Server Session with Client Session Architecture**



This section explains the advantages of using server sessions and client sessions in your TopLink application, including the following:

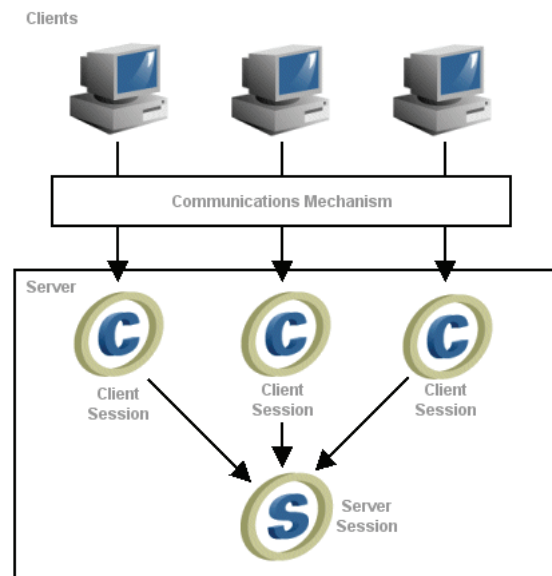
- [Three-Tier Architecture Overview](#)
- [Advantages of the TopLink Three-Tier Architecture](#)

For more information, see the following:

- [Section 88.4, "Creating a Server Session"](#)
- [Chapter 91, "Configuring Server Sessions"](#)
- [Section 90.3, "Acquiring a Session from the Session Manager"](#)
- [Section 90.4, "Acquiring a Client Session"](#)

### 87.3.1 Three-Tier Architecture Overview

In a TopLink three-tier architecture, client sessions and server sessions both reside on the server. Client applications access the TopLink application through a client session, and the client session communicates with the database using the server session.

**Figure 87-3 Server Session and Client Session Usage**

### 87.3.2 Advantages of the TopLink Three-Tier Architecture

Although the server session and the client session are two different session types, you can treat them as a single unit in most cases, because they are both required to provide three-tier functionality to the application. The server session provides the client session to client applications, and also supplies the majority of the session functionality.

This section discusses some of the advantages and general concepts associated with the TopLink three-tier design, including the following:

- [Shared Resources](#)
- [Providing Read Access](#)
- [Providing Write Access](#)
- [Security and User Privileges](#)
- [Concurrency](#)
- [Connection Allocation](#)

#### 87.3.2.1 Shared Resources

The three-tier design enables multiple clients to share persistent resources. The server session provides its client sessions with a shared live object cache, read and write connection pooling, and parameterized named queries. Client sessions also share descriptor metadata.

You can use client sessions and server sessions in any application server architecture that allows for shared memory and supports multiple clients. These architectures can include HyperText Markup Language (HTML), Servlet, JavaServer Pages (JSP), Remote Method Invocation (RMI), Common Object Request Broker Architecture (CORBA), Web services, and EJB.

To support a shared object cache, client sessions must do the following:

- Implement any changes to the database with the TopLink unit of work.

- Share a common database login for reading (you can implement separate logins for writing).

### 87.3.2.2 Providing Read Access

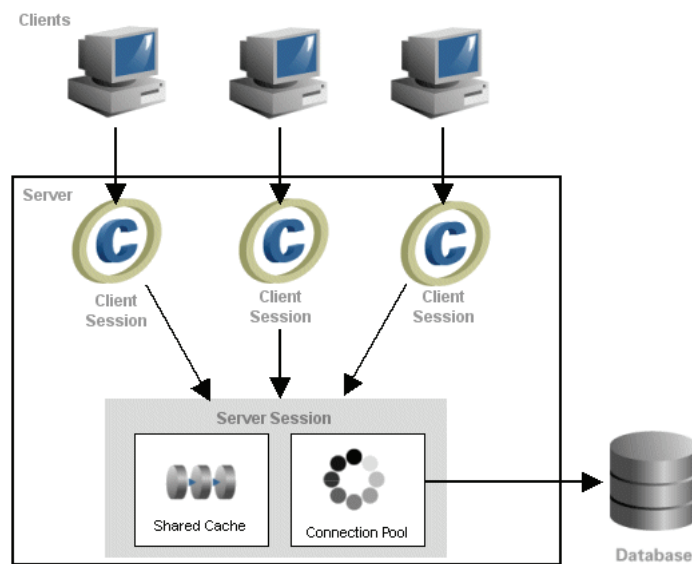
To read objects from the database, the client must first acquire a client session from the server session. Acquiring a client session gives the client access to the session cache and the database through the server session. The server session behaves as follows:

- If the object or data is in the session cache, then the server session returns the information back to the client.
- If the object or data is not in the cache, then the server session reads the information from the database and stores the object in the session cache. The objects are then available for retrieval from the cache.

Because a server session processes each client request in a separate thread, this enables multiple clients to access the database connection pool concurrently.

Figure 87–4 illustrates how multiple clients read from the database using the server session.

**Figure 87–4 Multiple Client Sessions Reading the Database Using the Server Session**



To read objects from the database using a client session, do the following:

1. Acquire a Session from the Server:

```
Server server =
    (Server) SessionManager.getManager().getSession(
        sessionName, MyServerSession.class.getClassLoader()
    );
Session clientSession = (Session) server.acquireClientSession();
```

For more information, see [Chapter 90, "Acquiring and Using Sessions at Run Time"](#).

2. Use the Session object to perform read operations (for more information, see [Chapter 108, "Introduction to TopLink Queries"](#) and [Chapter 110, "Introduction to TopLink Expressions"](#)).

---



---

**Note:** Oracle recommends that you do not use the Server session object directly to read objects from the database.

---



---

### 87.3.2.3 Providing Write Access

Because the client session disables all database modification methods, a client session cannot create, change, or delete objects directly. Instead, the client must obtain a unit of work from the client session to perform database modification methods.

To write to the database, the client acquires a client session from the server session and then acquires a unit of work within that client session. The unit of work acts as an exclusive transactional object space, and also ensures that any changes that are committed to the database also occur in the session cache.

---



---

**Note:** Although client sessions are thread-safe, do not use them to write across multiple threads. Multithread write operations from the same client session can result in errors and a loss of data. For more information, see [Section 87.3.2.5, "Concurrency"](#).

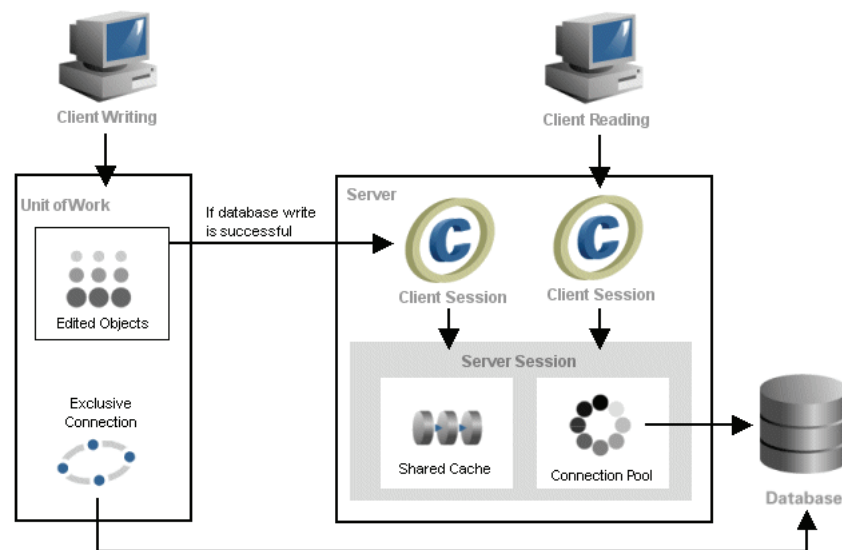
---



---

Figure 87–5 illustrates how to write to the database using a client session acquired from a server session.

**Figure 87–5 Writing with Client Sessions and Server Sessions**



To write to the database using a unit of work, use this procedure:

1. Acquire a session from the server session:

```
Server server =
    (Server) SessionManager.getManager().getSession(
        sessionName, MyServerSession.class.getClassLoader()
    );
Session clientSession = (Session) server.acquireClientSession();
```

For more information, see [Chapter 90, "Acquiring and Using Sessions at Run Time"](#).

2. Acquire a `UnitOfWork` object from the `Session` object.

```
UnitOfWork uow = clientSession.acquireUnitOfWork();
```

For more information, see [Section 87.4, "Unit of Work Sessions"](#).

3. Use the unit of work to perform the required updates and then commit the `UnitOfWork`.

For more information, see the following:

- [Chapter 108, "Introduction to TopLink Queries"](#)
- [Chapter 110, "Introduction to TopLink Expressions"](#)
- [Chapter 113, "Introduction to TopLink Transactions"](#)

#### 87.3.2.4 Security and User Privileges

You can define several different server sessions in your application to support users with different data access rights. For example, your application may serve a group called "Managers," who has access rights to salary information, and a group called "Employees," who do not. Because each session you define in the `sessions.xml` file has its own login information, you can create multiple sessions, each with its own login credentials, to meet the needs of both of these groups.

When you use internal TopLink connection pools (see [Section 96.1.6, "Connection Pools"](#)), each server session provides a read connection pool and a write connection pool. All read queries use connections from the read connection pool and all queries that write changes to the data store use connections from the write connection pool. This ensures that connections for one session are kept separate from the connections used in another.

To further isolate users from one another, you can use an isolated session: a special type of client session that provides its own session cache isolated from the shared object cache of its parent server session to provide improved user-based security, or to avoid caching highly volatile data. For more information, see [Section 87.5, "Isolated Client Sessions"](#).

#### 87.3.2.5 Concurrency

The server session supports concurrent clients by providing each client with a dedicated thread of execution. Dedicated threads enable clients to operate asynchronously—that is, client processes execute as they are called and do not wait for other client processes to complete.

TopLink safeguards thread safety with a concurrency manager. The concurrency manager ensures that no two threads interfere with each other when performing operations such as creating new objects, executing a transaction on the database, or accessing value holders.

For more information about handling concurrency issues, see [Section 102.2.3, "Handling Stale Data"](#).

#### 87.3.2.6 Connection Allocation

When you instantiate the server session, it creates a pool of data source connections. The session then manages the connection pool based on your session configuration, and shares the connections among its client sessions. When the client session releases the connection, the server session recovers the connection and makes it available to other client processes. Reusing connections reduces the number of connections



required by the application and allows a server session to support a larger number of clients.

The server session provides connections to client sessions as needed. By default, the server session does not allocate a data source connection for a client session until a transaction starts (a lazy data source connection). Alternatively, you can acquire a client session that allocates a connection immediately (see [Section 90.4.5, "How to Acquire a Client Session that Does Not Use Lazy Connection Allocation"](#)).

The server session allocates read connections from its read connection pool to all client sessions. If your application requires multiple read security levels then you must use multiple server sessions or TopLink isolated sessions (see [Section 87.5, "Isolated Client Sessions"](#)).

The server session also supports multiple write connection pools and nonpooled connections. By default, all client sessions use the default write connection pool. However, if your application requires multiple security levels or user logins for write access, then you can use multiple write connection pools. You can configure a client session to use a specific write connection pool or nonpooled connection when it is acquired (see [Section 90.4.4, "How to Acquire a Client Session that Uses a Named Connection Pool"](#)). This connection is only used for writes, not reads (reads still go through the server session read connection pool).

For more information, see the following:

- [Section 96.1.6.1, "Internal Connection Pools"](#)
- [Section 96.1.6.2, "External Connection Pools"](#)

## 87.4 Unit of Work Sessions

The unit of work ensures that the client edits objects in a separate object transaction space. This feature lets clients perform object transactions in parallel. When transactions are committed, the unit of work makes any required changes in the database, and then merges the changes into the shared TopLink session cache. The modified objects are then available to all other users.

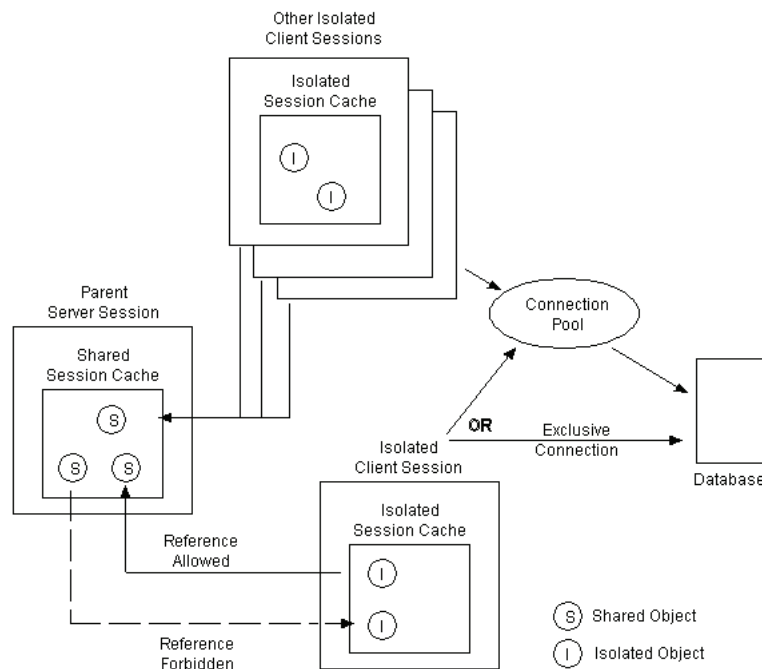
For information on creating, configuring, and using a unit of work, see [Chapter 113, "Introduction to TopLink Transactions"](#).

## 87.5 Isolated Client Sessions

An isolated client session is a special type of client session that provides its own session cache. This session cache is isolated from the shared session cache of its parent server session.

If in your TopLink project you configure all classes as isolated (see [Section 117.11, "Configuring Cache Isolation at the Project Level"](#)), or one or more classes as isolated (see [Section 119.13, "Configuring Cache Isolation at the Descriptor Level"](#)), then all client sessions that you acquire from a parent server session will be isolated client sessions.

[Figure 87–6](#) illustrates the relationship between a parent server session's shared session cache and its child isolated client sessions.

**Figure 87–6 Isolated Client Sessions**

Each isolated client session owns an initially empty cache and identity maps used exclusively for isolated objects that the isolated client session accesses while it is active. The isolated client session's isolated session cache is discarded when the isolated client session is released.

When you use an isolated client session to read an isolated class, the client session reads the isolated object directly from the database and stores it in that client session's isolated session cache. When you use the client session to read a shared class, the client session reads the shared object from the parent server session's shared session cache. If the shared object is not in the parent server session's shared session cache, it will read it from the database and store it in the parent server session's shared session cache.

Isolated objects in an isolated client session's isolated session cache may reference shared objects in the parent server session's shared session cache, but shared objects in the parent server session's shared session cache cannot reference isolated objects in an isolated client session's isolated session cache.

---

**Note:** You cannot define mappings from shared classes to isolated classes. If using CMP, you also cannot define references from isolated enterprise beans to shared EJB.

---

Client sessions can access the data source using a connection pool, or an exclusive connection. To use an exclusive connection, acquire the isolated client session using a `ConnectionPolicy` (see [Section 90.4.2, "How to Acquire a Client Session that Uses Exclusive Connections"](#)). Using an exclusive connection provides improved user-based security for reads and writes. Named queries can also use an exclusive connection (see [Section 119.7.1.10, "Configuring Named Query Advanced Options"](#)).

---

---

**Note:** If an isolated session contains an exclusive connection, you must release the session when you are finished using it. Oracle does not recommend relying on the finalizer to release the connection when the session is garbage-collected. If you are using an active unit of work in a JTA transaction, you do not need to release the client session—the unit of work will release it after the JTA transaction completes.

---

---

Use isolated client sessions to do the following:

- avoid caching highly volatile data in the shared session cache;
- achieve serializable transaction isolation (see [Section 115.15.1.5.1, "Isolated Client Session Cache"](#));
- use the Oracle Virtual Private Database (VPD) feature in your TopLink-enabled application (see [Section 87.5.1, "Isolated Client Sessions and Oracle Virtual Private Database \(VPD\)"](#)).

For more information, see the following:

- [Section 87.5.2, "Isolated Client Session Limitations"](#)
- [Section 90.4.1, "How to Acquire an Isolated Client Session"](#)
- [Chapter 92, "Configuring Exclusive Isolated Client Sessions for Virtual Private Database"](#)

## 87.5.1 Isolated Client Sessions and Oracle Virtual Private Database (VPD)

Oracle9i Database (and later) provides a server-enforced, fine-grained access control mechanism called Virtual Private Database (VPD). VPD ties a security policy to a table by dynamically appending SQL statements with a predicate to limit data access at the row level. You can create your own security policies, or use Oracle's custom implementation of VPD called Oracle Label Security (OLS). For more information on VPD and OLS, see the following:

<http://www.oracle.com/technology/deploy/security/index.html>.

To use Oracle Database VPD feature in your TopLink-enabled application, use isolated client sessions.

Any class that maps to a table that uses VPD must have the descriptor configured as isolated (see [Section 119.13, "Configuring Cache Isolation at the Descriptor Level"](#)).

When you use isolated client sessions with VPD, you typically use exclusive connections (see [Section 90.4.2, "How to Acquire a Client Session that Uses Exclusive Connections"](#)).

To support VPD, you are responsible for implementing session event handlers that the TopLink runtime invokes during the isolated client session life cycle (see [Section 87.5.1.3, "Isolated Client Session Life Cycle"](#)). The session event handler you must implement depends on whether or not you are using Oracle Database proxy authentication (see [Section 87.5.1.1, "VPD with Oracle Database Proxy Authentication"](#) and [Section 87.5.1.2, "VPD Without Oracle Database Proxy Authentication"](#)).

For information, see [Chapter 92, "Configuring Exclusive Isolated Client Sessions for Virtual Private Database"](#).

### 87.5.1.1 VPD with Oracle Database Proxy Authentication

If you are using Oracle Database proxy authentication ([Section 96.1.4.2, "Oracle Database Proxy Authentication"](#)), you must implement a session event handler for the following session events:

- `noRowsModifiedSessionEvent` (see [Section 92.4, "Using NoRowsModifiedSessionEvent Event Handler"](#))

By using Oracle Database proxy authentication, you can set up VPD support entirely in the database. That is, rather than making the isolated client session execute SQL (see [Section 92.2, "Using PostAcquireExclusiveConnection Event Handler"](#) and [Section 92.3, "Using PreReleaseExclusiveConnection Event Handler"](#)), the database performs the required setup in an after login trigger using the proxy `session_user`.

### 87.5.1.2 VPD Without Oracle Database Proxy Authentication

If you are not using Oracle Database proxy authentication, you must implement session event handlers for the following session events:

- `postAcquireExclusiveConnection` (see [Section 92.2, "Using PostAcquireExclusiveConnection Event Handler"](#)): used to perform VPD setup at the time TopLink allocates a dedicated connection to an isolated session and before the isolated session user uses the connection to interact with the database.
- `preReleaseExclusiveConnection` (see [Section 92.3, "Using PreReleaseExclusiveConnection Event Handler"](#)): used to perform VPD cleanup at the time the isolated session is released and after the user is finished interacting with the database.
- `noRowsModifiedSessionEvent` (see [Section 92.4, "Using NoRowsModifiedSessionEvent Event Handler"](#))

In your implementation of these handlers, you obtain the required user credentials from the `ConnectionPolicy` associated with the session (see [Section 90.4.3, "How to Acquire a Client Session that Uses Connection Properties"](#)).

### 87.5.1.3 Isolated Client Session Life Cycle

This section provides an overview of the key phases in the life cycle of an isolated session, including the following:

- Setup required before using an isolated session
- Interaction among isolated session objects
- Clean-up required after using an isolated session

To enable the life cycle of an isolated session, use this procedure:

1. Prepare VPD configuration in the database.
2. Configure your project and session:
  - Designate descriptors as isolated (see [Section 119.13, "Configuring Cache Isolation at the Descriptor Level"](#)).
  - Configure your server session to allocate exclusive connections (see [Section 89.12, "Configuring Connection Policy"](#)).
  - Implement session event listeners for the required connection events
    - If you are using Oracle Database proxy authentication (see [Section 96.1.4.2, "Oracle Database Proxy Authentication"](#)), see [Section 92.4, "Using NoRowsModifiedSessionEvent Event Handler"](#).

- If you are not using Oracle Database proxy authentication, see [Section 92.2, "Using PostAcquireExclusiveConnection Event Handler"](#), [Section 92.3, "Using PreReleaseExclusiveConnection Event Handler"](#), and [Section 92.4, "Using NoRowsModifiedSessionEvent Event Handler"](#)

---

**Note:** You must add these session event listeners to the server session from which you acquire your isolated client session. You cannot add them to the isolated client session itself. For more information, see [Section 89.10, "Configuring Session Event Listeners"](#)

---

- Implement exception handlers for the appropriate exceptions (see [Section 92.5, "Accessing Indirection"](#)).

3. Acquire an isolated session:

- If you are using Oracle Database proxy authentication (see [Section 96.1.4.2, "Oracle Database Proxy Authentication"](#)):

```
Session myIsolatedClientSession = server.acquireClientSession();
```

Because you configured one or more descriptors as isolated, `myIsolatedClientSession` is an isolated session with an exclusive connection.

- If you are not using Oracle Database proxy authentication:

```
ConnectionPolicy myConnPolicy =
    (ConnectionPolicy)server.getDefaultConnectionPolicy().clone();
myConnectionPolicy.setProperty("credentials", myUserCredentials);
Session myIsolatedClientSession =
    server.acquireClientSession(myConnectionPolicy);
```

Set the user's credentials as appropriate properties on `myConnectionPolicy`. Because you configured one or more descriptors as isolated, `myIsolatedClientSession` is an isolated session with an exclusive connection.

The `TopLink` runtime raises a

`SessionEvent.PostAcquireExclusiveConnection` event handled by your `SessionEventListener` (see [Section 92.2, "Using PostAcquireExclusiveConnection Event Handler"](#)).

4. Use `myIsolatedClientSession` to interact with the database.

If the `TopLink` runtime raises a `SessionEvent.NoRowsModified` event, it is handled by your `SessionEventListener` (see [Section 92.4, "Using NoRowsModifiedSessionEvent Event Handler"](#)).

5. When you are finished using `myIsolatedClientSession`, release the isolated session:

```
myIsolatedClientSession.release();
```

The `TopLink` runtime prepares to destroy the isolated cache and to close the exclusive connection associated with this isolated session.

The `TopLink` runtime raises a

`SessionEvent.PreReleaseExclusiveConnection` event handled by your `SessionEventListener` (see [Section 92.3, "Using PreReleaseExclusiveConnection Event Handler"](#)).

6. Repeat steps 3 to 5 (as required) until the application exits.

## 87.5.2 Isolated Client Session Limitations

For the purposes of security as well as efficiency, observe the limitations described in the following section, when you use isolated client sessions in your TopLink three-tier application:

- [Mapping](#)
- [Inheritance](#)
- [Caching and Cache Coordination](#)
- [Sequencing](#)
- [CMP](#)
- [Transactions and JTA](#)

### Mapping

Consider the following mapping and relationship restrictions when using isolated sessions with your relational model:

- Isolated objects may be related to shared objects, but shared objects cannot have any relationships with isolated objects.
- If a table has a VPD security policy associated with it, then the class mapped to that table must be isolated.
- If one of the tables in a multiple table mapping is isolated, then the main class must also be isolated.

The TopLink runtime enforces these restrictions during descriptor initialization.

### Inheritance

Aggregates and aggregate mappings inherit the isolated configuration of their parents.

If a class is isolated, then all inheriting classes should be isolated. Otherwise, if you relate a shared class to a shared superclass with isolated subclasses, it is possible that some of the isolated subclasses will lose object identity when the isolated session is released.

To give you the flexibility to mix shared and isolated classes, the TopLink runtime does not enforce these restrictions during descriptor initialization. If you wish to mix shared and isolated classes in your inheritance hierarchy, then you must be prepared to deal with this possible loss of object identity.

### Caching and Cache Coordination

Isolated classes are never loaded into the shared cache of a parent server session. Isolated classes cannot be used with cache coordination.

### Sequencing

Oracle recommends that you do not configure a sequencing object as isolated. TopLink does not access sequencing objects using the isolated session's dedicated connection, and so the sequence values are not available to the isolated session.

### CMP

For CMP, relationships between isolated and shared data is not allowed. This is because of the relationship maintenance requirements of having bidirectional references for all relationships.

### Transactions and JTA

Oracle recommends that you explicitly release an isolated session when you are finished using it, rather than wait for the Java garbage collector to invoke the finalizer. The finalizer is provided as a last resort: waiting for the garbage collector may cause errors when dealing with a JTA transaction.

## 87.6 Historical Sessions

By default, a session represents a view of the most current version of objects, and when you execute a query in that session, it returns the most current version of selected objects.

If your data source maintains past versions of objects, you can configure TopLink to access this historical data so that you can express read queries conditional on how your objects are changing over time. You can also do the following:

- Make series of queries relative to any point in time—not just the time of the first query.
- Provide read consistency so that a series of read operations or report queries all execute as if at the same time.
- Use the `mergeClone` method to provide deep recovery of an object by passing in a past version of it.

In addition, you can express query selection criteria as either of the following:

- A condition at a past time: for example, "employees who used to..."
- A change over time: for example, "employees who recently..."

For more information, see the following:

- [Section 87.6.1, "Historical Session Limitations"](#)
- [Chapter 93, "Configuring Historical Sessions"](#)
- [Section 90.5, "Acquiring a Historical Session"](#)
- [Section 108.11, "Historical Queries"](#).

### 87.6.1 Historical Session Limitations

The `HistoryPolicy` provides a very flexible means of accommodating a wide variety of historical schemas. However, be aware of the following restrictions:

- You cannot use the `HistoryPolicy`, if your design combines both current and historical data in a single schema.
- You cannot use historical sessions, nor historical queries, with EJB entity beans.
- TopLink assumes that the current version of an object corresponds to the row in the historical table whose row end field is `NULL`.
- You cannot directly map the start and end fields of a history table because they do not exist in the regular schema.
- You cannot query on ranges of historical objects, only as of a specific point in time.

## 87.7 Session Broker and Client Sessions

The **TopLink session broker** is a mechanism that enables client applications to transparently access multiple databases through a single TopLink session.

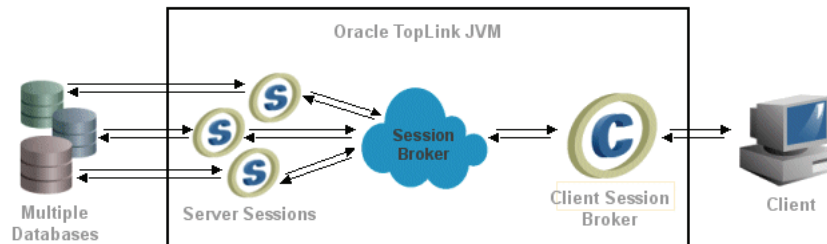
The TopLink session broker enables client applications to access two or more databases through a single session. If your application stores objects in multiple databases, the session broker, which provides seamless communication for client applications, enables the client to view multiple databases as if they were a single database.

When a three-tier session broker application uses server sessions to communicate with the database, clients require a client session to access the database. Similarly, when you implement a session broker, the client requires a *client session broker* to access the database.

A **client session broker** is a collection of client sessions, one from each server session associated with the session broker. When a client acquires a client session broker, the session broker collects one client session from each associated server session, and wraps the client sessions so that they appear to be a single client session to the client application.

As [Figure 87–7](#) illustrates, a session broker connects to the databases through two or more server sessions or database sessions.

**Figure 87–7 TopLink Session Broker with Server Session Architecture**



This section explains the following:

- [Session Broker Architecture](#)
- [Committing a Transaction with a Session Broker](#)
- [Session Broker Session Limitations](#)
- [Session Broker Alternatives](#)

For information, see the following:

- [Section 88.5, "Creating Session Broker and Client Sessions"](#)
- [Chapter 94, "Configuring Session Broker and Client Sessions"](#)
- [Section 90.3, "Acquiring a Session from the Session Manager"](#)
- [Section 90.4, "Acquiring a Client Session"](#)

## 87.7.1 Session Broker Architecture

As [Figure 87–7](#) illustrates, a session broker contains a broker object that acts as an intermediary between the application and the multiple sessions added to the session broker.

To construct a session broker, use TopLink Workbench to modify your `sessions.xml` file as follows:

1. Define two or more sessions (of the same type, either server sessions or database sessions).
2. Define a session broker.



### 3. Add the sessions to the session broker.

When you use `SessionManager` method `getSession(sessionBrokerName)` where `sessionBrokerName` is the name of the session broker you defined, the session manager returns the corresponding session broker session (call it `mySessionBroker`) that contains an instance of each of the sessions you added to it. When you use `mySessionBroker` method `login`, it logs into each defined session. Thereafter, you use `mySessionBroker` as you would any other session: `TopLink` transparently handles access to the multiple databases.

In the case of a three-tier architecture where the session broker contains two or more server sessions, you use session broker method `acquireClientSessionBroker` to acquire a single client session that lets you query across all the data sources managed by the various server sessions. You use this client session as you would any other client session.

## 87.7.2 Committing a Transaction with a Session Broker

By default, when you commit a transaction with a session broker session, a two-stage commit is performed.

Ideally, you should incorporate a JTA external transaction controller in order to benefit from its two-phase commit.

### 87.7.2.1 Committing a Session with a JTA Driver: Two-Phase Commits

If you use a session broker, incorporate a JTA external transaction controller wherever possible. The external transaction controller provides a *two-phase commit*, which passes the SQL statements that are required to commit the transaction to the JTA driver. The JTA driver handles the entire commit process.

JTA guarantees that the transaction commits or rolls back completely, even if the transaction involves more than one database. If the commit operation to any one database fails, then all database transactions roll back. The two-phase commit operation is the safest method available to commit a transaction to the database.

Two-phase commit support requires integration with a compliant JTA driver.

### 87.7.2.2 Committing a Session Without a JTA Driver: Two-Stage Commits

If there is no JTA driver available, then the session broker provides a *two-stage commit* algorithm. A two-stage commit differs from a two-phase commit in that it guarantees data integrity only up to the point of the final commit of the transaction. If the SQL script executes successfully on all databases, but the commit operation then fails on one database, only the database that experiences the commit failure rolls back.

Although unlikely, this scenario is possible. As a result, if your system does not include a JTA driver and you use a two-stage commit, build a mechanism into your application to deal with this type of potential problem.

## 87.7.3 Session Broker Session Limitations

Although the session broker is a powerful tool that lets you use data that is distributed across multiple databases from a single application, it has some limitations including the following:

- It may not meet the needs of your particular distributed data application (see [Section 87.7.4, "Session Broker Alternatives"](#)).
- You cannot split multiple table descriptors across databases.

- Each class must reside on only one database.
- You cannot use joins through expressions across databases.
- Many-to-many join tables must reside on the same database as the target object (See [Section 87.7.3.1, "Many-to-Many Join Tables and Direct Collection Tables"](#) for a work-around for this limitation).

### 87.7.3.1 Many-to-Many Join Tables and Direct Collection Tables

By default, TopLink assumes that many-to-many and direct collection tables are on the same database as the source object. If they are on a different database, then you must configure the mapping's session name using `ManyToManyMapping` or `DirectCollectionMapping` method `setSessionName`, as [Example 87-4](#) illustrates.

Note that a many-to-many join table must still reside on the same database as the target object.

#### **Example 87-4 Using Mapping `setSessionName` in a Descriptor Amendment Method**

```
public void addToDescriptor(ClassDescriptor descriptor) {  
    descriptor.getMappingForAttributeName("projects").setSessionName("branch-database");  
}
```

To work around this problem for data-level queries, use the `DatabaseQuery` method `setSessionName`.

## 87.7.4 Session Broker Alternatives

When evaluating whether or not to use a session broker in your application, consider the following alternatives:

- [Database Linking](#)
- [Multiple Sessions](#)

### 87.7.4.1 Database Linking

Most enterprise databases, such as the Oracle Database, support linking other databases on the database server. This allows querying and two-phase commit across linked databases. Using the session broker is not the same as linking databases. If your database allows linking, Oracle recommends that you use that functionality to provide multiple database access instead of using a session broker.

### 87.7.4.2 Multiple Sessions

An alternative to the session broker is to use multiple sessions to work with multiple databases, as follows:

- If the data on each database is unrelated to data on the other databases, and relationships do not cross database boundaries, then you can create a separate session for each database. For example, you might have individual databases and associated sessions dedicated to each department.

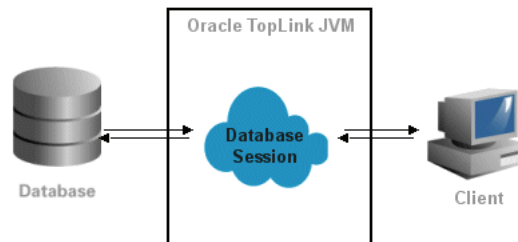
This arrangement requires that you to manage each session manually and ensure that the class descriptors for your project reside in the correct session.

- You can use additional sessions to house a standard batch job. In this case, you can create two or more sessions on the same database. In addition to the main session that supports client queries, you create other sessions that support batch inserts at low-traffic times in your system. This lets you maintain the client cache.

## 87.8 Database Sessions

A **database session** provides a client application with a single data source connection, for simple, standalone applications in which a single connection services all data source requests for one user.

**Figure 87–8 TopLink Database Session Architecture**



A database session is the simplest session TopLink offers. It provides both client and server communications and supports only a single client and a single database connection. It is suitable for simple applications or 2-tier applications.

---

**Note:** Oracle does not recommend using this session type in a 3-tier application because it is not as flexible or scalable as a server and client session. Oracle recommends that you use server sessions and client sessions (see [Section 87.3, "Server and Client Sessions"](#)). Applications that are built using database sessions may be difficult to migrate to a scalable architecture in the future.

---

A database session contains and manages the following information:

- An instance of `Project` and `DatabaseLogin`, which store database login and configuration information
- The JDBC connection and the database access
- The descriptors for each of the application persistent classes
- Identity maps that maintain object identity and act as a cache

For more information, see the following:

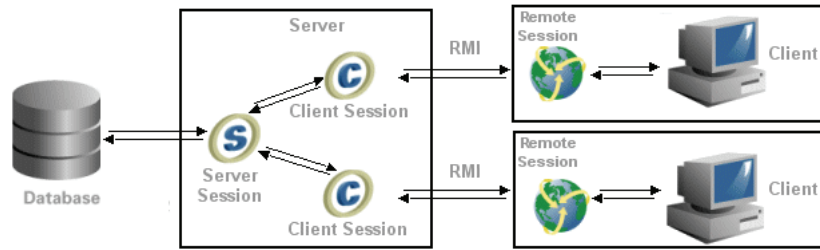
- [Section 88.6, "Creating Database Sessions"](#)
- [Chapter 95, "Configuring Database Sessions"](#)
- [Section 90.3, "Acquiring a Session from the Session Manager"](#)

## 87.9 Remote Sessions

A **remote session** is a client-side session that communicates over RMI with a corresponding client session and server session on the server-side. Remote sessions handle object identity and marshalling and unmarshalling between client-side and server-side.

A remote session resides on the client rather than the TopLink server. The remote session does not replace the client session; rather, a remote session requires a client session to communicate with the server session.

**Figure 87–9 Typical TopLink Server Session with Remote Session Architecture**



The remote session provides a full TopLink session, complete with a session cache, on the client system. TopLink manages the remote session cache and enables client applications to execute operations on the server.

A remote session offers database access to clients that do not reside on the server. The remote session resides on the client and connects by way of RMI to a corresponding client session, which, in turn, connects to its server session on the server.

This section describes the following:

- [Architectural Overview](#)
- [Remote Session Concepts](#)

For more information, see [Section 88.7, "Creating Remote Sessions"](#).

### 87.9.1 Architectural Overview

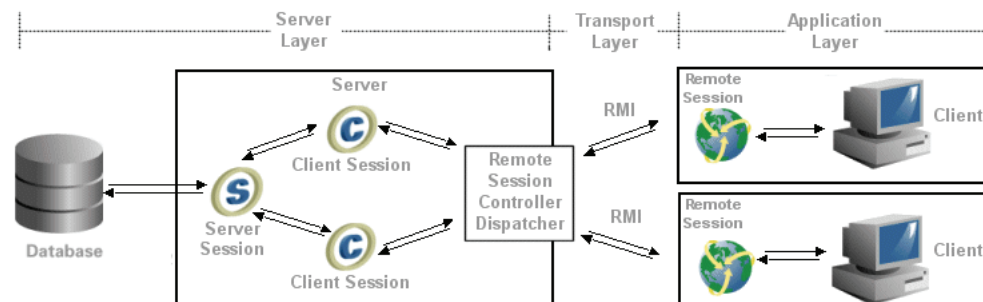
As [Figure 87–10](#) illustrates, the remote session model consists of the following layers:

- The application layer—a client-side application talking to a remote session
- The transport layer—a communication layer, RMI or RMI-IIOP
- The server layer—a TopLink session communicating with a database

The request from the client application to the server travels down through the layers of a distributed system. A client that makes a request to the server session uses the remote session as a conduit to the server session. The client references the remote session, and the remote session forwards a request to the server session through the transport layer.

At run time, the remote session builds its knowledge base by reading descriptors and mappings from the server side as they are needed. These descriptors and mappings are lightweight, because not all information is passed on to the remote session. The information needed to traverse an object tree and to extract primary keys from the given object is passed with the mappings and descriptors.

**Figure 87–10 An Architectural Overview of the Remote Session**



### 87.9.1.1 Application Layer

The application layer includes the application client and the remote session. The remote session is a subclass of `Session` and maintains all the public protocols of the session, giving the appearance of working with the corresponding client session.

The remote session maintains its own identity map and a project of all the descriptors read from the server. If the remote session can handle a request by itself, the request is not passed to the server. For example, a request for an object that is in the remote session cache is processed by the remote session. However, if the object is not in the remote session cache, the request passes to the server session.

### 87.9.1.2 Transport Layer

The transport layer is responsible for carrying the semantics of the invocation. It is a layer that hides all the protocol dependencies from the application and server layers.

The transport layer includes a remote connection that is an abstract entity, through which all requests to the server are forwarded. Each remote session maintains a single remote connection that marshals and unmarshals all requests and responses on the client side.

The remote session supports communications over RMI.

### 87.9.1.3 Server Layer

The server layer includes a remote session controller dispatcher and a `TopLink` sessions: [Figure 87–10](#) illustrates a three-tier server and its client sessions. The remote session controller dispatcher is an interface between the session and transport layers: it marshals and unmarshals all responses and requests between the sessions on the server and their corresponding remote sessions on the client.

## 87.9.2 Remote Session Concepts

When using remote sessions, consider the following:

- [Securing Remote Session Access](#)
- [Queries](#)
- [Refreshing](#)
- [Indirection](#)
- [Cursored Streams](#)
- [Unit of Work](#)

### 87.9.2.1 Securing Remote Session Access

The remote session represents a potential security risk because it requires you to register a remote session controller dispatcher as a service that anyone can access. This can expose the entire database to nonprivileged access.

To reduce this threat, run a server manager as a service to hold the remote session controller dispatcher. All the clients must then communicate through the server manager, which implements the security model for accessing the remote session controller dispatcher.

On the client side, the user requests the remote session controller dispatcher. The manager returns a remote session controller dispatcher only if the user has access rights according to the security model built into the server manager.

To access the system, the remote session controller dispatcher on the client side creates a remote connection, and acquires a remote session from the remote connection. The API for the remote session is the same as for the session, and there is no user-visible difference between working on a session or a remote session.

### 87.9.2.2 Queries

Read queries are publicly available on the client side, but queries that modify objects must be performed using the unit of work.

### 87.9.2.3 Refreshing

Calling refresh methods on the remote session causes database read operations, and may also cause cache updates if the data being refreshed is modified in the database. This can lead to poor performance.

To improve performance, configure refresh methods to run against the server session cache, by configuring the descriptor to always remotely refresh the objects in the cache on all queries. This technique ensures that all queries against the remote session refresh the objects from the server session cache, without the database access.

Cache hits on remote sessions still occur on read object queries based on the primary keys. To avoid this, disable the remote session cache hits on read object queries based on the primary key.

For more information, see [Section 119.9, "Configuring Cache Refreshing"](#).

### 87.9.2.4 Indirection

The remote session supports indirection (lazily loaded) objects. An indirection object is a value holder that can be invoked remotely on the client side. When invoked, the value holder first checks to see if the requested object exists on the remote session. If not, then the associated value holder on the server is instantiated to get the value that is then passed back to the client. Remote value holders are used automatically; the application's code does not change.

### 87.9.2.5 Cursored Streams

A remote session supports both cursored streams and scrollable cursors.

For more information, see [Section 108.5.3, "Stream and Cursor Query Results"](#).

### 87.9.2.6 Unit of Work

Use a unit of work acquired from the remote session to modify objects on the database. A unit of work acquired from the remote session offers the user the same functionality as a unit of work acquired from the client session or the database session.

## 87.10 Sessions and the Cache

Server, database, isolated, and historical sessions include an identity map that maintains object identity, and acts as a cache.

This section explains how the cache differs between the following sessions:

- [Server and Database Session Cache](#)
- [Isolated Session Cache](#)
- [Historical Session Cache](#)

For more information, see [Chapter 102, "Introduction to Cache"](#).

### 87.10.1 Server and Database Session Cache

When a server or database session reads objects from the database, it instantiates them and stores them in its identity map (cache). When the application subsequently queries for the same object, TopLink returns the object in the cache, rather than read the object from the database again.

This cache plays an important role in the performance of your application.

In the case of a server session, all client sessions acquired from it share the server session's cache.

To define how the cache manages objects, specify a strategy for cache management in TopLink Workbench.

### 87.10.2 Isolated Session Cache

When an isolated session reads an object, whose descriptor is configured as isolated, that object is instantiated and stored in the isolated session's cache only—it is not stored in the parent server session's shared object cache. Objects in the isolated session's cache may reference objects in the parent server session's shared object cache, but objects in the parent server session's shared object cache can never reference objects in the isolated session's cache.

### 87.10.3 Historical Session Cache

When a historical session reads objects, it does so only from its static, read-only cache, which is populated with all objects as of a specified time.

## 87.11 Session API

The session API is defined by the following interfaces:

- `oracle.toplink.sessions.Session`
- `oracle.toplink.sessions.DatabaseSession`
- `oracle.toplink.sessions.UnitOfWork`
- `oracle.toplink.threetier.Server`

These APIs are used at run time to access objects and the data source. Always use the session public interfaces, not the corresponding implementation classes.

You should use the `Session` interface when reading and querying with any of client sessions, session brokers, isolated client sessions, historical sessions, remote sessions, and database sessions.

You should use the `UnitOfWork` interface for all units of work acquired from any type of session.

You should use the `Server` interface to configure and acquire a client session from a `Server` session.

The `DatabaseSession` interface can be used for a database session.

Typically, you define server sessions, database sessions, and session broker sessions in a `sessions.xml` file and acquire them at run time using the `SessionManager`. You can also acquire a server session or database session from a `Project`. The only session that should ever be instantiated directly is the `SessionBroker`, and only when not using the `SessionManager`.

You acquire a client session from a server session.

You can also acquire a client session broker from a session broker composed of server sessions.

You acquire a unit of work from any session instance, client session broker, or session broker which contains `DatabaseSession` instances.

[Example 87-5](#) illustrates the session interfaces that derive from `oracle.toplink.sessions.Session` interface.

***Example 87-5 Session Interface Inheritance Hierarchy***

```
oracle.toplink.sessions.Session
    oracle.toplink.sessions.DatabaseSession
        oracle.toplink.threetier.Server
    oracle.toplink.sessions.UnitOfWork
```



---

---

## Creating a Session

This chapter explains how to create TopLink sessions.

This chapter includes the following sections:

- [Introduction to the Session Creation](#)
- [Creating a Sessions Configuration](#)
- [Configuring a Sessions Configuration](#)
- [Creating a Server Session](#)
- [Creating Session Broker and Client Sessions](#)
- [Creating Database Sessions](#)
- [Creating Remote Sessions](#)

For information on the various types of session available, see [Section 87.1, "Session Types"](#).

### 88.1 Introduction to the Session Creation

Each TopLink session is contained within a sessions configuration (`sessions.xml`) file. You can create a sessions configuration using Oracle JDeveloper TopLink Editor, TopLink Workbench, or Java code. Oracle recommends that you use Oracle JDeveloper to create and manage your sessions (see [Section 88.2, "Creating a Sessions Configuration"](#)).

Alternatively, you can create sessions in Java. For more information on creating sessions in Java, see Oracle Fusion Middleware Java API Reference for Oracle TopLink.

After you create a session, you must configure its various options (see [Chapter 89, "Configuring a Session"](#)). After configuring the session, you can use it in your application to manage persistence (see [Chapter 90, "Acquiring and Using Sessions at Run Time"](#)).

### 88.2 Creating a Sessions Configuration

Oracle JDeveloper TopLink Editor and TopLink Workbench let you create session instances and save them in the `sessions.xml` file. These tools represent the `sessions.xml` file as a sessions configuration. Individual session instances are contained within the sessions configuration. You can create multiple sessions configurations, each corresponding to its own uniquely named `sessions.xml` file.

Oracle recommends that you use Oracle JDeveloper to create and manage sessions. It is the most efficient and flexible approach to session management. For more

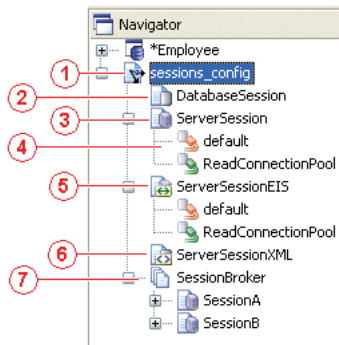
information about the advantages of this approach, see [Section 87.2.2, "Session Configuration and the sessions.xml File"](#).

TopLink Workbench displays sessions configurations and their contents in the Navigator window. When you select a session configuration, its attributes are displayed in the Editor window.

[Figure 88–1](#) calls out the following user interface elements:

1. Sessions Configuration
2. Database Session
3. Relational Server Session
4. Connection Pool
5. EIS Server Session
6. XML Session
7. Session Broker

**Figure 88–1 Sessions Configurations in Navigator Window**



### 88.2.1 How to Create a Sessions Configuration Using TopLink Workbench

To create a TopLink sessions configuration (`sessions.xml` file), use this procedure:



1. Click **New** on the toolbar and select **Sessions Configuration**.



You can also create a new sessions configuration by selecting **File > New > Session Configuration** from the menu, or by clicking **Create New Sessions Configuration** in the standard toolbar.

2. The new sessions configuration element appears in the Navigator window; the Sessions Configuration property sheet appears in the Editor window.

Enter data in each field on the Sessions Configuration property sheet as [Section 88.3, "Configuring a Sessions Configuration"](#) describes.

## 88.3 Configuring a Sessions Configuration

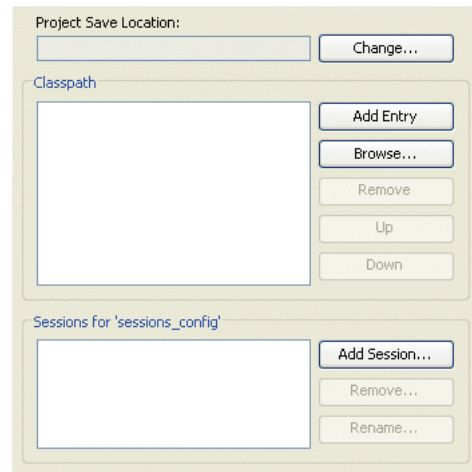
Each TopLink sessions configuration (`sessions.xml` file) can contain multiple sessions and session brokers. In addition, you can specify a classpath for each sessions configuration that applies to all the sessions it contains.

### 88.3.1 How to Configure a Sessions Configuration Using TopLink Workbench

To configure a session configuration, use this procedure:

1. Select the session configuration in the **Navigator**. Its properties appear in the Editor.

**Figure 88–2 Sessions Configuration Property Sheet**



Use the following information to enter data in each field of the Sessions configuration property sheet:

Field	Description
<b>Project Save Location</b>	Click <b>Change</b> and select the directory in which to save the sessions configuration.
<b>Classpath</b>	<p>Lists the JAR or ZIP files that contain the compiled Java classes on which this sessions configuration depends for features that require an external Java class (for example, session event listeners).</p> <ul style="list-style-type: none"> <li>■ To add a JAR or ZIP file, click <b>Add Entries</b> or <b>Browse</b> add the file.</li> <li>■ To remove a JAR or ZIP file, select the file and click <b>Remove</b>.</li> <li>■ To change the order in which TopLink searches these JAR or ZIP files, select a file and click <b>Up</b> to move it forward or click <b>Down</b> to move it back in the list.</li> </ul>
<b>Sessions for &lt;sessions configuration name&gt;</b>	<p>Lists the available sessions defined in this sessions configuration:</p> <ul style="list-style-type: none"> <li>■ To add a session, click <b>Add Session</b>.</li> <li>■ To remove a session, select the session and click <b>Remove</b>.</li> <li>■ To rename a session, select the session and click <b>Rename</b>.</li> </ul> <p>For more information on creating sessions using TopLink Workbench, seen the following:</p> <ul style="list-style-type: none"> <li>■ <a href="#">Section 88.4, "Creating a Server Session"</a></li> <li>■ <a href="#">Section 88.5, "Creating Session Broker and Client Sessions"</a></li> <li>■ <a href="#">Section 88.6, "Creating Database Sessions"</a></li> </ul>

## 88.4 Creating a Server Session

Oracle recommends that you create server sessions using Oracle JDeveloper or TopLink Workbench (see [Section 88.4.1, "How to Create a Server Session Using TopLink Workbench"](#)).

After you create a server session, you create a client session by acquiring it from the server session (see [Section 90.4, "Acquiring a Client Session"](#)).

### 88.4.1 How to Create a Server Session Using TopLink Workbench

Before you create a server session, you must first create a sessions configuration (see [Section 88.2, "Creating a Sessions Configuration"](#)).

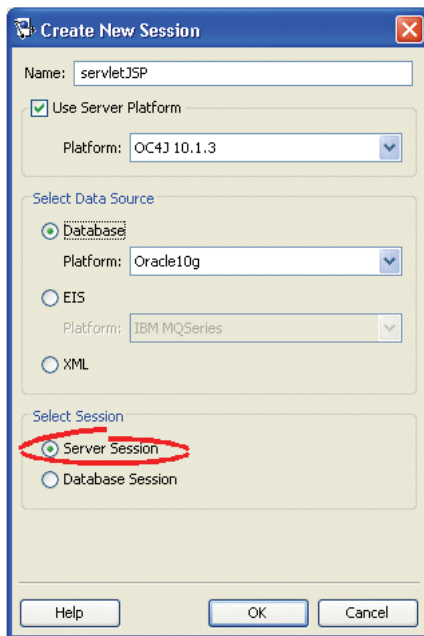
To create a new TopLink server session, use this procedure:

1. Select the sessions configuration in the **Navigator** window in which you want to create a session.
2. Click **Add Session** on the toolbar. The Create New Session dialog box appears.



You can also create a new server session by right-clicking the sessions configuration in the **Navigator** and selecting **New > Session** from the context menu, or by clicking **Add Session** on the Sessions Configuration property sheet.

**Figure 88–3 Create New Session Dialog Box, Server Session Option**



Use the following information to enter data in each field of the dialog box:

Field	Description
Name	Specify the name of the new session.
Use Server Platform	Check this field if you intend to deploy your application to a Java EE application server. If you check this field, you must configure the target application server by selecting a <b>Platform</b> .

Field	Description
<b>Platform</b>	<p>This option is only available if you check <b>Use Server Platform</b>.</p> <p>Select the Java EE application server to which you will deploy your application.</p> <p>TopLink supports the following Java EE application servers:</p> <ul style="list-style-type: none"> <li>■ WebLogic 10.n</li> <li>■ WebLogic 9.n</li> <li>■ OC4J 10.1.3.n</li> <li>■ SunAS 9.0</li> <li>■ WebSphere 6.1</li> <li>■ JBoss</li> <li>■ Custom</li> </ul> <p>The server platform you select is the default server platform for all sessions you create in this sessions configuration. At the session level, you can override this selection or specify a custom server platform class (see <a href="#">Section 89.9, "Configuring the Server Platform"</a>).</p>
<b>Select Data Source</b>	<p>Select the data source for this session configuration. Each session configuration must contain one data source. Choose one of the following:</p> <ul style="list-style-type: none"> <li>■ <b>Database</b> to create a session for a relational project.</li> <li>■ <b>EIS</b> to create a session for an EIS project.</li> <li>■ <b>XML</b> to create a session for an XML project<sup>1</sup>.</li> </ul> <p>See <a href="#">Section 15.1, "TopLink Project Types"</a> for more information.</p>
<b>Select Session</b>	<p>Select <b>Server Session</b> to create a session for a single data source (including shared object cache and connection pools) for multiple clients in a three-tier application.</p>

<sup>1</sup> You cannot create a server session for an XML project.

## 88.4.2 How to Create a Server Session Using Java

You can create a server session in Java code using a project. You can create a project in Java code, or read a project from a `project.xml` file.

[Example 88–1](#) illustrates creating an instance (called `serverSession`) of a `Server` class using a `Project` class.

### **Example 88–1** Creating a Server Session from a Project Class

```
Project myProject = new Project();
Server serverSession = myProject.createServerSession();
```

[Example 88–2](#) illustrates creating an instance (called `serverSession`) of a `Server` class using a `Project` read in from a `project.xml` file.

### **Example 88–2** Creating a Server Session from a project.xml File Project

```
Project myProject = XMLProjectReader.read("myproject.xml");
Server serverSession = myProject.createServerSession();
```

[Example 88–3](#) illustrates creating a server session with a specified write connection pool minimum and maximum size (when using TopLink internal connection pooling). The default write connection pool minimum size is 5 and maximum size is 10.

**Example 88–3 Creating a Server Session with Custom Write Connection Pool Size**

```
XMLProjectReader.read("myproject.xml");
Server serverSession = myProject.createServerSession(32, 32);
```

## 88.5 Creating Session Broker and Client Sessions

A session broker may contain both server sessions and database sessions. Oracle recommends that you use the session broker with server sessions because server sessions are the most scalable session type.

Oracle recommends that you create server sessions using Oracle JDeveloper or TopLink Workbench (see [Section 88.5.1, "How to Create a Session Broker and Client Sessions Using TopLink Workbench"](#)).

After you create and configure a session broker with server sessions, you can acquire a client session from the session broker at run time to provide a dedicated connection to all the data sources managed by the session broker for each client. For more information, see [Section 90.4, "Acquiring a Client Session"](#).

### 88.5.1 How to Create a Session Broker and Client Sessions Using TopLink Workbench

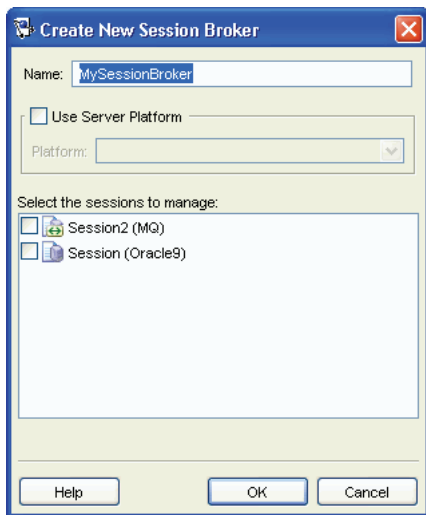
Before you create a session broker session, you must first create a sessions configuration (see [Section 88.2, "Creating a Sessions Configuration"](#)) and one or more server sessions ([Section 88.4, "Creating a Server Session"](#)), or one or more database sessions ([Section 88.6, "Creating Database Sessions"](#)).

To create a new TopLink session broker, use this procedure:

1. Select the sessions configuration in the **Navigator** window in which you want to create a session broker.
2. Click **Add Session Broker** on the toolbar. The Create New Session Broker dialog box appears.

You can also create a new session broker by right-clicking the sessions configuration in the **Navigator** window and selecting **Add Session Broker** from the context menu or by clicking **Add Session Broker** on the Sessions Configuration property sheet.

**Figure 88–4 Create New Session Broker Dialog Box**



Use the following information to enter data in each field of the dialog box:

Field	Description
<b>Name</b>	Specify the name of the new session broker.
<b>Use Server Platform</b>	Check this field if you intend to deploy your application to a Java EE application server. If you check this field, you must configure the target application server by selecting a <b>Platform</b> .
<b>Platform</b>	This option is available only if you check <b>Use Server Platform</b> . Select the Java EE application server to which you will deploy your application. TopLink supports the following Java EE application servers: <ul style="list-style-type: none"> <li>■ WebLogic 10.n</li> <li>■ WebLogic 9.n</li> <li>■ OC4J 10.1.3.n</li> <li>■ SunAS 9.0</li> <li>■ WebSphere 6.1</li> <li>■ JBoss</li> <li>■ Custom</li> </ul> The server platform you select is the default server platform for all sessions you create in this sessions configuration. At the session level, you can override this selection or specify a custom server platform class (see <a href="#">Section 89.9, "Configuring the Server Platform"</a> ).
<b>Select the sessions to Manage</b>	Select sessions to be managed by this new session broker (from the list of available sessions) and click <b>OK</b> . <b>Note:</b> This field appears only if the configuration contains valid sessions.

Continue with [Chapter 89, "Configuring a Session"](#).

## 88.5.2 How to Create a Session Broker and Client Sessions Using Java

[Example 88–4](#) illustrates how you can create a session broker in Java code by instantiating a `SessionBroker` and registering the brokered sessions with it.

Because the session broker references other sessions, configure these sessions before instantiating the session broker. Add all required descriptors to the session, but do not initialize the descriptors or log the sessions. The session broker manages these issues when you instantiate it.

### **Example 88–4** Creating a Session Broker

```
Project databaseProject = new MyDatabaseProject();
Server databaseSession = databaseProject.createServerSession();
```

```
Project eisProject = new MyEISProject();
Server eisSession = eisProject.createServerSession();
```

```
SessionBroker sessionBroker = new SessionBroker();
sessionBroker.registerSession("myDatabase", databaseSession);
sessionBroker.registerSession("myEIS", eisSession);
```

```
sessionBroker.login();
```

## 88.6 Creating Database Sessions

Oracle recommends that you create database sessions using Oracle JDeveloper or TopLink Workbench (see [Section 88.6.1, "How to Create Database Sessions Using TopLink Workbench"](#)).

After you create a database session, you can acquire and use it at run time. For more information on acquiring a database session, see [Section 90.3, "Acquiring a Session from the Session Manager"](#).

### 88.6.1 How to Create Database Sessions Using TopLink Workbench

Before you create a database session, you must first create a sessions configuration (see [Section 88.2, "Creating a Sessions Configuration"](#)).

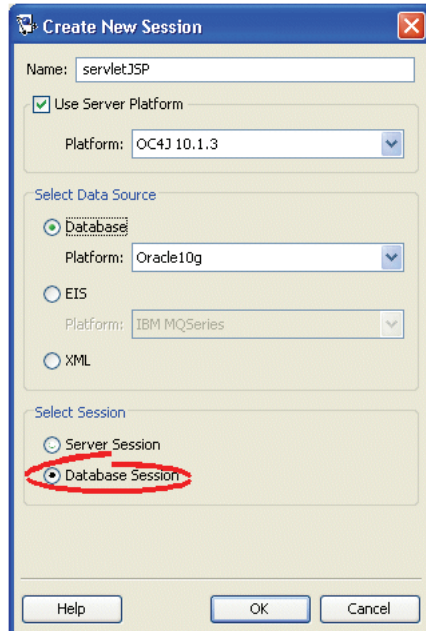
To create a new TopLink database session, use this procedure:

1. Select the session configuration in the **Navigator** window in which you want to create a session.
2. Click **Add Session** on the toolbar. The Create New Session dialog box appears.



You can also create a new configuration by right-clicking the sessions configuration in the **Navigator** window and selecting **New > Session** from the context menu.

**Figure 88–5** Create New Session Dialog Box, Database Session Option



Use the following information to enter data in each field of the dialog box:

Field	Description
Name	Specify the name of the new session.



Field	Description
<b>Use Server Platform</b>	<p>Check this field if you intend to deploy your application to a Java EE application server.</p> <p>If you check this field, you must configure the target application server by selecting a <b>Platform</b>.</p>
<b>Platform</b>	<p>This option is available only if you check <b>Use Server Platform</b>.</p> <p>Select the Java EE application server to which you will deploy your application.</p> <p>TopLink supports the following Java EE application servers:</p> <ul style="list-style-type: none"> <li>■ WebLogic 10.n</li> <li>■ WebLogic 9.n</li> <li>■ OC4J 10.1.3.n</li> <li>■ SunAS 9.0</li> <li>■ WebSphere 6.1</li> <li>■ JBoss</li> <li>■ Custom</li> </ul> <p>The server platform you select is the default server platform for all sessions you create in this sessions configuration. At the session level, you can override this selection or specify a custom server platform class (see <a href="#">Section 89.9, "Configuring the Server Platform"</a>).</p>
<b>Select Data Source</b>	<p>Select the data source for this session configuration. Each session configuration must contain one data source. Choose one of the following:</p> <ul style="list-style-type: none"> <li>■ <b>Database</b> to create a session for a relational project.</li> <li>■ <b>EIS</b> to create a session for an EIS project.</li> <li>■ <b>XML</b> to create a session for an XML project.</li> </ul> <p>See <a href="#">Section 15.1, "TopLink Project Types"</a> for more information.</p>
<b>Select Session</b>	<p>Select <b>Database Session</b> to create a session for a single database (including shared object cache and connection pools) for a single client suitable for simple applications or prototyping.</p>

Enter the necessary information and click **OK**.

TopLink Workbench window appears, showing the database session in the Navigator window.

Continue with [Chapter 89, "Configuring a Session"](#).

## 88.6.2 How to Create Database Sessions Using Java

You can create an instance of the `DatabaseSession` class in Java code using a `Project`. You can create a project in Java code or read a project from a `project.xml` file.

[Example 88–5](#) illustrates creating a `DatabaseSession` using a `Project` class.

### **Example 88–5** *Creating a Database Session from a Project Class*

```
Project myProject = new Project();
DatabaseSession databaseSession = myProject.createDatabaseSession();
```

[Example 88–6](#) illustrates creating a `DatabaseSession` using a `Project` read in from a `project.xml` file.

**Example 88–6 Creating a Database Session from a project.xml File Project**

```
Project myProject = XMLProjectReader.read("myproject.xml");
DatabaseSession databaseSession = myProject.createDatabaseSession();
```

## 88.7 Creating Remote Sessions

Remote sessions are acquired through a remote connection to their server-side session. Remote sessions are acquired through Java code on the remote client. The server-side session must also be registered with an `oracle.toplink.remote.ejb.RemoteSessionController` and accessible from the RMI naming service.

You create remote sessions entirely in Java code (see [Section 88.7.1, "How to Create Remote Sessions Using Java"](#)).

### 88.7.1 How to Create Remote Sessions Using Java

[Example 88–7](#) and [Example 88–8](#) demonstrate how to create a remote TopLink session on a client that communicates with a remote session controller on a server that uses RMI. After creating the connection, the client application uses the remote session as it does with any other TopLink session.

#### 88.7.1.1 Server

[Example 88–7](#) shows the code you add to your application's RMI service (`MyRMIServerManagerImpl`) to create and return an instance of an `RMIRemoteSessionController` to the client. The controller sits between the remote client and the local TopLink session.

The `RMIRemoteSessionController` you create on the server is based on a TopLink server session. You create and configure this server session as described in [Section 88.4, "Creating a Server Session"](#) and [Section 91, "Configuring Server Sessions"](#).

**Example 88–7 Server Creating RMIRemoteSessionController for Client**

```
RMIRemoteSessionController controller = null;
try {
    // Create instance of RMIRemoteSessionControllerDispatcher which implements
    // RMIRemoteSessionController. The constructor takes a TopLink session as a parameter
    controller = new RMIRemoteSessionControllerDispatcher (localTopLinkSession);
}
catch (RemoteException exception) {
    System.out.println("Error in invocation " + exception.toString());
}
return controller;
```

#### 88.7.1.2 Client

The client-side code gets a reference to the application's RMI service (in this example it is called `MyRMIServerManager`) and uses this code to get the `RMIRemoteSessionController` running on the server. The reference to the session controller is then used to create the `RMIConnection` from which it acquires a remote session.

**Example 88–8 Client Acquiring RMIRemoteSessionController from Server**

```
MyRMIServerManager serverManager = null;
// Set the client security manager
try {
    System.setSecurityManager(new MyRMISecurityManager());
```

```
}
catch(Exception exception) {
    System.out.println("Security violation " + exception.toString());
}
// Get the remote factory object from the Registry
try {
    serverManager = (MyRMIServerManager) Naming.lookup("SERVER-MANAGER");
}
catch (Exception exception) {
    System.out.println("Lookup failed " + exception.toString());
}
// Start RMIRemoteSession on the server and create an RMIConnection
RMIConnection rmiConnection = null;
try {
    rmiConnection = new RMIConnection(
        serverManager.createRemoteSessionController()
    );
}
catch (RemoteException exception) {
    System.out.println("Error in invocation " + exception.toString());
}
// Create a remote session which we can use as a normal TopLink session
Session session = rmiConnection.createRemoteSession();
```



---



---

## Configuring a Session

This chapter describes how to configure TopLink sessions.

This chapter includes the following sections:

- [Configuring Common Session Options](#)
- [Configuring a Primary Mapping Project](#)
- [Configuring a Session Login](#)
- [Configuring Logging](#)
- [Configuring Multiple Mapping Projects](#)
- [Configuring a Performance Profiler](#)
- [Configuring an Exception Handler](#)
- [Configuring a Session Customizer Class](#)
- [Configuring the Server Platform](#)
- [Configuring Session Event Listeners](#)
- [Configuring the Integrity Checker](#)
- [Configuring Connection Policy](#)
- [Configuring Named Queries at the Session Level](#)

Table 89–1 lists the types of TopLink sessions that you can configure and provides a cross-reference to the type-specific chapter that lists the configurable options supported by that type.

**Table 89–1** *Configuring TopLink Sessions*

<b>If you are creating...</b>	<b>See...</b>
Server and client sessions (see Section 87.3, "Server and Client Sessions")	<a href="#">Chapter 91, "Configuring Server Sessions"</a>
Unit of work sessions (see Section 87.4, "Unit of Work Sessions")	<a href="#">Chapter 113, "Introduction to TopLink Transactions"</a>
Isolated client sessions (see Section 87.5, "Isolated Client Sessions")	<a href="#">Chapter 92, "Configuring Exclusive Isolated Client Sessions for Virtual Private Database"</a>
Historical sessions (see Section 87.6, "Historical Sessions")	<a href="#">Chapter 93, "Configuring Historical Sessions"</a>

**Table 89–1 (Cont.) Configuring TopLink Sessions**

If you are creating...	See...
Session broker and client sessions (see Section 87.7, "Session Broker and Client Sessions")	Chapter 94, "Configuring Session Broker and Client Sessions"
Database sessions (see Section 87.8, "Database Sessions")	Chapter 95, "Configuring Database Sessions"

Table 89–2 lists the configurable options shared by two or more TopLink sessions types.

For more information, see the following:

- Chapter 87, "Introduction to TopLink Sessions"
- Chapter 88, "Creating a Session"

## 89.1 Configuring Common Session Options

Table 89–2 lists the configurable options shared by two or more TopLink session types. In addition to the configurable options described here, you must also configure the options described for the specific *Session Types*, as shown in Table 89–1

**Table 89–2 Configurable Options for Session**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Primary mapping project (see Section 89.2, "Configuring a Primary Mapping Project")	✓	✓	✓
Session login (see Section 89.3, "Configuring a Session Login")	✓	✓	✓
Logging (see Section 89.4, "Configuring Logging")	✓	✓	✓
Multiple mapping projects (see Section 89.5, "Configuring Multiple Mapping Projects")	✓	✓	✓
Performance profiler (see Section 89.6, "Configuring a Performance Profiler")	✓	✓	✓
Exception handler (see Section 89.7, "Configuring an Exception Handler")	✓	✓	✓
Session customizer class (see Section 89.8, "Configuring a Session Customizer Class")	✓	✓	✓
Server platform (see Section 89.9, "Configuring the Server Platform")	✓	✓	✓
Session event listener (see Section 89.10, "Configuring Session Event Listeners")	✓	✓	✓
Coordinated cache (see Section 103, "Configuring a Coordinated Cache")	✓	✓	✓
Integrity checker (see Section 89.11, "Configuring the Integrity Checker")			✓
Connection policy (see Section 89.12, "Configuring Connection Policy")	✓	✓	✓
Named queries (see Section 89.13, "Configuring Named Queries at the Session Level")			✓

## 89.2 Configuring a Primary Mapping Project

The mapping project contains your TopLink mapping metadata (see [Chapter 15, "Introduction to Projects"](#)), including descriptors and mappings. Each session is associated with at least one project so that the session can register the descriptors.

[Table 89–3](#) summarizes which sessions support a primary mapping project configuration.

**Table 89–3 Session Support for Primary Mapping Project**

Session	How to Use Oracle JDeveloper	How to Configure a Primary Mapping Project Using TopLink Workbench	How to Configure a Primary Mapping Project Using Java
Server and client sessions (see <a href="#">Section 87.3, "Server and Client Sessions"</a> )	✓	✓	✓
Session broker and client sessions (see <a href="#">Section 87.7, "Session Broker and Client Sessions"</a> )	✓	✓	✓
Database sessions (see <a href="#">Section 87.8, "Database Sessions"</a> )	✓	✓	✓

Using Oracle JDeveloper TopLink Editor or TopLink Workbench, you can export your mapping metadata as either a deployment XML file or as a Java class. Consequently, in a session, you can specify the mapping project as an XML file or as a Java class.

Oracle recommends that you export your mapping metadata from Oracle JDeveloper as a deployment XML file (see [Section 116.3, "Exporting Project Information"](#)).

If you export your mapping metadata as a Java class, you must compile it and add it to the session configuration classpath (see [Section 88.3, "Configuring a Sessions Configuration"](#)) before adding it to a session.

---

**Note:** When specifying the mapping project using XML, you can specify the Java resource path. In most applications, the `sessions.xml` and `project.xml` files are deployed inside the JAR file, and the project XML path is specified as a Java resource path.

When specifying the Java resource path, ensure that you are using the forward slash character ( / ) for directories, not the back slash ( \ ). For example, `com/myapp/my-persistence/my-project.xml`, or `META-INF/my-project.xml`.

---

See [Section 89.5, "Configuring Multiple Mapping Projects"](#) for information on configuring additional TopLink projects for the session.

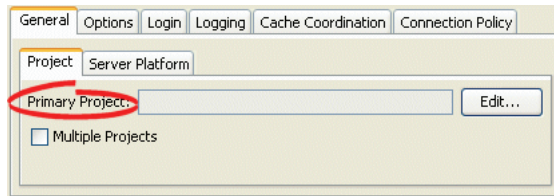
### 89.2.1 How to Configure a Primary Mapping Project Using TopLink Workbench

To specify the primary TopLink project metadata for your session, use this procedure:

1. Select a server or database session in the **Navigator**. Its properties appear in the Editor.

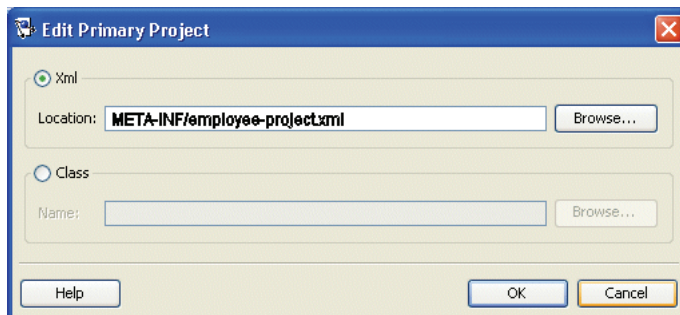
2. Click the **General** tab. The General tab appears.
3. Click the **Project** subtab. The Project subtab appears.

**Figure 89–1 General Tab, Project Subtab, Primary Project Option**



4. Select the following options:
  - Click **Edit** to define the primary project. The Edit PrimaryProject dialog box appears.
  - Select the **Multiple Projects** option to add additional projects to the session. See [Section 89.5, "Configuring Multiple Mapping Projects"](#) for more information.

**Figure 89–2 Edit Primary Project Dialog Box**



Use this information to enter data in each field of the Edit Primary Project dialog box:

Field	Description
XML	Select <b>XML</b> to add a mapping project as a deployment XML file. Click <b>Browse</b> to select the file.
Class	Select <b>Class</b> to add a mapping project as a <i>compiled</i> Java class file. Click <b>Browse</b> to select the file.

## 89.2.2 How to Configure a Primary Mapping Project Using Java

Using Java, you can register descriptors with a session using the following API:

- **Project API**—Read your `project.xml` file (or instantiate your project class) and create your session using `Project` method `createServerSession` or `createDatabaseSession`.
- **Session API**—Add a descriptor or set of descriptors to a session using the `DatabaseSession` API that [Table 89–4](#) lists. Descriptors should be registered before login, but independent sets of descriptors can be added after login.



**Table 89–4 DatabaseSession API for Registering Descriptors**

Session Method	Description
<code>addDescriptors(Project)</code>	Add to the session all the descriptors owned by the passed in <code>Project</code> .
<code>addDescriptors(Vector)</code>	Add to the session all the descriptors in the passed in <code>Vector</code> .
<code>addDescriptor(Descriptor)</code>	Add an individual descriptor to the session.

## 89.3 Configuring a Session Login

A session login encapsulates details of data source access for any session that persists to a data source. The session login overrides any other login configuration.

Table 89–5 summarizes which sessions support session login configuration.

**Table 89–5 Session Support for Session Login**

Session	Session Login
Server and Client Sessions	✓
Session Broker and Client Sessions	✓
Database Sessions	✓

The session login provides access to a variety of features, including the following:

- Connection configuration such as whether or not to use external connection pooling.
- Sequencing configuration (that overrides sequencing configuration made at the project level, if any).
- Miscellaneous options specific to your chosen data source.
- Properties (arbitrary, application-specific named values).

For more information, see the following:

- [Section 96.1.2, "Data Source Login Types"](#)
- [Chapter 97, "Configuring a Data Source Login"](#)

## 89.4 Configuring Logging

Use the TopLink logging framework to record TopLink behavior to a log file or session console.

Table 89–6 summarizes which sessions support logging configuration.

**Table 89–6 Session Support for Logging**

Session	How to Use Oracle JDeveloper	How to Configure Logging Using TopLink Workbench	How to Configure Logging Using Session API in Java	How to Configure Logging in a Java EE Container
Server and client sessions (see Section 87.3, "Server and Client Sessions")	✓	✓	✓	✓
Unit of work sessions (see Section 87.4, "Unit of Work Sessions")	✓	✓	✓	✓
Session broker and client sessions (see Section 87.7, "Session Broker and Client Sessions")	✓	✓	✓	
Database sessions (see Section 87.8, "Database Sessions")	✓	✓	✓	

**Note:** If the session belongs to a session broker, you must specify the logging information in the session broker—not in the session itself.

By default, TopLink uses its own native logger. Alternatively, you can configure TopLink to use the `java.util.logging` package (see Section 89.4.5, "How to Configure a Session to use the `java.util.logging` Package").

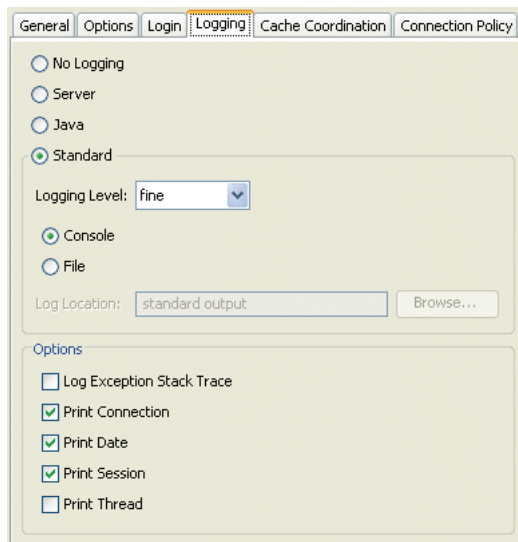
For more information, see Section 87.2.6, "Logging".

### 89.4.1 How to Configure Logging Using TopLink Workbench

To specify the logging information for a session, use this procedure:

1. Select a database session in the **Navigator**. Its properties appear in the Editor.
2. Click the **Logging** tab. The Logging tab appears.

**Figure 89–3 Logging Tab**



Use the following information to enter data in each field of the Logging tab to select the profiler option to use with this session:

Option	Description
<b>No Logging</b>	Select this option to specify that nothing is logged for this session.
<b>Server</b>	Select this option to use logging capabilities of the application server to which you are deploying this application.
<b>Java</b>	Select this option to use <code>java.util.logging</code> package.
<b>Standard</b>	Select this option to use the TopLink logging framework. When selected, you can optionally configure the following options.
<b>Logging Level</b>	Define the amount of logging information to record (in ascending order of information): <ul style="list-style-type: none"> <li>■ <b>Config</b>—Log only login, JDBC connection, and database information.</li> <li>■ <b>Info</b> (default)—Log the login/logout per sever session, with user name. After acquiring the session, detailed information is logged.</li> <li>■ <b>Warning</b>—Log exceptions that do not force TopLink to stop, including all exceptions not logged with <b>Severe</b> level. This does not include a stack trace.</li> <li>■ <b>Severe</b> —Log exceptions indicating TopLink cannot continue, and any exceptions generated during login. This includes a stack trace.</li> <li>■ <b>Fine</b>—Log SQL (including thread information).</li> <li>■ <b>Finer</b>—Similar to warning. Includes stack trace.</li> <li>■ <b>Finest</b>—Includes additional low-level information.</li> <li>■ <b>All</b>—Log everything.</li> </ul>
<b>Console</b>	Select this option to display logging information to the standard console output.
<b>File</b>	Select this option to record logging information in a file. Click <b>Browse</b> to specify the name and location of the log file.
<b>Options</b>	Select this option to override additional logging option defaults for <b>Java</b> and <b>Standard</b> logging only.
<b>Log Exception Stack Trace</b>	Select this option to include the stack trace with any exception written to the log.  Default: For <b>SEVERE</b> messages, log stack trace. For <b>WARNING</b> messages, only log stack trace at log level <b>FINER</b> or lower.
<b>Print Connection</b>	Select this option to include the connection identifier in any connection related log messages.  Default: Enabled for all message and log levels.
<b>Print Date</b>	Select this option to include the date and time at which the log message was generated.  Default: Enabled for all message and log levels.

Option	Description
<b>Print Session</b>	Select this option to include the session name in any session related log messages. Default: Enabled for all message and log levels.
<b>Print Thread</b>	Select this option to include the thread name in any thread related log messages. Default: Log only at log level FINER or lower.

## 89.4.2 How to Configure Logging Using Session API in Java

If you use TopLink native logging (the default), then at run time, you can configure logging options using `oracle.toplink.sessions.Session` logging API.

The `Session` interface defines the following logging methods:

- `setSessionLog`—specify the type of logging to use (any implementor of `oracle.toplink.logging.SessionLog`)
- `dontLogMessages`—disable logging
- `setLog`—specify the `java.io.Writer` to which the session logs messages
- `setLogLevel`—specify the level at which the session logs using `oracle.toplink.logging.SessionLog` constants:
  - OFF
  - SEVERE
  - WARNING
  - INFO
  - CONFIG
  - FINE
  - FINER
  - FINEST
  - ALL

[Example 89-1](#) illustrates how to configure a session to use `java.util.logging` package.

### **Example 89-1 Configuring a Session to Use `java.util.logging`**

```
session.setSessionLog(new JavaLog());
```

[Example 89-2](#) illustrates how to configure a session to use the server log that OC4J provides. For more information about server logging, see [Section 87.2.6.1.3, "Server Logging"](#).

### **Example 89-2 Configuring a Session to Use Application Server Logging**

```
session.setSessionLog(new OjdlLog());
```

[Example 89-3](#) illustrates how to configure a session to log to a `java.io.Writer`:

### **Example 89-3 Configuring a Session to Log to a `java.io.Writer`**

```
session.setLog(myWriter);
```

### 89.4.3 How to Configure Logging Using Oracle Enterprise Manager

When you deploy a EclipseLink JPA or TopLink CMP application to Oracle Application Server, you can use Oracle Enterprise Manager to configure TopLink logging.

For more information, see the following:

- [Section A.1, "TopLink Support for Oracle Application Server Manageability and Diagnosability"](#)
- *Oracle Fusion Middleware Administrator's Guide*

### 89.4.4 How to Configure Logging in a Java EE Container

For a TopLink-enabled CMP application deployed to an application server, you do not configure a session directly. In this case, you specify the type of logging by configuring system property `toplink.log.destination` with one of the following values:

- fully qualified file specification (for example, `C:\logs\toplink.log`)—use TopLink native logging to write log messages to the specified file.
- `JAVA`—use `java.util.logging` package to write log messages to any destination you configure in the `<JRE_HOME>/lib/logging.properties` file.
- `SERVER`—use server logging to write log messages to the application server's log file (there is no separate TopLink log file in this case).
- `SYSOUT`—write log messages to `System.out`.

You can set the log level for TopLink standard (default) logging through the `toplink.log.level` system property.

To configure other logging options, use a `customization-class` (see [Section 9.9.1.1, "Configuring pm-properties"](#)).

### 89.4.5 How to Configure a Session to use the `java.util.logging` Package

If you use `java.util.logging` package, then you configure logging options in the `<JRE_HOME>/lib/logging.properties` file. Messages are written to zero or multiple destinations based on this configuration file.

If you configure a session to use `java.util.logging` package, consider the following:

- [logging.properties](#)
- [Formatters](#)
- [Namespace](#)

#### 89.4.5.1 logging.properties

Configure the `logging.properties` file as [Example 89-4](#) illustrates:

**Example 89-4 `java.util.logging` Configuration in `logging.properties`**

```
handlers = java.util.logging.ConsoleHandler
java.util.logging.ConsoleHandler.level = CONFIG
java.util.logging.ConsoleHandler.formatter = oracle.toplink.logging.TopLinkSimpleFormatter
oracle.toplink.LoggingSession.connection.level = CONFIG
```

For information about the types of formatters available, see [Section 89.4.5.2, "Formatters"](#).

### 89.4.5.2 Formatters

TopLink provides two formatters: `TopLinkSimpleFormatter` and `TopLinkXMLFormatter`. They override the corresponding `java.util.logging` formatters and always log session and connection info when available. They also log thread and exception stack trace information at certain levels as specified by the logging level.

### 89.4.5.3 Namespace

Namespace is supported for `java.util.logging`. [Table 89–7](#) lists the static constants defined in `oracle.toplink.sessions.SessionLog` for TopLink components and the corresponding strings in `logging.properties`.

**Table 89–7 Logging Property File Names**

SessionLog	logging.properties
Not Applicable	<code>oracle.toplink</code>
Not Applicable	<code>oracle.toplink.&lt;sessionname&gt;</code>
SQL	<code>oracle.toplink.&lt;sessionname&gt;.sql</code>
TRANSACTION	<code>oracle.toplink.&lt;sessionname&gt;.transaction</code>
EVENT	<code>oracle.toplink.&lt;sessionname&gt;.event</code>
CONNECTION	<code>oracle.toplink.&lt;sessionname&gt;.connection</code>
QUERY	<code>oracle.toplink.&lt;sessionname&gt;.query</code>
CACHE	<code>oracle.toplink.&lt;sessionname&gt;.cache</code>
PROPAGATION	<code>oracle.toplink.&lt;sessionname&gt;.propagation</code>
SEQUENCING	<code>oracle.toplink.&lt;sessionname&gt;.sequencing</code>
EJB	<code>oracle.toplink.&lt;sessionname&gt;.ejb</code>
DMS	<code>oracle.toplink.&lt;sessionname&gt;.dms</code>
EJB_OR_METADATA	<code>oracle.toplink.&lt;sessionname&gt;.ejb_or_metadata</code>
WEAVER	<code>oracle.toplink.&lt;sessionname&gt;.weaver</code>
PROPERTIES	<code>oracle.toplink.&lt;sessionname&gt;.properties</code>
SERVER	<code>oracle.toplink.&lt;sessionname&gt;.server</code>

In the `logging.properties` names listed in [Table 89–7](#), note that `<sessionname>` is the name of the session that the application is running in. For example, if the name of the session is `MyApplication`, then you would use `oracle.toplink.MyApplication.sql` for the SQL logging property.

An application can also define its own namespace and write to it through the logging API, as long as the logger for that namespace is defined in the logging configuration. Otherwise messages are written to the parent logger, `oracle.toplink.<sessionname>`.

## 89.5 Configuring Multiple Mapping Projects

Each session is associated with at least one mapping project (see [Section 89.2](#), "[Configuring a Primary Mapping Project](#)"). You can include additional TopLink mapping projects for a session.

Table 89–8 summarizes which sessions support additional mapping project configuration.

**Table 89–8 Session Support for Additional Mapping Project**

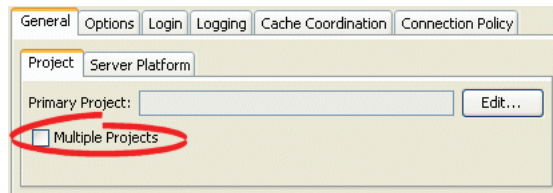
Session	How to Use Oracle JDeveloper	How to Configure Multiple Mapping Projects Using TopLink Workbench	How to Configure Multiple Mapping Projects Using Java
Server and client sessions (see Section 87.3, "Server and Client Sessions")		✓	✓
Session broker and client sessions (see Chapter 87.7, "Session Broker and Client Sessions")		✓	✓
Database sessions (see Section 87.8, "Database Sessions")		✓	✓

### 89.5.1 How to Configure Multiple Mapping Projects Using TopLink Workbench

To specify additional TopLink projects for your session, use this procedure:

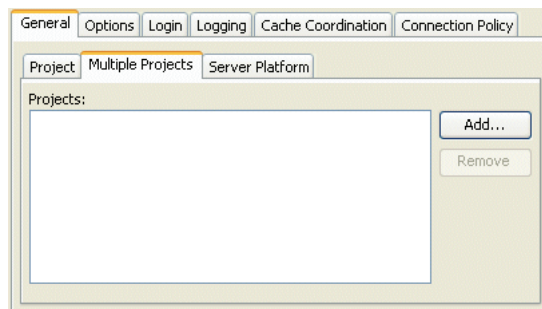
1. Select a server or database session in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.
3. Click the **Project** subtab. The Project subtab appears.

**Figure 89–4 General Tab, Project Subtab, Multiple Projects Options**



4. Select **Multiple Projects** option. The Multiple Projects subtab appears.
5. Click the **Multiple Projects** subtab.

**Figure 89–5 General Tab, Multiple Projects Subtab**



To add an additional mapping project to this session, click **Add**. For more information, see [Section 89.2, "Configuring a Primary Mapping Project"](#).

To remove TopLink mapping projects, select the project file and click **Remove**.

## 89.5.2 How to Configure Multiple Mapping Projects Using Java

Using Java, you can register descriptors from more than one project with a session using the `DatabaseSession` API that [Table 89–9](#) lists. You can register descriptors before login, but you can add independent sets of descriptors after login.

**Table 89–9 DatabaseSession API for Registering Descriptors**

Session Method	Description
<code>addDescriptors(Project)</code>	Add additional descriptor to the session in the form of a project.
<code>addDescriptors(Vector)</code>	Add a vector of individual descriptor files to the session in the form of a project.
<code>addDescriptor(Descriptor)</code>	Add individual descriptor to the session.

## 89.6 Configuring a Performance Profiler

To successfully improve the performance of a TopLink application, you must measure performance before and after each optimization. TopLink provides a variety of built-in performance measuring features (known as profilers) that you can configure at the session level.

[Table 89–10](#) summarizes which sessions support performance profiler configuration.

**Table 89–10 Session Support for Performance Profiler Configuration**

Session	How to Use Oracle JDeveloper	How to Configure a Performance Profiler Using TopLink Workbench	How to Configure a Performance Profiler Using Java
Server and client sessions (see <a href="#">Section 87.3, "Server and Client Sessions"</a> )	✓	✓	✓
Session broker and client sessions (see <a href="#">Section 87.7, "Session Broker and Client Sessions"</a> )	✓	✓	✓
Database sessions (see <a href="#">Section 87.8, "Database Sessions"</a> )	✓	✓	✓

TopLink provides the following profilers:

- TopLink profiler: logs performance statistics for every executed query in a given session (see [Section 12.3, "Measuring TopLink Performance with the TopLink Profiler"](#))
- Oracle Dynamic Monitoring System (DMS): includes DMS instrumentation in essential objects to provide efficient Web browser based monitoring of run-time data in TopLink-enabled applications (see [Section 12.4, "Measuring TopLink Performance with the Oracle Dynamic Monitoring System \(DMS\)"](#))

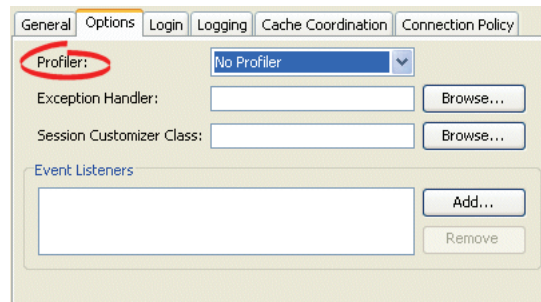


## 89.6.1 How to Configure a Performance Profiler Using TopLink Workbench

To specify the type of profiler in a session, use this procedure:

1. Select a session in the **Navigator**. Its properties appear in the Editor.
2. Click the **Options** tab. The Options tab appears.

**Figure 89–6 Options Tab, Profiler Options**



Use the following information to select the profiler option to use with this session:

Option	Description
No Profiler	Disable all profiling.
DMS	Enable Oracle Dynamic Monitoring (DMS) profiling. For more information, see the following: <ul style="list-style-type: none"> <li>▪ <a href="#">Section 12.4.1, "How to Configure the Oracle DMS Profiler"</a></li> <li>▪ <a href="#">Section 12.4, "Measuring TopLink Performance with the Oracle Dynamic Monitoring System (DMS)"</a>.</li> </ul>
Standard (TopLink)	Enable TopLink profiling. For more information, see the following: <ul style="list-style-type: none"> <li>▪ <a href="#">Section 12.3.1, "How to Configure the TopLink Performance Profiler"</a></li> <li>▪ <a href="#">Section 12.3, "Measuring TopLink Performance with the TopLink Profiler"</a></li> </ul>

## 89.6.2 How to Configure a Performance Profiler Using Java

You can use Java to configure a session with a profiler using `Session` method `setProfiler`, as [Example 89–5](#) shows.

**Example 89–5 Configuring a Session with a TopLink Profiler**

```
session.setProfiler(new PerformanceProfiler());
```

To end a profiling session, use `Session` method `clearProfiler`.

## 89.7 Configuring an Exception Handler

You can associate a single exception handling class with each session. This class must implement the `oracle.toplink.exceptions.ExceptionHandler` interface.

[Table 89–11](#) summarizes which sessions support exception handler configuration.

**Table 89–11 Session Support for Exception Handler Configuration**

Session	How to Use Oracle JDeveloper	How to Configure an Exception Handler Using TopLink Workbench	How to Configure an Exception Handler Using Java
Server and client sessions (see Section 87.3, "Server and Client Sessions")	✓	✓	✓
Session broker and client sessions (see Section 87.7, "Session Broker and Client Sessions")	✓	✓	✓
Database sessions (see Section 87.8, "Database Sessions")	✓	✓	✓

For an example exception handler implementation, see Section 89.7.2, "How to Configure an Exception Handler Using Java".

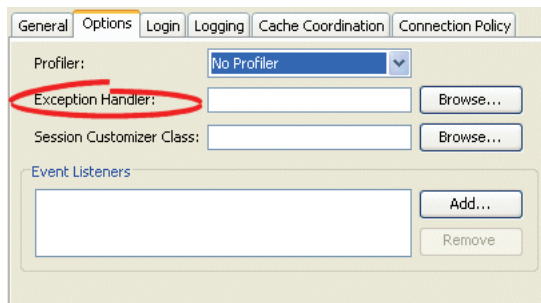
For more information, see Section 87.2.9, "Exception Handlers".

### 89.7.1 How to Configure an Exception Handler Using TopLink Workbench

To specify the exception handler class in a session, use this procedure:

1. Select a session in the **Navigator**. Its properties appear in the Editor.
2. Click the **Options** tab. The Options tab appears.

**Figure 89–7 Options Tab, Exception Handler Field**



3. Complete the **Exception Handler** field.

Click **Browse** and select the exception handler class for this session.

### 89.7.2 How to Configure an Exception Handler Using Java

**Example 89–6** shows an example exception handler implementation. In this implementation, the exception handler always tries to reestablish the connection if it has been reset by peer, but only retries a query if it is an instance of `ReadQuery`. Note that this exception handler either returns the result of the reexecuted `ReadQuery` or throws an exception.

**Example 89–6 Implementing an Exception Handler**

```
session.setExceptionHandler(
    new ExceptionHandler() {
```

```

public Object handleException(RuntimeException exception) {
    if (exception instanceof DatabaseException) {
        DatabaseException dbex = (DatabaseException) exception;
        if ((dbex.getInternalException() instanceof SQLException) &&
            (((SQLException) dbex.getInternalException()).getErrorCode() == MyDriver.CONNECTION_RESET_BY_PEER)) {
            dbex.getAccessor().reestablishConnection(dbex.getSession());
            if (dbex.getQuery() instanceof ReadQuery) {
                return dbex.getSession().executeQuery(dbex.getQuery(), dbex.getQuery().getTranslationRow());
            }
            throw exception;
        }
    }
    throw exception;
}
};

```

---

**Note:** Unhandled exceptions must be rethrown by the exception handler code.

---

## 89.8 Configuring a Session Customizer Class

A session customizer class is a Java class that implements the `oracle.toplink.tools.sessionconfiguration.SessionCustomizer` interface and provides a default (zero-argument) constructor. You can use a session customizer to customize a session at run time on a loaded session before login occurs, similar to how you can use an amendment method to customize a descriptor (see [Section 119.35, "Configuring Amendment Methods"](#)). For example, you can use a session customizer class to define and register session event listeners with the session event manager (see [Section 89.10, "Configuring Session Event Listeners"](#)).

[Table 89–12](#) summarizes which sessions support customizer class configuration.

**Table 89–12** Session Support for Customizer Class Configuration

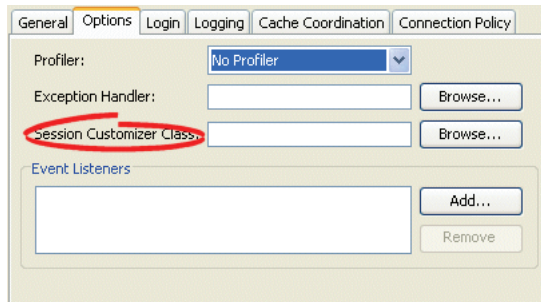
Session	How to Use Oracle JDeveloper	How to Configure Customizer Class Using TopLink Workbench	How to Use Java
Server and client sessions (see <a href="#">Section 87.3, "Server and Client Sessions"</a> )	✓	✓	✓
Session broker and client sessions (see <a href="#">Section 87.7, "Session Broker and Client Sessions"</a> )	✓	✓	✓
Database sessions (see <a href="#">Section 87.8, "Database Sessions"</a> )	✓	✓	✓

For more information, see [Section 87.2.3, "Session Customization"](#).

### 89.8.1 How to Configure Customizer Class Using TopLink Workbench

To specify the session customizer class in a session, use this procedure:

1. Select a session in the **Navigator**. Its properties appear in the Editor.
2. Click the **Options** tab. The Options tab appears.

**Figure 89–8 Options Tab, Session Customizer Class Field**

Click **Browse** and select the customizer class for this session.

## 89.8.2 How to Configure Customizer Class Using Java

When using Java, create a customize class that implements the `oracle.toplink.tools.sessionconfiguration.SessionCustomizer` interface. [Example 89–7](#) illustrates the creation of the session customizer. The customize method contains the configuration of the `Login` owned by the `Session` with the appropriate transaction isolation.

### **Example 89–7 Creating a SessionCustomizer Class**

```
import oracle.toplink.tools.sessionconfiguration.SessionCustomizer;
import oracle.toplink.sessions.Session;
import oracle.toplink.sessions.DatabaseLogin;

public class EmployeeSessionCustomizer implements SessionCustomizer {

    public void customize(Session session) {
        DatabaseLogin login = (DatabaseLogin)session.getDatasourceLogin();
        login.setTransactionIsolation(DatabaseLogin.TRANSACTION_READ_UNCOMMITTED);
    }
}
```

## 89.9 Configuring the Server Platform

The TopLink server platform defines how a session integrates with a Java EE server including the following:

- **Run-time services:** Enables the deployment of a Java Management Extensions (JMX) MBean that allows monitoring of the TopLink session.
- **External transaction controller:** Integrates the TopLink session with the server's Java Transaction API (JTA) service. This should always be used when using EJB or JTA transactions. You configure TopLink to integrate with the container's external transaction service by specifying a TopLink external transaction controller. For more information on external transaction services, see [Section 113.1.2, "Unit of Work Transaction Demarcation"](#).

[Table 89–8](#) summarizes which sessions support a server platform.

**Table 89–13 Session Support for Server Platform**

Session	How to Use Oracle JDeveloper	How to Configure the Server Platform Using TopLink Workbench	How to Configure the Server Platform Using Java
Server and client sessions (see Section 87.3, "Server and Client Sessions")	✓	✓	✓
Session broker and client sessions (see Section 87.7, "Session Broker and Client Sessions")	✓	✓	✓
Database sessions (see Section 87.8, "Database Sessions")	✓	✓	✓

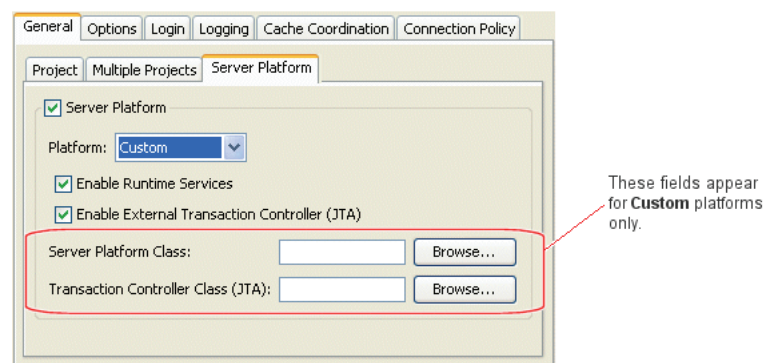
If the primary mapping project that you associate with a session has a persistence type of bean-managed persistence (BMP) or Java objects, you may configure a server platform using Oracle JDeveloper or TopLink Workbench. For more information on primary mapping project, see Section 89.2, "Configuring a Primary Mapping Project".

If the primary mapping project you associate with a session has a persistence type of container-managed persistence (CMP), by default, the TopLink runtime automatically configures a server platform to accommodate the application server on which it is deployed.

### 89.9.1 How to Configure the Server Platform Using TopLink Workbench

To specify the server platform options for a session, use this procedure:

1. Select a session in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.
3. Click the **Server Platform** subtab. The Server Platform subtab appears.

**Figure 89–9 General Tab, Server Platform Subtab**

Use the following information to enter data in each field of the Server Platform subtab:

Field	Description
<b>Server Platform</b>	<p>Check this field if you intend to deploy your application to a Java EE application server.</p> <p>If you check this field, you must configure the target application server by selecting a <b>Platform</b>.</p>
<b>Platform</b>	<p>Select the Java EE application server to which you will deploy your application.</p> <p>TopLink supports the following Java EE application servers:</p> <ul style="list-style-type: none"> <li>▪ WebLogic 10.n</li> <li>▪ WebLogic 9.n</li> <li>▪ OC4J 10.1.3.n</li> <li>▪ SunAS 9</li> <li>▪ WebSphere 6.1</li> <li>▪ Custom</li> </ul> <p>For detailed information about supported application server versions and configuration requirements, see <a href="#">Chapter 8, "Integrating TopLink with an Application Server"</a>.</p> <p>Select <b>Custom</b> if you have created your own <code>oracle.toplink.platform.server.ServerPlatform</code> class to use an application server not currently supported by TopLink or to override an existing <code>ServerPlatform</code>. If you select <b>Custom</b>, you must specify your custom <code>ServerPlatform</code> class by selecting a <b>Server Platform Class</b>.</p> <p>The server platform you select overrides the default server platform set at the sessions configuration level (see <a href="#">Section 88.2, "Creating a Sessions Configuration"</a>).</p>
<b>Enable Runtime Services</b>	<p>Check this field to configure the TopLink runtime to enable the deployment of a JMX MBean that allows monitoring of the TopLink session.</p> <p>To use this feature, you must enable DMS data collection. For more information, see <a href="#">Section 12.4.1, "How to Configure the Oracle DMS Profiler"</a>.</p>
<b>Enable External Transaction Controller (JTA)</b>	<p>Check this field if you intend to integrate your application with an external transaction controller. For more information, see <a href="#">Section 113.1.2, "Unit of Work Transaction Demarcation"</a>.</p> <p>If you configure <b>Platform</b> for a Java EE application server that TopLink supports, the TopLink runtime will automatically select the appropriate external transaction controller class.</p> <p>If you configure <b>Platform</b> as <b>Custom</b>, you must specify an external transaction controller class by selecting an <b>External Transaction Controller</b>.</p>
<b>Server Platform Class</b>	<p>This option is only available if you configure <b>Platform</b> as <b>Custom</b>.</p> <p>Click Browse to select your custom <code>ServerPlatform</code> class.</p>
<b>Transaction Controller Class (JTA)</b>	<p>This option is only available if you configure <b>Platform</b> as <b>Custom</b>.</p> <p>If you checked <b>Enable External Transaction Controller (JTA)</b>, click <b>Browse</b> to select the transaction controller class that corresponds with your custom <code>ServerPlatform</code> class.</p>

## 89.9.2 How to Configure the Server Platform Using Java

When using Java, you must pass the session in a server platform constructor.

[Example 89–8](#) illustrates using a session customizer (see [Section 13.4, "Using the](#)

[Session Customizer Class](#)") to configure a session with a server platform from the `oracle.toplink.platform.server` package.

**Example 89–8 Configuring a Session with a Server Platform**

```
import oracle.toplink.tools.sessionconfiguration.SessionCustomizer;
...
public class MySessionCustomizer implements SessionCustomizer {
    public void customize (Session session) {
        Server server = (Server)session;
        server.setServerPlatform(new WebLogic_10_Platform(DatabaseSession)server));
    }
}
```

## 89.10 Configuring Session Event Listeners

As you perform persistence operations with a session, the session produces various events (see [Section 87.2.5.1, "Session Event Manager Events"](#)) that the TopLink runtime uses to coordinate its various components. You can configure a session with one or more session event listeners (see [Section 87.2.5.2, "Session Event Listeners"](#)) to customize session behavior and debug session operations. For example, session event listeners play an important role in the configuration of isolated sessions (see [Chapter 92, "Configuring Exclusive Isolated Client Sessions for Virtual Private Database"](#)).

Table 89–14 summarizes which sessions support event listeners.

**Table 89–14 Session Support for Event Listeners**

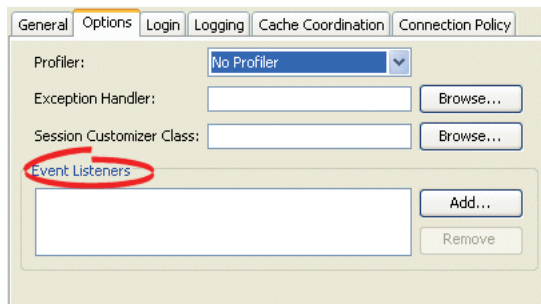
Session	How to Use Oracle JDeveloper	How to Configure Session Event Listeners Using TopLink Workbench	How to Configure Session Event Listeners Using Java
Server and client sessions (see <a href="#">Section 87.3, "Server and Client Sessions"</a> )	✓	✓	✓
Session broker and client sessions (see <a href="#">Section 87.7, "Session Broker and Client Sessions"</a> )	✓	✓	✓
Database sessions (see <a href="#">Section 87.8, "Database Sessions"</a> )	✓	✓	✓

### 89.10.1 How to Configure Session Event Listeners Using TopLink Workbench

#### Session Event Listeners

To specify the event listener class in a session, use this procedure:

1. Select a session in the **Navigator**. Its properties appear in the Editor.
2. Click the **Options** tab. The Options tab appears.

**Figure 89–10 Options Tab, Event Listeners field**

To add a new event listener, click **Add**, then select the event listener class for this session.

To remove an existing event listener, select the **Event Listener** and click **Remove**.

### 89.10.2 How to Configure Session Event Listeners Using Java

[Example 89–9](#) illustrates how to use Java to register a session event listener with a session. TopLink provides a `SessionEventAdapter` to simplify creating a `SessionEventListener`. The `SessionEventAdapter` provides a default implementation of all the methods of the `SessionEventListener` interface. You need only override the specific methods of interest. Typically, you would define session event listeners in a session customizer class (see [Section 89.8, "Configuring a Session Customizer Class"](#)).

#### **Example 89–9 Using the Session Event Adapter to Create a Session Event Listener**

```
...
SessionEventAdapter myEventListener = new SessionEventAdapter() {
    // Listen for PostCommitUnitOfWork events
    public void postCommitUnitOfWork(SessionEvent event) {
        // Call the handler routine
        unitOfWorkCommitted();
    }
};
mySession.getEventManager().addListener(myEventListener);
...
```

For information on how to add logging to your listeners, see [Section 87.2.6, "Logging"](#).

## 89.11 Configuring the Integrity Checker

When you log into a session, TopLink initializes and validates the descriptors you registered with it. By configuring the integrity checker, you can customize this validation process to do the following:

- [Check Database](#)
- [Catch All Exceptions](#)
- [Catch Instantiation Policy Exceptions](#)

[Table 89–15](#) summarizes which sessions support descriptor integrity checking configuration.



**Table 89–15 Session Support for Checking Descriptor Integrity**

Session	How to Use Oracle JDeveloper	How to Use TopLink Workbench	How to Configure the Integrity Checker Using Java
Server and client sessions (see <a href="#">Section 87.3, "Server and Client Sessions"</a> )			✓
Session broker and client sessions (see <a href="#">Section 87.7, "Session Broker and Client Sessions"</a> )			✓
Database sessions (see <a href="#">Section 87.8, "Database Sessions"</a> )			✓

**Check Database**

The `IntegrityChecker` method `setShouldCheckDatabase` specifies whether or not the integrity checker should verify the descriptor's metadata against the database metadata. This will report any errors due to missing or incorrect table or fields specified in the descriptors. This is turned off by default as it adds a significant overhead to connecting a session.

**Catch All Exceptions**

By default, the integrity checker catches all exceptions that occur during initialization, and throws a single exception at the end of initialization reporting all of the errors detected. If you only want the first exception encountered, you can disable this feature using `IntegrityChecker` method `setShouldCatchExceptions(false)`.

**Catch Instantiation Policy Exceptions**

By default, the integrity checker tests the default or configured constructor for each descriptor initialized in the session. To disable this feature, use `IntegrityChecker` method `setShouldCheckInstantiationPolicy(false)`.

**89.11.1 How to Configure the Integrity Checker Using Java**

As [Example 89–10](#) shows, you can configure the integrity checker validation process.

**Example 89–10 Configuring the Integrity Checker**

```
session.getIntegrityChecker().setShouldCheckDatabase(true);
session.getIntegrityChecker().setShouldCatchExceptions(false);
session.getIntegrityChecker().setShouldCheckInstantiationPolicy(false);
session.login();
```

**89.12 Configuring Connection Policy**

Using a connection policy, you can control how a TopLink session acquires and uses read and write connections, including the following:

- [Exclusive Write Connections](#)
- [Lazy Connection Acquisition](#)

[Table 89–15](#) summarizes which sessions support connection policy configuration.

**Table 89–16 Session Support for Connection Policy**

Session	How to Use Oracle JDeveloper	How to Configure Connection Policy Using TopLink Workbench	How to Configure Connection Policy Using Java
Server and client sessions (see <a href="#">Section 87.3, "Server and Client Sessions"</a> )	✓	✓	✓
Session broker and client sessions (see <a href="#">Section 87.7, "Session Broker and Client Sessions"</a> )			
Database sessions (see <a href="#">Section 87.8, "Database Sessions"</a> )			

**Exclusive Write Connections**

An exclusive connection is one that TopLink allocates to a client session for reading (of isolated data) and writing for the duration of the client session's life cycle.

By default, exclusive connections are not used and a client session uses the server session's read connection pool for all non-pessimistic read queries. A connection is obtained from the read connection pool for each read query execution and released back to the pool after the query is executed. A connection is only obtained from the write connection pool for the unit of work commit operation, or, potentially, earlier if data modify queries, or read queries using pessimistic locking are used. The connection will be release back to the write connection pool after the unit of work is committed or released.

Exclusive connections are provided for use with database read security or Virtual Private Database (VPD) support. When using an exclusive connection, you will obtain it from the server session's write connection pool. When you acquire the client, the exclusive connection will be used for read queries to isolated classes (see [Section 87.5, "Isolated Client Sessions"](#)), exclusive read queries, pessimistic read queries, and for the unit of work commit operation. The exclusive connection will only be released when the client session is released. TopLink still acquires a shared connection from the read connection pool for reading nonisolated data. If you use a JTA-managed external connection pool with exclusive connections, do not reuse a client session across JTA transaction boundaries, as the physical JTA database connection is released and acquired from the connection pool relative to the JTA transaction life cycle. A new client session, or the active unit of work, should be used for each JTA transaction. For more information, see [Section 101.6, "Configuring Exclusive Read Connections"](#).

You can also configure exclusive connections on a client-session-by-client-session basis (see [Section 90.4.2, "How to Acquire a Client Session that Uses Exclusive Connections"](#)) and for named queries (see [Section 119.7.1.10, "Configuring Named Query Advanced Options"](#)).

---



---

**Note:** If any client session contains an exclusive connection, you must release the session (see [Section 90.8, "Logging Out of a Session"](#)) when you are finished using it. Oracle does not recommend relying on the finalizer to release the connection when the session is garbage -collected. If you are using an active unit of work in a JTA transaction, you do not need to release the client session—the unit of work will release it after the JTA transaction completes.

---



---

### Lazy Connection Acquisition

By default, TopLink acquires write connections lazily, when you perform the first unit of work commit operation, exclusive read query, or pessimistic read query with your client session. The write connection will also be released after each unit of work it committed or released.

Alternatively, you can configure TopLink to acquire the write connection at the time you acquire a client session, and release the connection when you release the client session.

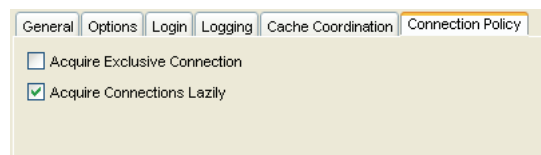
You can also configure lazy connection acquisition on a client-session-by-client-session basis (see [Section 90.4.5, "How to Acquire a Client Session that Does Not Use Lazy Connection Allocation"](#)).

## 89.12.1 How to Configure Connection Policy Using TopLink Workbench

To specify the connection policy in a session, use this procedure:

1. Select a session in the **Navigator**. Its properties appear in the Editor.
2. Click the **Connection Policy** tab. The Connection Policy tab appears.

**Figure 89–11 Connection Policy Tab**



## 89.12.2 How to Configure Connection Policy Using Java

To configure whether or not an exclusive connection is allocated to a particular isolated session, use `ConnectionPolicy` method `setShouldUseExclusiveConnection`.

To define a map of properties used to support an isolated session, use the following `ConnectionPolicy` methods:

- `setProperty(Object key, Object value)`: Adds the property value to the Map under key, overwriting the existing value if key already exists in the Map.
- `Object getProperty(Object key)`: Returns the value associated with key as an Object.
- `boolean hasProperties`: Returns true if one or more properties exist in the Map; otherwise returns false.

The TopLink runtime passes this Map into `SessionEvent` events `PostAcquireExclusiveConnection` and `PreReleaseExclusiveConnection` so that your implementation can make the appropriate PL/SQL calls to the underlying database platform (see [Section 92.2, "Using PostAcquireExclusiveConnection Event Handler"](#) and [Section 92.3, "Using PreReleaseExclusiveConnection Event Handler"](#)).

To configure the session to use a named connection pool, use the `ConnectionPool` constructor that takes a `String` connection pool name as an argument:

```
Session clientSession = server.acquireClientSession(
    new ConnectionPolicy("myConnectionPool")
);
```

## 89.13 Configuring Named Queries at the Session Level

A **named query** is a TopLink query that you create and store, by name, in a session for later retrieval and execution. Named queries improve application performance, because they are prepared once and they (and all their associated supporting objects) can be efficiently reused thereafter making them well-suited for frequently executed operations.

If a named query is global to a project, configure it at the session level. Alternatively, you can configure a named query at the descriptor level (see [Section 119.7](#), "Configuring Named Queries at the Descriptor Level").

Use named queries to specify SQL, EJB QL, or TopLink Expression queries to access your data source.

[Table 89-17](#) summarizes which sessions support named query configuration.

**Table 89-17** Session Support for Named Queries

Session	How to Use Oracle JDeveloper	How to Use TopLink Workbench	How to Configure Named Queries at the Session Level Using Java
Server and client sessions (see <a href="#">Section 87.3</a> , "Server and Client Sessions")			✓
Session broker and client sessions (see <a href="#">Section 87.7</a> , "Session Broker and Client Sessions")			✓
Database sessions (see <a href="#">Section 87.8</a> , "Database Sessions")			✓

After you create a named query, you can execute it by name on the TopLink session (see [Section 109.3](#), "Using Named Queries").

For more information about named queries, see [Section 108.8](#), "Named Queries".

### 89.13.1 How to Configure Named Queries at the Session Level Using Java

You can store a query by name in a Session using Session method `addQuery(String name, DatabaseQuery query)`.

---

---

## Acquiring and Using Sessions at Run Time

After you create and configure sessions, you can use the session manager to acquire a session instance at run time.

This chapter includes the following sections:

- [Introduction to Session Acquisition](#)
- [Acquiring the Session Manager](#)
- [Acquiring a Session from the Session Manager](#)
- [Acquiring a Client Session](#)
- [Acquiring a Historical Session](#)
- [Logging In to a Session](#)
- [Using Session API](#)
- [Logging Out of a Session](#)
- [Storing Sessions in the Session Manager Instance](#)
- [Destroying Sessions in the Session Manager Instance](#)

### 90.1 Introduction to Session Acquisition

Oracle recommends that you export session instances from Oracle JDeveloper TopLink Editor or TopLink Workbench to one or more uniquely named `sessions.xml` files and then use the session manager to load sessions from these `sessions.xml` files.

The TopLink session manager lets you build a series of sessions that are maintained under a single entity. The session manager is a static utility class that loads TopLink sessions from the `sessions.xml` file, caches the sessions by name in memory, and provides a single access point for TopLink sessions.

The session manager supports the following session types:

- Server Session
- Database Session
- SessionBroker

See [Chapter 87, "Introduction to TopLink Sessions"](#) for detailed information on these available sessions.

The session manager has two main functions: it creates instances of the sessions and it ensures that only a single instance of each named session exists for any instance of a session manager.

This is particularly useful for EJB applications in that an enterprise bean can acquire the session manager and acquire the desired session from it.

### 90.1.1 Session Manager

When a client application requires a session, it requests the session from the TopLink session manager. The two main functions of the session manager are to instantiate TopLink sessions for the server, and to hold the sessions for the life of the application. The session manager instantiates database sessions, server sessions, or session brokers based on the configuration information in the `sessions.xml` file.

The session manager instantiates sessions as follows:

- The client application requests a session by name.
- The session manager looks up the session name in the `sessions.xml` file. If the session name exists, the session manager instantiates the specified session; otherwise, it throws an exception.
- After instantiation, the session remains viable until the application is shut down.

### 90.1.2 Multiple Sessions

Oracle recommends that you acquire sessions from the session manager and perform all persistence operations using a client session or the unit of work.

Note that in the case of a server session or a session broker that contains server sessions, after you acquire the session you will acquire a client session from it. From a given server session (or session broker that contains server sessions), you can acquire as many client sessions as you have clients.

Each client can easily manage concurrent access and referential constraints by acquiring a unit of work from its client session and performing all persistence operations using the unit of work.

## 90.2 Acquiring the Session Manager

TopLink maintains only one instance of the session manager class. The singleton session manager maintains all the named TopLink sessions at run time. When an application requests a session by name, the session manager retrieves the specified session from the appropriate configuration file.

As [Example 90-1](#) illustrates, to access the session manager instance, use the `oracle.toplink.tools.sessionmanagement.SessionManager` method `getManager`. You can then use the session manager instance to load TopLink sessions.

#### **Example 90-1 Acquiring a Session Manager Instance**

```
import oracle.toplink.tools.sessionmanagement.SessionManager;  
SessionManager sessionManager = SessionManager.getManager();
```

### 90.3 Acquiring a Session from the Session Manager

When the session manager loads a session that is not yet in its cache, the session manager creates an instance of the appropriate session type and configures it according to the `sessions.xml` file configuration.

---



---

**Note:** To best use the methods associated with the session type that is being instantiated, cast the session that is returned from the `getSession` method. This type must match the session type that is defined in the `sessions.xml` file for the named session.

---



---

This section explains the following:

- [How to Load a Session from sessions.xml Using Defaults](#)
- [How to Load a Session from sessions.xml with an Alternative Class Loader](#)
- [How to Load a Session from an Alternative Session Configuration File](#)
- [How to Load a Session Without Logging In](#)
- [How to Reload and Refresh Session Configuration](#)
- [How to Refresh a Session when the Class Loader Changes](#)

### 90.3.1 How to Load a Session from sessions.xml Using Defaults

If you have a single sessions configuration file (`sessions.xml`) that contains all the session instances created by Oracle JDeveloper or TopLink Workbench, then you can load a session by name, as [Example 90-2](#) illustrates.

#### **Example 90-2 Acquiring a Named Session from Session Manager Using Defaults**

```
// Load a named session (mysession) defined in the sessions.xml file
SessionManager manager = SessionManager.getManager();
Session session = manager.getSession("mysession");
```

In this example, the following session manager default configuration applies:

- **Class loader**—The thread-based class loader is used to find and load the `sessions.xml` resource and resolve any classes referenced in the `sessions.xml` and `project.xml` files.
 

If you acquire the session in an application class, this will typically be the application's class loader, which is correct. In a Java EE application, it is best to specify this as the class loader from a class in the same JAR file that the `sessions.xml` file is deployed in.
- **File**—By default, the file named `sessions.xml` in the root directory relative to the class loader is used.
 

If the file is named differently, or not in the root directory, the relative path must be specified. Relative resource paths in Java must use " / ", not " \ ".
- **Session name**—The name passed into the `getSession` call.
 

This name must be unique for the entire application server, not just unique within the application.
- **Login**—`true`. The session will be connected by default.
 

If you must manually configure the session before login, set this option to `false` (see [Section 90.3.4, "How to Load a Session Without Logging In"](#)).
- **Refresh**—`false`. If already loaded, the same session will be returned.
 

Refresh should only be used, if it is known that the existing session is not being used, and the configuration has changed, such as in a Java EE environment redeployment scenario.

- Verify class loader=false. The session manager will not refresh the session if the class loader changes.

This should normally be set to true. It must be set to true in a Java EE environment, if hot deployment or redeployment to a running application server is required (see [Section 90.3.6, "How to Refresh a Session when the Class Loader Changes"](#)).

### 90.3.2 How to Load a Session from sessions.xml with an Alternative Class Loader

You can use an alternative class loader to load sessions. This is common when your TopLink application integrates with a Java EE container. The session manager uses the class loader to find and load the sessions.xml resource and resolve any classes referenced in the sessions.xml and project.xml files.

In most cases, you use the class loader from the current thread context, as [Example 90-3](#) illustrates. In this example, the session named mysession is loaded from the first file in the application classpath named sessions.xml using the class loader associated with the current thread context.

#### **Example 90-3 Loading a Session Using the Current Thread Context Class Loader**

```
/* Use the specified ClassLoader to load a session (mysession) defined in the
sessions.xml file */
SessionManager manager = SessionManager.getManager();
Session session = manager.getSession(
    "mysession", // session name to load
    Thread.currentThread().getContextClassLoader() // ClassLoader instance to use
);
```

However, if your Java EE container does not support using the current thread context class loader, you can use the class loader from the current class, as [Example 90-4](#) illustrates.

#### **Example 90-4 Loading a Session Using the Current Class's Class Loader**

```
/* Use the specified ClassLoader to load a session (mysession) defined in the
sessions.xml file */
SessionManager manager = SessionManager.getManager();
Session session = manager.getSession(
    "mysession", // session name to load
    this.getClass().getClassLoader() // ClassLoader instance to use
);
```

---

---

**Note:** Oracle Containers for Java EE supports the use of the class loader from the current thread.

---

---

### 90.3.3 How to Load a Session from an Alternative Session Configuration File

If your session instances are contained in multiple, uniquely named session configuration files (sessions.xml files), then you must explicitly create an XMLSessionConfigLoader object initialized with the name of the sessions.xml file and pass that XMLSessionConfigLoader into the SessionManager method getSession, as [Example 90-5](#) illustrates.

The file path you specify is relative to the class loader root directory. Relative resource paths in Java must use the forward slash (/), not back slash (\).



In this example, the session named `mysession` is loaded by the specified class loader from the first file in the application classpath named `toplink-sessions.xml`.

**Example 90–5 Loading a Session from an Alternative Configuration File**

```
// XMLSessionConfigLoader loads the toplink-sessions.xml file
SessionManager manager = SessionManager.getManager();
manager.getSession(
    new XMLSessionConfigLoader("toplink-sessions.xml"),
    "mysession",
    this.class.getClassLoader()
);
```

### 90.3.4 How to Load a Session Without Logging In

The `XMLSessionConfigLoader` (see [Section 90.3.3, "How to Load a Session from an Alternative Session Configuration File"](#)) lets you call a session using the `SessionManager` method `getSession`, without invoking the `Session` method `login`, as [Example 90–6](#) shows. This lets you prepare a session for use and leave `login` to the application.

**Example 90–6 Open Session with No Login**

```
SessionManager manager = SessionManager.getManager();
Session session = manager.getSession(
    new XMLSessionConfigLoader(), // XMLSessionConfigLoader (sessions.xml file)
    "mysession", // session name
    YourApplicationClass.getClassLoader(), // class loader
    false, // do not log in session
    false); // do not refresh session
```

### 90.3.5 How to Reload and Refresh Session Configuration

You can tell the session manager to refresh an existing session from the `sessions.xml` file. Typically, this would only ever be used in a Java EE environment at redeployment time, or after a reset of a running server. You should only use this option when you know that the existing session is not being used.

**Example 90–7 Forcing a Reparse of the sessions.xml File**

```
//In this example, XMLSessionConfigLoader loads sessions.xml from the classpath
SessionManager manager = SessionManager.getManager();
Session session = manager.getSession(
    new XMLSessionConfigLoader(), // XMLSessionConfigLoader (sessions.xml file)
    "mysession", // session name
    YourApplicationClass.getClassLoader(), // class loader
    true, // log in session
    true // refresh session
);
```

### 90.3.6 How to Refresh a Session when the Class Loader Changes

In an unmanaged (POJO) Java EE environment, if you require hot deployment or redeployment to a running application server, you must tell the session manager to refresh your session if the class loader changes, as [Example 90–8](#) shows. This option makes the session manager refresh the session if the class loader changes, which occurs when the application is redeployed. When this option is set to `true`, the same class loader must always be used to retrieve the session.

**Example 90–8 Forcing a Reparse of the sessions.xml File**

```
//In this example, XMLSessionConfigLoader loads sessions.xml from the classpath
SessionManager manager = SessionManager.getManager();
Session session = manager.getSession(
    new XMLSessionConfigLoader(), // XMLSessionConfigLoader (sessions.xml file)
    "mysession", // session name
    YourApplicationClass.getClassLoader(), // class loader
    true, // log in session
    false, // do not refresh session when loaded
    true // do refresh session if class loader changes
);
```

In a CMP Java EE environment, the TopLink runtime and CMP integration handles this for you automatically.

## 90.4 Acquiring a Client Session

Before you can acquire a client session, you must first use the session manager to acquire a server session or a session broker that contains server sessions (see [Section 90.3, "Acquiring a Session from the Session Manager"](#)).

[Table 90–1](#) summarizes the methods used to acquire various types of client sessions from a server session and a session broker session that contains server sessions.

**Table 90–1 Method Used to Acquire a Client Session**

Client Session	Server Session Method	Session Broker Session Method
Regular or Isolated	<code>acquireClientSession()</code>	<code>acquireClientSessionBroker()</code>
Regular or Isolated	<code>acquireClientSession(ConnectionPolicy)</code>	<i>not applicable</i>
Historical	<code>acquireHistoricalSession(AsOfClause)</code>	<code>acquireHistoricalSession(AsOfClause)</code>

The `acquireClientSession` method returns a session of type `ClientSession`.

The `acquireClientSessionBroker` method returns a session of type `SessionBroker`.

In both cases, you should cast the returned object to type `Session` and use it as you would any other session.

For more information, see the following:

- [Section 90.4.1, "How to Acquire an Isolated Client Session"](#)
- [Section 90.5, "Acquiring a Historical Session"](#)
- [Section 90.4.2, "How to Acquire a Client Session that Uses Exclusive Connections"](#)
- [Section 90.4.3, "How to Acquire a Client Session that Uses Connection Properties"](#)
- [Section 90.4.4, "How to Acquire a Client Session that Uses a Named Connection Pool"](#)
- [Section 90.4.5, "How to Acquire a Client Session that Does Not Use Lazy Connection Allocation"](#)

### 90.4.1 How to Acquire an Isolated Client Session

If in your TopLink project you configure all classes as isolated (see [Section 117.11, "Configuring Cache Isolation at the Project Level"](#)), or one or more classes as isolated

(see [Section 119.13, "Configuring Cache Isolation at the Descriptor Level"](#)), then all client sessions that you acquire from a parent server session will be isolated client sessions (see [Section 87.5, "Isolated Client Sessions"](#)).

Using a `ConnectionPolicy`, you can acquire an isolated client session that uses exclusive connections (see [Section 90.4.2, "How to Acquire a Client Session that Uses Exclusive Connections"](#)). This isolated client session can be configured with connection properties for use with the Oracle Virtual Private Database (VPD) feature (see [Section 90.4.3, "How to Acquire a Client Session that Uses Connection Properties"](#)). Typically, you use Oracle Database proxy authentication to pass user credentials to the Oracle Database. For more information about Oracle Database proxy authentication, see [Section 96.1.4.2, "Oracle Database Proxy Authentication"](#).

For more information about VPD, see [Section 87.5.1, "Isolated Client Sessions and Oracle Virtual Private Database \(VPD\)"](#).

## 90.4.2 How to Acquire a Client Session that Uses Exclusive Connections

[Example 90–9](#) illustrates how to configure a `ConnectionPolicy` and use it to acquire a client session that uses exclusive connections.

### **Example 90–9 Acquiring a Client Session that Uses Connection Properties**

```
ConnectionPolicy connectionPolicy = new ConnectionPolicy();
// Use an exclusive connection for the session
connectionPolicy.setShouldUseExclusiveConnection(true);

Session clientSession = server.acquireClientSession(connectionPolicy);
// By default, an exclusive connection will be acquired lazily
```

An exclusive connection is allocated from a shared connection pool. The connection is dedicated to the client session that acquires it.

---

**Note:** Typically, the life cycle of a client session is the duration of a server request. However, if you are using JTA, it is the life cycle of a JTA transaction.

You cannot hold the client session across the JTA transaction boundaries. If you are not using a unit of work in your transaction and you are configuring a client session to use an exclusive connection (see [Chapter 92, "Configuring Exclusive Isolated Client Sessions for Virtual Private Database"](#)), you must explicitly acquire and release the session when you are finished using it. Although client sessions have a finalizer that would release the session when it is garbage-collected, you must not rely on the finalizer and release the exclusive client session (or a non-lazy session) in the application to release the data source connection. Note that the requirement to release the session is not JTA-specific.

If you are using a unit of work (see [Chapter 115, "Using Advanced Unit of Work API"](#)), you do not have to worry about releasing its client session, as the unit of work always automatically releases it at the end of the JTA transaction.

---

A named query can also use an exclusive connection (see [Section 119.7.1.10, "Configuring Named Query Advanced Options"](#)).

For more information, see the following:

- [Section 90.4.5, "How to Acquire a Client Session that Does Not Use Lazy Connection Allocation"](#)
- [Section 89.12, "Configuring Connection Policy"](#).

### 90.4.3 How to Acquire a Client Session that Uses Connection Properties

[Example 90–10](#) illustrates how to configure a `ConnectionPolicy` and use it to acquire a client session that uses connection properties. In this example, the properties are used by the Oracle VPD feature (see [Section 87.5.1, "Isolated Client Sessions and Oracle Virtual Private Database \(VPD\)"](#)). You can use connection properties for other application purposes.

#### **Example 90–10 Acquiring an Isolated Session Using Connection Properties**

```
ConnectionPolicy connectionPolicy = new ConnectionPolicy();  
// Set VPD specific properties to be used in the events  
connectionPolicy.setProperty("userLevel", new Integer(5));
```

```
Session clientSession = server.acquireClientSession(connectionPolicy);
```

For more information, see [Section 89.12, "Configuring Connection Policy"](#).

### 90.4.4 How to Acquire a Client Session that Uses a Named Connection Pool

Before you can acquire a client session that uses a named connection pool, you must configure your session with a named connection pool. For more information on named connection pools, see [Section 96.1.6.5, "Application-Specific Connection Pools"](#). For more information on creating a named connection pool, see [Section 100.1, "Introduction to the Internal Connection Pool Creation"](#).

To acquire a client session that uses a named connection pool, use `Server` method `acquireClientSession`, passing in a `ConnectionPolicy` configured with the desired connection pool. The acquired `ClientSession` uses connections from the specified pool for writes (reads still go through the `Server` read connection pool).

[Example 90–11](#) illustrates how to configure a `ConnectionPolicy` to specify a named connection pool named `myConnectionPool`.

#### **Example 90–11 Acquiring a Client Session that Uses a Named Connection Pool**

```
// Assuming you created a connection pool named "myConnectionPool"  
Session clientSession = server.acquireClientSession(  
    new ConnectionPolicy("myConnectionPool")  
);
```

For more information, see [Section 89.12, "Configuring Connection Policy"](#).

### 90.4.5 How to Acquire a Client Session that Does Not Use Lazy Connection Allocation

By default, the server session does not allocate a data source connection for a client session until a transaction starts (a lazy data source connection). Alternatively, you can acquire a client session that allocates a connection immediately.

[Example 90–12](#) illustrates how to configure a `ConnectionPolicy` to specify that lazy connection allocation is not used.

#### **Example 90–12 Acquiring a Client Session that Does Not Use Lazy Connections**

```
ConnectionPolicy connectionPolicy = new ConnectionPolicy();  
connectionPolicy.setIsLazy(false);  
Session clientSession = server.acquireClientSession(connectionPolicy);
```

For more information, see [Section 89.12, "Configuring Connection Policy"](#).

## 90.5 Acquiring a Historical Session

After you configure TopLink to access historical data (see [Section 93.1, "Introduction to Historical Session Configuration"](#)), you can query historical data using any session type.

When you query historical data using a regular client session or database session, you must always set `ObjectLevelReadQuery` method `maintainCache` to `false` in order to prevent old (historical) data from corrupting the session cache. However, you can query both current and historical object versions.

As a convenience, TopLink provides a historical session to simplify this process. When you query historical data using a historical session, you do not need to set `ObjectLevelReadQuery` method `maintainCache` to `false`. However, you can query objects only as of the specified time.

Before you can acquire a historical session, you must first use the session manager to acquire a server session.

To acquire a historical session, use `Server` method `acquireHistoricalSession` passing in an `AsOfClause`.

The `AsOfClause` specifies a point in time that applies to all queries and expressions subsequently executed on the historical session. The historical session's cache is a read-only snapshot of object versions as of the specified time. Its cache is isolated from its parent server session's shared object cache.

## 90.6 Logging In to a Session

Before you can use a session, you must first log in to the session using `Session` method `login`.

By default, when you load a session using the session manager, TopLink automatically logs in to the session using the zero-argument `login` method. For information on loading a session without automatically logging into the session, see [Section 90.3.4, "How to Load a Session Without Logging In"](#).

If you load a session without logging in, you can choose from the following signatures of the `login` method:

- `login()`: Use the `Login`, user name, and password defined in the corresponding `sessions.xml` file.
- `login(Login login)`: Override the `Login` defined in the corresponding `sessions.xml` file with the specified `Login`.
- `login(String username, String password)`: Override the user name and password defined in the corresponding `sessions.xml` file with the specified user name and password.

When you log in to a session broker, the session broker logs in all contained sessions and initializes the descriptors in the sessions. After `login`, the session broker appears and functions as a regular session. TopLink handles the multiple database access transparently.

## 90.7 Using Session API

For more information on using session API, for caching, see [Chapter 102, "Introduction to Cache"](#).

For more information on using session API for queries, see [Chapter 108, "Introduction to TopLink Queries"](#).

For more information on using session API for transactions, see [Chapter 113, "Introduction to TopLink Transactions"](#).

## 90.8 Logging Out of a Session

When you are finished using a server session, session broker session, or database session, you must log out of the session using `Session` method `logout`. Logging out of a session broker session logs out of all sessions registered with the session broker.

When you are finished using a client session, you must release the session using `Session` method `release`.

You can configure a `Session` with a finalizer to release the session using `Session` method `setIsFinalizersEnabled(true)`. By default, finalizers are disabled. If you choose to enable a finalizer for a session, you should do so only as a last resort. Oracle recommends that you always log out of or release your sessions.

## 90.9 Storing Sessions in the Session Manager Instance

Although Oracle recommends that you export all session instances from Oracle JDeveloper or TopLink Workbench to one or more `sessions.xml` files, alternatively, you can manually create a session in your application and, as [Example 90–13](#) illustrates, manually store it in the session manager using `SessionManager` method `addSession`. Then, you can acquire a session by name using the `SessionManager` method `getSession`.

---

---

**Note:** The `addSession` method is not necessary if you are loading sessions from a session configuration file.

---

---

### *Example 90–13 Storing Sessions Manually in the Session Manager*

```
// create and log in to the session programmatically
Session theSession = project.createDatabaseSession();
theSession.login();
// store the session in the SessionManager instance
SessionManager manager = SessionManager.getManager();
manager.addSession("mysession", theSession);
// retrieve the session
Session session = SessionManager.getManager().getSession("mysession");
```

## 90.10 Destroying Sessions in the Session Manager Instance

You can destroy sessions individually by name or destroy all sessions.

---

---

**Note:** You should only do this when a Java EE application is un-deployed, or when the entire application is shut down and only when it is known that the session is no longer in use. You should log out of a session before destroying it (see [Section 90.8, "Logging Out of a Session"](#)). If you do not log out of a session, the session manager will at the time you use it to destroy a session.

---

---

To destroy one session instance by name, use `SessionManager` method `destroySession`, as [Example 90–14](#) illustrates. If the specified session is not in the session manager cache, a `ValidationException` is thrown.

**Example 90–14 Destroying a Session in the Session Manager**

```
SessionManager manager = SessionManager.getManager();
Server server = (Server) manager.getSession("myserversession");
...
// Destroy session by name. If the session named myserversession is not in the
// session manager cache, throw a ValidationException
manager.destroySession("myserversession");
```

To destroy all session instances, use the `SessionManager` method `destroyAllSessions`, as [Example 90–15](#) illustrates.

**Example 90–15 Destroying All Sessions in the Session Manager**

```
SessionManager manager = SessionManager.getManager();
Server server = (Server) manager.getSession("myserversession");
SessionBroker broker = (SessionBroker) manager.getSession("mysessionbroker");
...
// Destroy all sessions stored in the session manager
manager.destroyAllSessions();
```





## Configuring Server Sessions

This chapter describes the various components that you must configure to use server and client sessions.

This chapter includes the following sections:

- [Introduction to Server Session Configuration](#)
- [Configuring Internal Connection Pools](#)
- [Configuring External Connection Pools](#)

### 91.1 Introduction to Server Session Configuration

Table 91–1 lists the configurable options for server sessions.

**Table 91–1 Configurable Options for Server Sessions**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Internal connection pools (see <a href="#">Section 91.2, "Configuring Internal Connection Pools"</a> )	✓	✓	✓
Primary mapping project (see <a href="#">Section 89.2, "Configuring a Primary Mapping Project"</a> )	✓	✓	✓
Session login (see <a href="#">Section 89.3, "Configuring a Session Login"</a> )	✓	✓	✓
Logging (see <a href="#">Section 89.4, "Configuring Logging"</a> )	✓	✓	✓
External connection pools (see <a href="#">Section 91.3, "Configuring External Connection Pools"</a> )	✓	✓	✓
Multiple mapping projects (see <a href="#">Section 89.5, "Configuring Multiple Mapping Projects"</a> )	✓	✓	✓
Performance profiler (see <a href="#">Section 89.6, "Configuring a Performance Profiler"</a> )	✓	✓	✓
Exception handler (see <a href="#">Section 89.7, "Configuring an Exception Handler"</a> )	✓	✓	✓
Session customizer class (see <a href="#">Section 89.8, "Configuring a Session Customizer Class"</a> )	✓	✓	✓
Server platform (see <a href="#">Section 89.9, "Configuring the Server Platform"</a> )	✓	✓	✓
Session event listener (see <a href="#">Section 89.10, "Configuring Session Event Listeners"</a> )	✓	✓	✓
Coordinated cache (see <a href="#">Section 103, "Configuring a Coordinated Cache"</a> )	✓	✓	✓
Integrity checker (see <a href="#">Section 89.11, "Configuring the Integrity Checker"</a> )	✓	✓	✓

**Table 91–1 (Cont.) Configurable Options for Server Sessions**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Named queries (see <a href="#">Section 89.13</a> , "Configuring Named Queries at the Session Level")			✓

## 91.2 Configuring Internal Connection Pools

An internal connection pool is a collection of reusable connections to a single data source provided by any session that persists to a data source. By default, such a session provides both an internal read and write connection pool.

In this case, you can do the following:

- Configure read and write connection pool options such as minimum and maximum number of connections, alternate connection configuration, and properties (arbitrary, application-specific named values).
- Create named connection pools for whatever application-specific purpose you choose.
- Create sequence connection pools that TopLink uses exclusively for obtaining object identifiers.

For more information about creating and configuring internal connection pools, see the following:

- [Chapter 100](#), "Creating an Internal Connection Pool"
- [Chapter 101](#), "Configuring an Internal Connection Pool"

For more information about configuring the type of connection pool your session uses, see [Section 97.4](#), "Configuring External Connection Pooling".

## 91.3 Configuring External Connection Pools

An external connection pool is a collection of reusable connections to a single data source provided by a JDBC driver or Java EE container.

By default, a session uses internal connection pools (see [Section 91.2](#), "Configuring Internal Connection Pools"). For more information about configuring a session to use an external connection pool, see [Section 97.4](#), "Configuring External Connection Pooling".

## Configuring Exclusive Isolated Client Sessions for Virtual Private Database

This chapter describes the various components that you must configure before you can acquire an exclusive isolated client session from a server session.

This chapter includes the following sections:

- [Introduction to Exclusive Isolated Client Session Configuration](#)
- [Using PostAcquireExclusiveConnection Event Handler](#)
- [Using PreReleaseExclusiveConnection Event Handler](#)
- [Using NoRowsModifiedSessionEvent Event Handler](#)
- [Accessing Indirection](#)

### 92.1 Introduction to Exclusive Isolated Client Session Configuration

Table 92–1 lists the configurable options for isolated sessions.

**Table 92–1 Configurable Options for Isolated Client Sessions**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Cache isolation at the descriptor level (see <a href="#">Section 119.13</a> , "Configuring Cache Isolation at the Descriptor Level")			✓
Connection policy (see <a href="#">Section 89.12</a> , "Configuring Connection Policy")			✓
Oracle Database proxy authentication using Java (see <a href="#">Section 98.8.1</a> , "How to Configure Oracle Database Proxy Authentication Using Java")			✓
PostAcquiredExclusiveConnection event handler (see <a href="#">Section 92.2</a> , "Using PostAcquireExclusiveConnection Event Handler")			✓
PreReleaseExclusiveConnection event handler (see <a href="#">Section 92.3</a> , "Using PreReleaseExclusiveConnection Event Handler")			✓
NoRowsModifiedSessionEvent event handler (see <a href="#">Section 92.4</a> , "Using NoRowsModifiedSessionEvent Event Handler")			✓
ValidationException handler (see <a href="#">Section 92.5</a> , "Accessing Indirection")			✓

These options are used throughout the isolated session life cycle (see [Section 87.5.1.3](#), "Isolated Client Session Life Cycle").

## 92.2 Using PostAcquireExclusiveConnection Event Handler

TopLink raises this event after an exclusive connection is allocated to an isolated session after the user has logged in to the database with it.

If you are using Oracle Database proxy authentication (see [Section 96.1.4.2, "Oracle Database Proxy Authentication"](#)), then you do not need to implement this session event handler.

If you are not using Oracle Database proxy authentication, then, as part of the isolated session life cycle, you must implement a `SessionEventListener` for `SessionEvent.PostAcquireExclusiveConnection`.

---



---

**Note:** You must add this session event listener to the server session from which you acquire your isolated client session. You cannot add them to the isolated client session itself. For more information, see [Section 89.10, "Configuring Session Event Listeners"](#)

---



---

### 92.2.1 How to Use Java

The `SessionEvent.PostAcquireExclusiveConnection` event listener is your opportunity to authenticate your user and interact with the underlying database platform: for example, to execute PL/SQL to create VPD packages and set VPD context information.

[Example 92–1](#) illustrates a typical session event listener used to handle `postAcquireExclusiveConnection` events for an isolated session.

#### **Example 92–1 Session Event Listener for an Isolated Session**

```
class VPDEventListener extends SessionEventAdaptor{
    public void postAcquireExclusiveConnection(SessionEvent event){
        ClientSession session = (ClientSession)event.getSession();
        // Get property set on the ConnectionPolicy prior to acquiring the connection
        String userLevel = session.getConnectionPolicy().getProperty("userLevel");
        // Make the Stored Procedure call for VPD to set up the Context Information
        session.executeNonSelectingSQL("StoreProcSetContextUser(" + userLevel + ")");
    }
}
```

To get the required user credentials, use `ClientSession` method `getConnectionPolicy` to get the associated `ConnectionPolicy`, and then use `ConnectionPolicy` method `getProperty`. The `ConnectionPolicy` associated with the `ClientSession` should contain all required user credentials (see [Section 89.12, "Configuring Connection Policy"](#)).

After you implement the required `SessionEventListener`, add it to the parent server session from which you acquire your isolated client session. For more information, see [Section 89.10, "Configuring Session Event Listeners"](#).

## 92.3 Using PreReleaseExclusiveConnection Event Handler

TopLink raises a `SessionEvent.PreReleaseExclusiveConnection` event after you call the isolated session method `release`.

If you are using Oracle Database proxy authentication (see [Section 96.1.4.2, "Oracle Database Proxy Authentication"](#)), then you do not need to implement this session event handler.

If you are not using Oracle Database proxy authentication, then as part of the isolated session life cycle, you must implement a `SessionEventListener` for `SessionEvent.PreReleaseExclusiveConnection`.

---

**Note:** You must add this session event listener to the server session from which you acquire your isolated client session. You cannot add them to the isolated client session itself. For more information, see [Section 89.10, "Configuring Session Event Listeners"](#)

---

### 92.3.1 How to Use Java

The `SessionEvent.PreReleaseExclusiveConnection` event listener gives you an opportunity to interact with the underlying database platform: for example, to perform any VPD-specific cleanup such as executing PL/SQL to delete VPD packages or context information.

[Example 92-1](#) illustrates a typical session event listener used to handle `preReleaseExclusiveConnection` events for an isolated session.

#### **Example 92-2 Session Event Listener for an Isolated Session**

```
class VPDEventListener extends SessionEventAdaptor{
    public void preReleaseExclusiveConnection(SessionEvent event){
        Session session = event.getSession();
        // Make the Stored Procedure call for VPD to reset the Context Information
        session.executeNonSelectingSQL("StoreProcResetContext()");
    }
}
```

To get the required user credentials, use `ClientSession` method `getConnectionPolicy` to get the associated `ConnectionPolicy`, and then use `ConnectionPolicy` method `getProperty`. The `ConnectionPolicy` associated with the `ClientSession` should contain all required user credentials (see [Section 89.12, "Configuring Connection Policy"](#)).

After you implement the required `SessionEventListener`, add it to the parent server session, from which you acquire your isolated client session. For more information, see [Section 89.10, "Configuring Session Event Listeners"](#).

## 92.4 Using NoRowsModifiedSessionEvent Event Handler

As part of your general error handling strategy, you should implement a `SessionEventListener` for `SessionEvent.NoRowsModifiedSessionEvent`.

`TopLink` raises this event when an update or delete query is executed against the database, but no rows are updated, that is, a zero row count is returned.

If optimistic locking is not enabled and you query the database and violate your VPD security configuration, no exception is thrown: the query simply returns zero rows updated.

If optimistic locking is enabled and you query the database and violate your VPD security configuration, an `OptimisticLockException` is thrown even though the root cause of the failure was a security violation, not an optimistic locking issue.

---

---

**Note:** You must add this session event listener to the server session from which you acquire your isolated client session. You cannot add them to the isolated client session itself. For more information, see [Section 89.10, "Configuring Session Event Listeners"](#)

---

---

### 92.4.1 How to Use Java

This event listener gives you an opportunity to determine whether the update failure was due to a security violation (in which case you should not retry the operation), or due to an optimistic lock issue (in which case a retry may be appropriate).

You can use the existing session event API, such as `getQuery().getResult()`, to get the affected object, if any.

After you implement the required `SessionEventListener`, add it to the parent server session, from which you acquire your isolated client session. For more information, see [Section 89.10, "Configuring Session Event Listeners"](#).

## 92.5 Accessing Indirection

As part of your general error handling strategy, your application should be prepared to handle a `ValidationException` of type `ISOLATED_SESSION_IS_NO_LONGER_AVAILABLE`.

`TopLink` throws an `ISOLATED_SESSION_IS_NO_LONGER_AVAILABLE` when a client triggers the indirection (lazy loading) on an isolated object when the isolated session used to load that object is no longer available, that is, after you call the isolated session method `release`.

---

---

**Note:** Ensure that you have instantiated every relationship that you need prior to calling the `release` method: to instantiate a one-to-one relationship, call the `get` method; to instantiate a one-to-many relationship, call the `size` method on the collection.

---

---

For more information, see the following:

- [Section 87.2.9, "Exception Handlers"](#)
- [Section 89.7, "Configuring an Exception Handler"](#)

---

---

## Configuring Historical Sessions

This chapter describes the various components that you must configure in order to be able to use historical sessions.

This chapter includes the following section:

- [Introduction to Historical Session Configuration](#)

For more information about historical sessions, see [Section 87.6, "Historical Sessions"](#).

### 93.1 Introduction to Historical Session Configuration

There are two following ways to configure TopLink to access the historical versions of objects maintained by your data source:

- using an Oracle platform (see [Section 93.1.1, "How to Configure Historical Sessions Using an Oracle Platform"](#))
- using TopLink `HistoryPolicy` (see [Section 93.1.2, "How to Configure Historical Sessions Using a TopLink HistoryPolicy"](#))

#### 93.1.1 How to Configure Historical Sessions Using an Oracle Platform

Oracle9i Database (or later) automatically maintains historical versions of objects and extends SQL with an `AS_OF` clause used to query this historical data. Oracle refers to these as flashback queries.

If you configure your `Session` with an `OraclePlatform` (see [Section 98.2, "Configuring a Relational Database Platform at the Session Level"](#)) for Oracle9i Database (or later), you can query the historical versions of objects automatically maintained by Oracle Database.

No further session configuration is required.

For more information, see the following:

- [Section 90.5, "Acquiring a Historical Session"](#)
- [Section 108.11, "Historical Queries"](#).

#### 93.1.2 How to Configure Historical Sessions Using a TopLink HistoryPolicy

If you use a schema that you designed to maintain historical versions of objects and if that schema can be described by TopLink `HistoryPolicy`, you can query the historical versions of objects maintained by your database in accordance with your schema.

For more information, see the following:

- [Section 119.31, "Configuring a History Policy"](#)
- [Section 90.5, "Acquiring a Historical Session"](#)
- [Section 108.11, "Historical Queries"](#).



## Configuring Session Broker and Client Sessions

This chapter describes the various components that you must configure in order to use session broker sessions.

This chapter includes the following sections:

- Introduction to Session Broker and Client Session Configuration
- Removing, Renaming, or Adding Sessions

### 94.1 Introduction to Session Broker and Client Session Configuration

Table 94–1 lists the configurable options for session broker sessions.

**Table 94–1 Configurable Options for Session Broker Session**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Removing, renaming, or adding sessions (see Section 94.2, "Removing, Renaming, or Adding Sessions")	✓	✓	✓
Primary mapping project (see Section 89.2, "Configuring a Primary Mapping Project")	✓	✓	✓
Session login (see Section 89.3, "Configuring a Session Login")	✓	✓	✓
Logging (see Section 89.4, "Configuring Logging")	✓	✓	✓
Multiple mapping projects (see Section 89.5, "Configuring Multiple Mapping Projects")	✓	✓	✓
Performance profiler (see Section 89.6, "Configuring a Performance Profiler")	✓	✓	✓
Exception handler (see Section 89.7, "Configuring an Exception Handler")	✓	✓	✓
Session customizer class (see Section 89.8, "Configuring a Session Customizer Class")	✓	✓	✓
Server platform (see Section 89.9, "Configuring the Server Platform")	✓	✓	✓
Session event listeners (see Section 89.10, "Configuring Session Event Listeners")	✓	✓	✓
Coordinated cache (see Section 103, "Configuring a Coordinated Cache")	✓	✓	✓
Integrity checker (see Section 89.11, "Configuring the Integrity Checker")	✓	✓	✓
Named queries (see Section 89.13, "Configuring Named Queries at the Session Level")			✓

## 94.2 Removing, Renaming, or Adding Sessions

You can manage the sessions contained by a session broker with Oracle JDeveloper or TopLink Workbench.

---

**Note:** Add only sessions of the same type to any given session broker. Do not mix sessions of different types within a session broker.

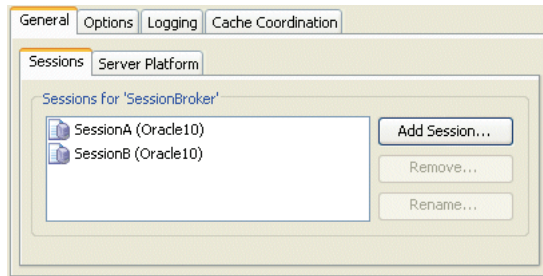
---

### 94.2.1 How to Use TopLink Workbench to Remove, Rename, or Add Sessions

To add sessions to, remove sessions from, or rename sessions in a session broker, use this procedure:

1. Select a session broker in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.
3. Click the **Sessions** subtab. The Sessions subtab appears.

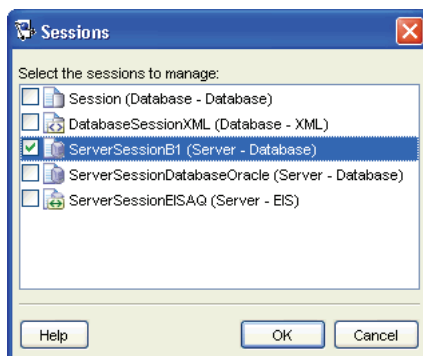
**Figure 94–1 General Tab, Sessions Subtab**



To manage the sessions in this session broker, choose one of the following:

- To remove a session, select the session in the Sessions tab's list and click **Remove**.
- To rename a session, select the session in the Sessions tab's list and click **Rename**. The Rename dialog box appears. Enter a new name and click **OK**.
- To add a session, click **Add Session**. The Sessions dialog box appears showing a list of all the sessions currently configured in the session configuration that owns this session broker.

**Figure 94–2 Sessions Dialog Box**



Check the sessions in the Session dialog that you want to add to the session broker and click **OK**.

## Configuring Database Sessions

This chapter describes the various components that you must configure in order to use database sessions.

This chapter includes the following sections:

- [Introduction to Database Session Configuration](#)
- [Configuring External Connection Pools](#)

### 95.1 Introduction to Database Session Configuration

Table 95–1 lists the configurable options for database sessions.

**Table 95–1 Configurable Options for Database Session**

Option	Oracle JDeveloper	TopLink Workbench	Java
External connection pools (see Section 95.2, "Configuring External Connection Pools")	✓	✓	✓
Primary mapping project (see Section 89.2, "Configuring a Primary Mapping Project")	✓	✓	✓
Session login (see Section 89.3, "Configuring a Session Login")	✓	✓	✓
Logging (see Section 89.4, "Configuring Logging")	✓	✓	✓
Multiple mapping projects (see Section 89.5, "Configuring Multiple Mapping Projects")	✓	✓	✓
Performance profiler (see Section 89.6, "Configuring a Performance Profiler")	✓	✓	✓
Exception handler (see Section 89.7, "Configuring an Exception Handler")	✓	✓	✓
Session customizer class (see Section 89.8, "Configuring a Session Customizer Class")	✓	✓	✓
Server platform (see Section 89.9, "Configuring the Server Platform")	✓	✓	✓
Session event listeners (see Section 89.10, "Configuring Session Event Listeners")	✓	✓	✓
Coordinated cache (see Chapter 103, "Configuring a Coordinated Cache")	✓	✓	✓
Integrity checker (see Section 89.11, "Configuring the Integrity Checker")	✓	✓	✓
Named queries (see Section 89.13, "Configuring Named Queries at the Session Level")			✓

## 95.2 Configuring External Connection Pools

Unlike a server session, a database session does not provide internal connection pools. A database session only has a single database connection that it uses for its life cycle.

Oracle recommends that you use a server and client session in a three-tier environment. Alternatively, you can use a database session with an external connection pool (see [Section 97.4, "Configuring External Connection Pooling"](#)): in this case, you should allocate a new database session per user/thread or request.

---

---

**WARNING:** Do not allow the concurrent use of a database session by multiple users/threads.

---

---

The usage of an external connection pool reduces the number of the database session login and logout attempts to acquire the database connection.

# Part XXII

---

## Data Access

This part describes how TopLink defines connections to a data source. It contains the following chapters:

- [Chapter 96, "Introduction to Data Access"](#)  
This chapter describes each of the different TopLink data source login types and important data access concepts.
- [Chapter 97, "Configuring a Data Source Login"](#)  
This chapter explains how to configure TopLink data source login options common to two or more data source login types.
- [Chapter 98, "Configuring a Database Login"](#)  
This chapter explains how to configure a TopLink database login for a session used in a relational project.
- [Chapter 99, "Configuring an EIS Login"](#)  
This chapter explains how to configure a TopLink EIS login for a session used in an EIS project.
- [Chapter 100, "Creating an Internal Connection Pool"](#)  
This chapter explains how to create an internal connection pool.
- [Chapter 101, "Configuring an Internal Connection Pool"](#)  
This chapter explains how to configure an internal connection pool.



---

---

## Introduction to Data Access

One of the most important functions of a session is to provide access to a data source. This chapter explains session components specific to accessing a data source.

This chapter includes the following sections:

- [Data Access Concepts](#)
- [Data Access API](#)

### 96.1 Data Access Concepts

This section describes concepts unique to TopLink data access, including the following:

- [Externally Managed Transactional Data Sources](#)
- [Data Source Login Types](#)
- [Data Source Platform Types](#)
- [Authentication](#)
- [Connections](#)
- [Connection Pools](#)

#### 96.1.1 Externally Managed Transactional Data Sources

A TopLink transactional data source is *externally managed* if the connection pool is managed by a transaction service (such as an application server controlled transaction or a JTA transaction). A JTA managed data source or connection pool is commonly used in Java EE applications and normally required in EJB applications. Use an externally-managed connection pool as follows:

- Configure the session to use an `ExternalTransactionController` to integrate TopLink's unit of work with the external transaction service (see [Section 115.13, "Integrating the Unit of Work with an External Transaction Service"](#)).
- Use the `external-transaction-control` option to specify the connection's login and inform TopLink that the connection is maintained by the external controller (see [Section 97.4, "Configuring External Connection Pooling"](#)).
- You may need to configure the TopLink read connection pool or sequence connection pool to use a non-JTA connection pool in order to avoid transactional overhead (see [Section 96.1.6.3, "Default \(Write\) and Read Connection Pools"](#)).

For more information on transactional data sources, see the following:

- [Section 113.1.2.1, "JTA Controlled Transactions"](#)
- [Section 113.1.2.2, "OTS Controlled Transactions"](#)
- [Section 113.1.2.3, "CMP-Controlled Transactions"](#)

Refer to [Chapter 113, "Introduction to TopLink Transactions"](#) for more information on TopLink transactions.

## 96.1.2 Data Source Login Types

The login (if any) associated with a session determines how the TopLink runtime connects to the project's data source.

A login includes details of data source access, such as authentication, use of connection pools, and use of external transaction controllers. A `Login` owns a data source platform.

A data source platform includes options specific to a particular data source including such as binding, use of native SQL, use of batch writing, and sequencing. For more information about platforms, see [Section 96.1.3, "Data Source Platform Types"](#).

For projects that do not persist to a data source, a login is not required. For projects that do persist to a data source, a login is always required.

In TopLink Workbench, the project type determines the type of login that the project uses, if applicable.

You can use a login in a variety of roles. A login's role determines where and how you create it. The login role you choose depends on the type of project you are creating and how you intend to use the login, as follows:

- [Section 15.2.4.1, "POJO Session Role"](#)
- [Section 15.2.4.2, "CMP Deployment Role"](#)
- [Section 15.2.4.3, "Development Role"](#)

There is a session login type for each project type that persists to a data source. The following are the types:

- `DatabaseLogin`
- `EISLogin`

Note that there is no XML login. TopLink XML projects are used for nonpersistent, in-memory object to XML data transformation and consequently there is no data source to log in to. For more information about persistent and nonpersistent projects, see [Section 15.2.3, "Persistent and Nonpersistent Projects"](#).

For additional information, see the following:

- [Section 15.2.4, "Projects and Login"](#)
- [Section 97.1, "Configuring Common Data Source Login Options"](#)

### 96.1.2.1 DatabaseLogin

If you are creating a project that accesses a relational database, you must configure the project with a `DatabaseLogin`. Your choice of `DatabasePlatform` further customizes your project for a particular type of database (see [Section 96.1.3.1, "Database Platforms"](#)).

For more information, see [Section 98.1, "Introduction to Database Login Configuration"](#).



### 96.1.2.2 EISLogin

If you are creating a project that accesses a nonrelational data source using a JCA adapter, you must configure the project with an `EISLogin`. Your choice of `EISPlatform` further customizes your project for a particular JCA adapter and specifies what record type TopLink uses to exchange data with the EIS (see [Section 96.1.3.2, "EIS Platforms"](#)).

For more information, see [Section 99.1, "Introduction to EIS Login Configuration"](#).

## 96.1.3 Data Source Platform Types

TopLink abstracts the details of your underlying data source using data source platform classes. TopLink provides the following data source platforms:

- [Database Platforms](#)
- [EIS Platforms](#)

A data source platform is owned by your project's `Login`. For more information about logins, see [Section 96.1.2, "Data Source Login Types"](#).

To configure most platform options, you must use an amendment method (see [Section 119.35, "Configuring Amendment Methods"](#)), or a `preLogin` event listener (see [Section 87.2.5, "Managing Session Events with the Session Event Manager"](#)).

### 96.1.3.1 Database Platforms

TopLink interacts with databases using structured query language (SQL). Because each database platform uses its own variation on the basic SQL language, TopLink must adjust the SQL it uses to communicate with the database to ensure that the application runs smoothly.

The type of database platform you choose determines the specific means by which the TopLink runtime accesses the database, including the type of Java Database Connectivity (JDBC) driver to use. JDBC is an application programming interface (API) that gives Java applications access to a database. TopLink relational projects rely on JDBC connections to read objects from, and write objects to, the database. TopLink applications use either individual JDBC connections or a JDBC connection pool (see [Section 96.1.6, "Connection Pools"](#)), depending on the application architecture.

TopLink provides a variety of database-specific platforms that let you customize your project for your target database.

Oracle Database platforms are located in `oracle.toplink.platform.database.oracle` package and include the following:

- `Oracle8Platform`
- `Oracle9Platform`
- `Oracle10Platform`
- `Oracle11Platform`

Non-Oracle Database platforms are located in `oracle.toplink.platform.database` package and include the following:

- `AccessPlatform` for Microsoft Access databases
- `AttunityPlatform` for Attunity Connect JDBC drivers
- `CloudscapePlatform`

- `DB2MainframePlatform`
- `DB2Platform`
- `DBasePlatform`
- `DerbyPlatform`
- `HSQLPlatform`
- `InformixPlatform`
- `JavaDBPlatform`
- `MySQL4Platform`
- `PointBasePlatform`
- `PostgreSQLPlatform`
- `SQLAnywherePlatform`
- `SQLServerPlatform`
- `SybasePlatform`
- `TimesTen7Platform` for TimesTen 7 database

Specify your database platform at the project level (see [Section 20.2, "Configuring Relational Database Platform at the Project Level"](#)) for all sessions, or override this project-level configuration at the session level (see [Section 98.2, "Configuring a Relational Database Platform at the Session Level"](#)).

If you set your database platform in TopLink Workbench, then TopLink Workbench manages the database platform configuration for you automatically.

### 96.1.3.2 EIS Platforms

TopLink interacts with an EIS data source indirectly by way of a JCA adapter. TopLink abstracts the details of an EIS data source using the `oracle.toplink.eis.EISPlatform` class.

The type of EIS platform you choose determines the specific means by which the TopLink runtime accesses the EIS, including the type of JCA adapter to use. TopLink EIS projects rely on EIS connections to read objects from, and write objects to, the EIS. TopLink applications use individual EIS connections returned by the EIS connection factory specified by the EIS platform.

TopLink provides a variety of `EISPlatform` classes that let you customize your project for your target EIS.

EIS platforms for production are located in `oracle.toplink.eis.adapters` package and include the following:

- `oracle.toplink.eis.adapters.aq.AQPlatform` to access an EIS using Oracle Advanced Queuing messages.
- `oracle.toplink.eis.adapters.attunity.AttunityPlatform` to access an EIS using an Attunity JCA adapter.
- `oracle.toplink.eis.adapters.jms.JMSPlatform` to access an EIS using JMS messages.
- `oracle.toplink.eis.adapters.mqseries.MQPlatform` to access an EIS using IBM MQSeries messages.

EIS platforms for testing are also located in `oracle.toplink.eis.adapters` and include the following:

- `oracle.toplink.eis.adapters.blackbox.BlackBoxPlatform` for testing your EIS project with the Sun BlackBox reference adapter using indexed records only.
- `oracle.toplink.eis.adapters.xmlfile.XMLFilePlatform` for testing your EIS project with an EIS emulated as one or more XML files in the local file system using XML records.

Specify your EIS platform at the session level (see [Section 99.2, "Configuring an EIS Data Source Platform at the Session Level"](#)).

If you set your platform in TopLink Workbench, then TopLink Workbench manages the EIS platform configuration for you automatically.

## 96.1.4 Authentication

**Authentication** is the means by which a data source validates a user's identity and determines whether or not the user has sufficient privileges to perform a given action.

For two-tier applications, simple JDBC authentication is usually sufficient (see [Section 96.1.4.1, "Simple JDBC Authentication"](#)).

For three-tier applications, you can use simple JDBC authentication or, proxy authentication (see [Section 96.1.4.2, "Oracle Database Proxy Authentication"](#)) when using the Oracle Call Interface (OCI) JDBC driver.

Authentication plays a central role in data security and user accountability and auditing (see [Section 96.1.4.3, "Auditing"](#)).

### 96.1.4.1 Simple JDBC Authentication

When you configure a TopLink database login with a user name and password ([Section 97.2, "Configuring User Name and Password"](#)), TopLink provides these credentials to the JDBC driver that you configure your application to use (see [Section 98.3, "Configuring Database Login Connection Options"](#)).

By default, TopLink writes passwords to and reads them from the `sessions.xml` file in encrypted form using JCE encryption. Optionally, you can configure a different encryption class (see [Section 97.3, "Configuring Password Encryption"](#)).

### 96.1.4.2 Oracle Database Proxy Authentication

TopLink supports proxy authentication with Oracle Database in JSE applications and JEE applications using OC4J native or managed data sources with Oracle JDBC driver release 10.1.0.2.0 or later and external connection pools (see [Section 96.1.6.2, "External Connection Pools"](#)) only.

---

**Note:** TopLink does not support Oracle Database proxy authentication with JTA.

---

Oracle Database proxy authentication delivers the following security benefits:

- A limited trust model, by controlling the users on whose behalf middle tiers can connect, and the roles the middle tiers can assume for the user.
- Scalability, by supporting user sessions through Oracle Call Interface (OCI) and thick JDBC, and eliminating the overhead of reauthenticating clients.

- Accountability, by preserving the identity of the real user through to the database, and enabling auditing of actions taken on behalf of the real user.
- Flexibility, by supporting environments in which users are known to the database, and in which users are merely "application users" of which the database has no awareness.

---



---

**Note:** Oracle Database supports proxy authentication in three-tiers only; it does not support it across multiple middle tiers.

---



---

For more information about authentication in Oracle Database, see "Preserving User Identity in Multitiered Environments" in the *Oracle Database Security Guide*.

Configure your TopLink database login to use proxy authentication (see [Section 98.8, "Configuring Oracle Database Proxy Authentication"](#)) to do the following:

- address the complexities of authentication in a three-tier architecture (such as client-to-middle-tier and middle-tier-to-database authentication, and client reauthentication through the middle -tier to the database)
- enhance database audit information (for even triggers and stored procedures) by using a specific user for database operations, rather than the generic pool user
- simplify VPD/OLS configuration (see [Section 87.5.1, "Isolated Client Sessions and Oracle Virtual Private Database \(VPD\)"](#)) by using a proxy user, rather than setting user information directly in the session context with stored procedures

### 96.1.4.3 Auditing

Regardless of what type of authentication you choose, TopLink logs the name of the user associated with all database operations. [Example 96–1](#) shows the CONFIG level TopLink logs when a `ServerSession` connects through the main connection for the sample user "scott", and a `ClientSession` uses proxy connection "jeff".

#### **Example 96–1 TopLink Logs with Oracle Database Proxy Authentication**

```
[TopLink
Config]--ServerSession(13)--Connection(14)--Thread(Thread[main,5,main])--connecting(DatabaseL
ogin(platform=>Oracle9Platform user name=> "scott" connector=>OracleJDBC10_1_0_
2ProxyConnector datasource name=>DS))
[TopLink Config]--ServerSession(13)--Connection(34)--Thread(Thread[main,5,main])--Connected:
jdbc:oracle:thin:@localhost:1521:orcl
User: SCOTT
[TopLink
Config]--ClientSession(53)--Connection(54)--Thread(Thread[main,5,main])--connecting(DatabaseL
ogin(platform=>Oracle9Platform user name=> "scott" connector=>OracleJDBC10_1_0_
2ProxyConnector datasource name=>DS))
[TopLink Config]--ClientSession(53)--Connection(56)--Thread(Thread[main,5,main])--Connected:
jdbc:oracle:thin:@localhost:1521:orcl
User: jeff
```

For more information on configuring TopLink log level and log options, see [Section 89.4, "Configuring Logging"](#).

Your database server likely provides additional user auditing options. Consult your database server documentation for details.

Alternatively, you may consider using the TopLink unit of work in conjunction with your database schema for auditing purposes (see [Section 115.12, "Implementing User and Date Auditing with the Unit of Work"](#)).

## 96.1.5 Connections

A connection is an object that provides access to a data source by way of the driver you configure your application to use (see [Section 98.3, "Configuring Database Login Connection Options"](#)). Relational projects use JDBC to connect to the data source; EIS and XML projects use JCA. TopLink uses the interface `oracle.toplink.internal.databaseaccess.Accessor` to wrap data source connections. This interface is accessible from certain events (see [Section 16.2.8, "Descriptor Event Manager"](#)).

Typically, when using a server session, TopLink uses a different connection for both reading and writing. This lets you use nontransactional connections for reading and avoid maintaining connections when not required. See [Section 115.15.1.4, "Reading Through the Write Connection"](#) and [Exclusive Write Connections](#) for more information.

By default, a TopLink server session acquires connections lazily: that is, only during the commit operation of a unit of work. Alternatively, you can configure TopLink to acquire a write connections at the time you acquire a client sessions (see [Lazy Connection Acquisition](#)).

Connections can be allocated from internal or external connection pools (see [Section 96.1.6, "Connection Pools"](#)).

## 96.1.6 Connection Pools

A **connection pool** is a service that creates and maintains a shared collection (pool) of data source connections on behalf of one or more clients. The connection pool provides a connection to a process on request, and returns the connection to the pool when the process is finished using it. When it is returned to the pool, the connection is available for other processes. Because establishing a connection to a data source can be time-consuming, reusing such connections in a connection pool can improve performance.

TopLink uses connection pools to manage and share the connections used by server and client sessions. This feature reduces the number of connections required and allows your application to support many clients.

You can configure your session to use internal connection pools provided by TopLink or external connection pools provided by a JDBC driver or Java EE container.

You can use connection pools in your TopLink application for a variety of purposes, such as reading, writing, sequencing, and other application-specific functions.

This section describes the following:

- [Internal Connection Pools](#)
- [External Connection Pools](#)
- [Default \(Write\) and Read Connection Pools](#)
- [Sequence Connection Pools](#)
- [Application-Specific Connection Pools](#)

### 96.1.6.1 Internal Connection Pools

For non-Java EE applications, you typically use *internal* connection pools. By default, TopLink sessions use internal connection pools.

Using internal connection pools, you can use TopLink Workbench to configure the default (write) and read connection pools (see [Section 96.1.6.3, "Default \(Write\) and](#)

Read Connection Pools") and you can create additional connection pools for object identity (see [Section 96.1.6.4, "Sequence Connection Pools"](#)), or any other purpose (see [Section 96.1.6.5, "Application-Specific Connection Pools"](#)).

Using internal connection pools, you can optimize the creation of read connections for applications that read data only to display it and only infrequently modify data (see [Section 101.4, "Configuring a Nontransactional Read Login"](#)).

For information on selecting the type of connection pool to use, see [Section 97.4, "Configuring External Connection Pooling"](#).

For more information on creating and configuring internal connection pools, see the following:

- [Section 100.1, "Introduction to the Internal Connection Pool Creation"](#)
- [Section 101.1, "Introduction to the Internal Connection Pool Configuration"](#)

### 96.1.6.2 External Connection Pools

For Java EE applications, you typically use *external* connection pools.

If you are using an external transaction controller (JTA), you must use external connection pools to integrate with the JTA (see [Section 115.13, "Integrating the Unit of Work with an External Transaction Service"](#)).

Using external connection pools, you can use either TopLink Workbench or Java to configure the default (write) and read connection pools (see [Section 96.1.6.3, "Default \(Write\) and Read Connection Pools"](#)) and create additional connection pools for object identity (see [Section 96.1.6.4, "Sequence Connection Pools"](#)), or any other purpose (see [Section 96.1.6.5, "Application-Specific Connection Pools"](#)).

For more information on selecting the type of connection pool to use, see [Section 97.4, "Configuring External Connection Pooling"](#).

### 96.1.6.3 Default (Write) and Read Connection Pools

A server session provides a read connection pool and a write connection pool. These could be different pools, or if you use external connection pooling, the same connection pool.

All read queries use connections from the read connection pool and all queries that write changes to the data source use connections from the write connection pool. You can configure attributes of the default read and write connection pools.

Whenever a new connection is established, TopLink uses the connection configuration you specify in your session's `DatasourceLogin`. Alternatively, when you use an external transaction controller, you can define a separate connection configuration for a read connection pool to avoid the additional overhead, if appropriate (see [Section 101.4, "Configuring a Nontransactional Read Login"](#)).

For more information on configuring read and write connection pools, see [Section 101.1, "Introduction to the Internal Connection Pool Configuration"](#).

### 96.1.6.4 Sequence Connection Pools

An essential part of maintaining object identity (see [Section 102.2.1, "Cache Type and Object Identity"](#)) is sequencing—managing the assignment of unique values to distinguish one instance from another. For more information, see [Section 15.2.6, "Projects and Sequencing"](#).

Sequencing involves reading and writing a special sequence resource maintained by your data source.

By default, TopLink includes sequence operations in a separate transaction. This avoids complications during the write transaction, which may lead to deadlocks over the sequence resource. However, when using an external transaction controller (such as a JTA data source or connection pool), TopLink cannot use a different transaction for sequencing. Use a sequence connection pool to configure a non-JTA transaction pool for sequencing. This is required only for table sequencing—not native sequencing.

In each server session, you can create one connection pool, called a sequence connection pool, that TopLink uses exclusively for sequencing. With a sequence connection pool, TopLink satisfies a request for a new object identifier outside of the transaction from which the request originates. This allows TopLink to immediately commit an update to the sequence resource, which avoids deadlocks.

---

---

**Note:** If you use a sequence connection pool and the original transaction fails, the sequence operation does not roll back.

---

---

You should use a sequence connection pool, if the following applies:

- You use table sequencing (that is, non-native sequencing). See [Section 18.2.2.1, "Table Sequencing"](#) and [Section 18.2.2.2, "Unary Table Sequencing"](#) for more information.
- You use external transaction controller (JTA).

You should not use a sequence connection pool, if the following applies:

- You do not use sequencing, or use the data source's native sequencing (see [Section 18.2.2.5, "Native Sequencing with an Oracle Database Platform"](#) and [Section 18.2.2.6, "Native Sequencing with a Non-Oracle Database Platform"](#)).
- You have configured the sequence table to avoid deadlocks.
- You use non-JTA data sources.

For more information, see the following:

- [Section 100.1, "Introduction to the Internal Connection Pool Creation"](#)
- [Section 101.1, "Introduction to the Internal Connection Pool Configuration"](#)

#### 96.1.6.5 Application-Specific Connection Pools

When you use internal TopLink connection pools in a session, you can create one or more connection pools that you can use for any application purpose. These are called named connection pools, as you can give them any name you want and use them for any purpose.

Typically, use these named connection pools to provide pools of different security levels. For example, the "default" connection pool may only allow access to specific tables but the "admin" connection pool may allow access to all tables.

For more information, see the following:

- [Section 100.1, "Introduction to the Internal Connection Pool Creation"](#)
- [Section 101.1, "Introduction to the Internal Connection Pool Configuration"](#)
- [Section 90.4.4, "How to Acquire a Client Session that Uses a Named Connection Pool"](#)

## 96.2 Data Access API

This section describes the following:

- [Login Inheritance Hierarchy](#)
- [Platform Inheritance Hierarchy](#)

### 96.2.1 Login Inheritance Hierarchy

[Example 96–2](#) illustrates the login types that are derived from abstract class `oracle.toplink.sessions.DatasourceLogin`.

**Example 96–2 Login Inheritance Hierarchy**

```
class oracle.toplink.sessions.DatasourceLogin
  class oracle.toplink.sessions.DatabaseLogin
  class oracle.toplink.eis.EISLogin
```

### 96.2.2 Platform Inheritance Hierarchy

[Example 96–3](#) illustrates the platform type class hierarchy.

**Example 96–3 Platform Inheritance Hierarchy**

```
oracle.toplink.platform.database
  AccessPlatform
  AttunityPlatform
  CloudscapePlatform
  DatabasePlatform
  DB2MainframePlatform
  DB2Platform
  DBasePlatform
  DerbyPlatform
  HSQLPlatform
  InformixPlatform
  JavaDBPlatform
  PointBasePlatform
  PostgreSQLPlatform
  SQLAnywherePlatform
  SQLServerPlatform
  SybasePlatform
  TimesTen7Platform
oracle.toplink.platform.database.oracle
  Oracle8Platform
  Oracle9Platform
  Oracle10Platform
  Oracle11Platform
  OraclePlatform
```



---



---

## Configuring a Data Source Login

This chapter describes how to configure TopLink data source logins.

This chapter includes the following sections:

- [Configuring Common Data Source Login Options](#)
- [Configuring User Name and Password](#)
- [Configuring Password Encryption](#)
- [Configuring External Connection Pooling](#)
- [Configuring Properties](#)
- [Configuring a Default Null Value at the Login Level](#)

Table 97–1 lists the types of TopLink data source logins that you can configure and provides a cross-reference to the type-specific chapter that lists the configurable options supported by that type.

**Table 97–1** *Configuring TopLink Data Source Logins*

If you are configuring a...	See...
DatabaseLogin	Chapter 98, "Configuring a Database Login"
EISLogin	Chapter 99, "Configuring an EIS Login"

Table 97–2 lists the configurable options shared by two or more TopLink data source login types.

When using the `sessions.xml` file to configure login information, TopLink will override any login information in the `project.xml` and instead use the information from the `sessions.xml` configuration.

For more information, see the following:

- [Chapter 96, "Introduction to Data Access"](#)
- [Section 20.4, "Configuring Login Information at the Project Level"](#)

### 97.1 Configuring Common Data Source Login Options

Table 97–2 lists the configurable options shared by two or more TopLink data source login types. In addition to the configurable options described here, you must also configure the options described for the specific data source login types (see [Section 96.1.2, "Data Source Login Types"](#)), as shown in [Table 97–1](#)

**Table 97–2 Common Data Source Login Options**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
User name and password (see <a href="#">Section 97.2, "Configuring User Name and Password"</a> )	✓	✓	✓
Password encryption (see <a href="#">Section 97.3, "Configuring Password Encryption"</a> )			✓
External connection pooling (see <a href="#">Section 97.4, "Configuring External Connection Pooling"</a> )	✓	✓	✓
Properties (see <a href="#">Section 97.5, "Configuring Properties"</a> )	✓	✓	✓
Default null value (see <a href="#">Section 97.6, "Configuring a Default Null Value at the Login Level"</a> )			✓

## 97.2 Configuring User Name and Password

Optionally, you can specify the user name and password of a login.

Oracle recommends that you do not save the password with a deployment login.

If you specify a password, using a TopLink tool or Java, enter the plain text (not encrypted) value. TopLink will encrypt the password using JCE encryption.

By default, TopLink writes passwords to and read passwords from the `sessions.xml` file in encrypted form using JCE encryption.

By default, TopLink does not write passwords to and read passwords from the `project.xml` file unless you configure your project-level data source login accordingly. When you configure TopLink to write passwords and read passwords from the `project.xml` file, by default, it does so in encrypted form using JCE encryption.

For more information, see the following:

- [Section 20.4, "Configuring Login Information at the Project Level"](#)
- [Section 97.3, "Configuring Password Encryption"](#)

### 97.2.1 How to Configure User Name and Password Using TopLink Workbench

To specify a user name and password, use this procedure:

1. Select a server or database session in the **Navigator**. Its properties appear in the Editor.
2. Click the **Login** tab. The Login tab appears.
3. Click the **Connection** subtab. The Connection subtab appears.

**Figure 97–1 Login Tab, Connection Subtab, User Name and Password Fields**

The screenshot shows the 'Login' tab of the configuration dialog. The 'Connection' subtab is selected. The 'User Name (Optional):' and 'Password (Optional):' fields are highlighted with red circles. The 'URL:' field contains 'jdbc:oracle:thin:@<HOST>:<PORT>:<SID>'. The 'Database Platform' is set to 'Oracle10g', 'Database Driver' is 'Driver Manager', and 'Driver Class' is 'oracle.jdbc.driver.OracleDriver'. The 'External Connection Pooling' checkbox is checked.

4. Enter a user name and password in plain text (not encrypted).  
Enter a user name and password in plain text (not encrypted).

**See Also**

[Configuring User Name and Password](#)

**See Also**

[Configuring User Name and Password](#)

## 97.3 Configuring Password Encryption

By default, TopLink writes passwords to and read passwords from the `sessions.xml` file in encrypted form using JCE encryption.

By default, TopLink does not write passwords to and read passwords from the `project.xml` file unless you configure your project-level data source login accordingly. When you configure TopLink to write passwords and read passwords from the `project.xml` file, by default, it does so in encrypted form using JCE encryption.

You can implement your own encryption class and configure your session `DatasourceLogin` to use it.

Currently, neither Oracle JDeveloper nor TopLink Workbench support specifying the encryption class used. To change the encryption class used, you must modify the login in Java.

For more information, see the following:

- [Section 20.4, "Configuring Login Information at the Project Level"](#)
- [Section 97.2, "Configuring User Name and Password"](#)

### 97.3.1 How to Configure Password Encryption Using Java

To configure a password encryption class, follow this procedure:

1. Create your encryption class.

Your encryption class must implement the `oracle.toplink.internal.security.Securable` interface, as [Example 97–1](#) shows.

**Example 97–1 Custom Encryption Class Implementing Securable**

```
import oracle.toplink.internal.security.Securable;

public class MyEncryptor implements Securable {

    public String encryptPassword(String pswd) {
        ...
    }

    public String decryptPassword(String encryptedPswd) {
        ...
    }

}
```

2. Create a session event listener class for the `preLogin` event that calls `DatasourceLogin` method `setEncryptionClassName` to configure your session with your encryption class.

Use the `SessionEventAdapter` to simplify your session event listener, as [Example 97–2](#) shows.

**Example 97–2 Specifying a Custom Encryption Class in a Session Event Listener**

```
import oracle.toplink.tools.sessionconfiguration.SessionEventAdapter;
import oracle.toplink.sessions.SessionEvent;
import oracle.toplink.sessions.Session;
import oracle.toplink.sessions.DatasourceLogin;

public class MySessionEventListener extends SessionEventAdapter {

    public void preLogin(SessionEvent event) {
        Session session = event.getSession();
        DatasourceLogin login = session.getDatasourceLogin();
        login.setEncryptionClassName(MyEncryptor.class.getName());
    }

}
```

3. Associate your session event listener class with your session.

For more information, see [Section 89.10, "Configuring Session Event Listeners"](#).

## 97.4 Configuring External Connection Pooling

For non-Java EE applications, you typically use internal connection pools provided by TopLink (see [Section 96.1.6.1, "Internal Connection Pools"](#)). In this case, you can use Oracle JDeveloper or TopLink Workbench to configure connection pool options and to create a sequence connection pool and application-specific (named) connection pools.

For Java EE applications, you typically use external connection pools provided by a JDBC driver or Java EE container (see [Section 96.1.6.2, "External Connection Pools"](#)). Optionally, you can configure a read connection pool to use a nontransactional login, and you can configure a sequence connection pool to use a separate (preferably nontransactional) login of its own.

Because JTA external transaction controllers are dependent upon the external transaction service that the application server provides, you must configure TopLink to use external connection pools if you are using an external transaction controller (see [Section 115.13, "Integrating the Unit of Work with an External Transaction Service"](#)).

External connection pools enable your TopLink application to do the following:

- Integrate into a Java EE-enabled system.

- Integrate with JTA transactions (JTA transactions require a JTA-enabled data source).
- Leverage a shared connection pool in which multiple applications use the same data source.
- Use a data source configured and managed directly on the server.

For more information about connection pools, see [Section 96.1.6, "Connection Pools"](#).

### 97.4.1 How to Configure External Connection Pooling Using TopLink Workbench

To specify if the session login uses external connection pooling, use this procedure:

1. Configure a data source on the application server.

If you are using the external connection pool with an external transaction controller (see [Section 89.9, "Configuring the Server Platform"](#)), be sure to configure a JTA-enabled data source.

For more information, see your Java EE container documentation.

2. Select a server or database session in the **Navigator**. Its properties appear in the Editor.
3. Click the **Login** tab. The Login tab appears.
4. Click the **Connection** subtab. The Connection subtab appears.

**Figure 97–2 Login Tab, Connection Subtab, External Connection Pooling Field, Database Driver**

The screenshot shows the 'Login' tab with the 'Connection' subtab selected. The 'Database Platform' is set to 'Oracle10g' and the 'Database Driver' is 'Driver Manager'. The 'Driver Class' is 'oracle.jdbc.driver.OracleDriver'. The 'URL' is 'jdbc:oracle:thin:@<HOST>:<PORT>:<SID>'. The 'User Name (Optional)' and 'Password (Optional)' fields are empty. The 'External Connection Pooling' checkbox is checked and circled in red.

**Figure 97–3 Connection Tab, External Connection Pooling Field, Java EE Data Source**

The screenshot shows the 'Login' tab with the 'Connection' subtab selected. The 'Database Platform' is set to 'Oracle10g' and the 'Database Driver' is 'J2EE Data Source'. The 'Data Source Name' is 'jdbc/OracleDS'. The 'User Name (Optional)' and 'Password (Optional)' fields are empty. The 'External Connection Pooling' checkbox is checked and circled in red.

Specify if this login uses External Connection Pooling. For a database driver, external connection pooling is optional. For a Java EE data source, external connection pooling is mandatory.

## 97.4.2 How to Configure External Connection Pooling Using Java

To configure the use of an external connection pool in Java, do the following:

1. Configure the data source on the application server.  
If you are using the external connection pool with an external transaction controller (see [Section 89.9, "Configuring the Server Platform"](#)), be sure to configure a JTA-enabled data source.  
For more information, see your Java EE container documentation.
2. Configure the `DatasourceLogin` to specify the data source and to use an external connection pool by using the `useExternalConnectionPooling` method.

## 97.5 Configuring Properties

For all `DatasourceLogin` types, you can specify custom named values, called properties. Some data sources require additional, driver-specific properties not supported in the `DatasourceLogin` API (for example, see [Section 12.11.1, "How to Optimize JDBC Driver Properties"](#)). Add these properties to the `DatasourceLogin` so that TopLink can pass them to the driver.

For relational sessions, you must first enable advanced option **Use Properties** (see [Section 98.7, "Configuring Advanced Options"](#)).

For EIS sessions, properties are always enabled.

---

---

**Note:** Do not set a password as a property. Always use Oracle JDeveloper, TopLink Workbench, or `DatabaseLogin` method `setPassword`. For more information on configuring a password, see [Section 97.2, "Configuring User Name and Password"](#).

---

---

When using TopLink Workbench, you can only set character values, which TopLink returns as `String` objects (see [Section 97.5.1, "How to Configure Properties Using TopLink Workbench"](#)).

When using Java, you can set any `Object` value (see [Section 97.5.2, "How to Configure Properties Using Java"](#)).

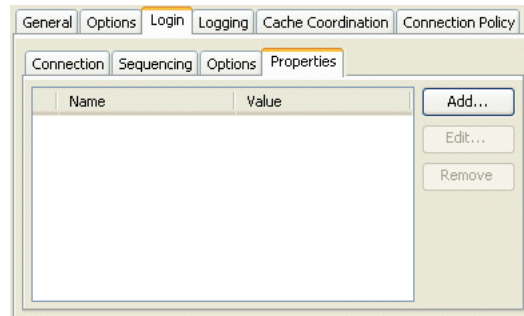
### 97.5.1 How to Configure Properties Using TopLink Workbench

To specify arbitrary named value pairs that TopLink associates with a `DatasourceLogin`, use this procedure:

1. Select a server or database session in the **Navigator**. Its properties appear in the Editor.
2. Click the **Login** tab. The Login tab appears.
3. If necessary, enable support for properties:
  - for relational sessions, you must first enable advanced option **Use Properties** (see [Section 98.7, "Configuring Advanced Options"](#));

- for EIS sessions, properties are always enabled.
4. Click the **Properties** subtab. The Properties subtab appears.

**Figure 97–4 Login Tab, Properties Subtab**



To add (or change) a new **Name/Value** property, click **Add** (or **Edit**). Add Property dialog box appears.

Use the following information to add or edit a login property on the Add Property dialog box:

Option	Description
<b>Name</b>	The name by which TopLink retrieves the property value using the <code>DatasourceLogin</code> method <code>getProperty</code> .
<b>Value</b>	The value TopLink retrieves using the <code>DatasourceLogin</code> method <code>getProperty</code> passing in the corresponding property name.  Using TopLink Workbench, you can set only character values that TopLink returns as <code>String</code> objects.

To delete an existing property, select the **Name/Value** row and click **Remove**.

### 97.5.2 How to Configure Properties Using Java

Using Java, you can set any `Object` value using `DatasourceLogin` method `setProperty`. To remove a property, use `DatasourceLogin` method `removeProperty`.

## 97.6 Configuring a Default Null Value at the Login Level

A default null value is the Java `Object` type and value that TopLink uses instead of `null` when TopLink reads a `null` value from a data source.

When you configure a default null value at the login level, it applies to all mappings used in a session. In this case, TopLink uses it to translate in one direction only: when TopLink reads `null` from the data source, it converts this `null` to the specified type and value.

You can also use TopLink to set a default null value on a per-mapping basis (see [Section 121.5, "Configuring a Default Null Value at the Mapping Level"](#)).

---

---

**Note:** A default null value must be an `Object`. To specify a primitive value (such as `int`), you must use the corresponding `Object` wrapper (such as `Integer`).

---

---

### 97.6.1 How to Configure a Default Null Value at the Login Level Using Java

Using Java API, you can configure a default null value for all mappings used in a session with the `DatabaseLogin` method `setDefaultNullValue(Class, Object)`.

For example:

```
// Defaults all null String values read from the database to empty String
session.getLogin().setDefaultNullValue(String.class, "");
```



## Configuring a Database Login

In a relational database project, TopLink retrieves the table information from the database, for each descriptor. Each TopLink Workbench project contains an associated database. You can create multiple logins for each database.

This chapter includes the following sections:

- [Introduction to Database Login Configuration](#)
- [Configuring a Relational Database Platform at the Session Level](#)
- [Configuring Database Login Connection Options](#)
- [Configuring Sequencing at the Session Level](#)
- [Configuring a Table Qualifier](#)
- [Configuring JDBC Options](#)
- [Configuring Advanced Options](#)
- [Configuring Oracle Database Proxy Authentication](#)

### 98.1 Introduction to Database Login Configuration

Table 98–1 lists the configurable options for a database login.

**Table 98–1** Configurable Options for Database Login

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Relational database (see Section 98.2, "Configuring a Relational Database Platform at the Session Level")	✓	✓	✓
Database login connection options (see Section 98.3, "Configuring Database Login Connection Options")	✓	✓	✓
Sequencing (see Section 98.4, "Configuring Sequencing at the Session Level")	✓	✓	✓
JDBC options (see Section 98.6, "Configuring JDBC Options")	✓	✓	✓
User name and password (see Section 97.2, "Configuring User Name and Password")	✓	✓	✓
Table qualifier (see Section 98.5, "Configuring a Table Qualifier")	✓	✓	✓
Advanced options (see Section 98.7, "Configuring Advanced Options")	✓	✓	✓
Password encryption (see Section 97.3, "Configuring Password Encryption")			✓
External connection pooling (see Section 97.4, "Configuring External Connection Pooling")	✓	✓	✓

**Table 98–1 (Cont.) Configurable Options for Database Login**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Properties (see Section 97.5, "Configuring Properties")	✓	✓	✓
Oracle Database proxy authentication (see Section 98.8, "Configuring Oracle Database Proxy Authentication")			✓

## 98.2 Configuring a Relational Database Platform at the Session Level

For each database session, you must specify the database platform (such as Oracle Database). This platform configuration overrides the platform at the project level, if configured.

For more information, see the following:

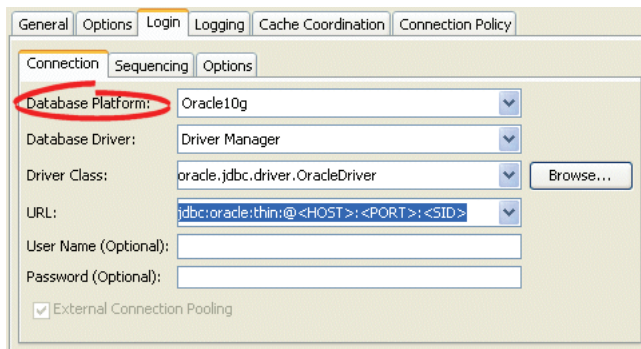
- Section 20.2, "Configuring Relational Database Platform at the Project Level"
- Section 96.1.3, "Data Source Platform Types"

### 98.2.1 How to Configure a Relational Database Platform at the Session Level Using TopLink Workbench

To specify the database platform options for a relational server (or database) session login, use this procedure:

1. Select a relational server (or database) session in the **Navigator**. Its properties appear in the Editor.
2. Click the **Login** tab. The Login tab appears.
3. Click the **Connection** subtab. The Connection subtab appears.

**Figure 98–1 Login Tab, Connection Subtab, Database Platform Option**



Select the database platform from the menu of options. This menu includes all instances of DatabasePlatform in the TopLink classpath.

## 98.3 Configuring Database Login Connection Options

You configure connection information at the session level for a POJO TopLink application. This information is stored in the `sessions.xml` file. The TopLink runtime uses this information whenever you perform a persistence operation using the session in your POJO TopLink application.

This connection configuration overrides the connection information at the project level, if configured. For more information about project-level configuration, see [Section 20.5, "Configuring Development and Deployment Logins"](#).

This connection configuration is overridden by the connection information at the connection pool level. For more information, see [Section 101.5, "Configuring Connection Pool Connection Options"](#).

### 98.3.1 How to Configure Database Login Connection Options Using TopLink Workbench

To specify the connection options for a relational server (or database) session login, use this procedure:

1. Select a relational server (or database) session in the **Navigator**. Its properties appear in the Editor.
2. Click the **Login** tab. The Login tab appears.
3. Click the **Connection** subtab. The Connection subtab appears.

**Figure 98–2 Login Tab, Connection Subtab, Database Driver**

The screenshot shows the 'Login' tab with the 'Connection' subtab selected. The 'Database Platform' is set to 'Oracle10g'. The 'Database Driver' is 'Driver Manager', the 'Driver Class' is 'oracle.jdbc.driver.OracleDriver', and the 'URL' is 'jdbc:oracle:thin:@<HOST>:<PORT>:<SID>'. The 'User Name (Optional)' and 'Password (Optional)' fields are empty. The 'External Connection Pooling' checkbox is checked.

**Figure 98–3 Login Tab, Connection Subtab**

The screenshot shows the 'Login' tab with the 'Connection' subtab selected. The 'Database Platform' is set to 'Oracle10g'. The 'Database Driver' is 'J2EE Data Source', the 'Lookup Type' is 'Composite Name', and the 'Data Source Name' field is empty. The 'User Name (Optional)' and 'Password (Optional)' fields are empty. The 'External Connection Pooling' checkbox is checked.

Use the following information to enter data in the driver fields on the tab:

Field	Description
Database Driver	<p>Specify the appropriate database driver:</p> <ul style="list-style-type: none"> <li>▪ <b>Driver Manager:</b> specify this option to configure the driver class and URL used to connect to the database. In this case, you must configure the <b>Driver Class</b> and <b>Driver URL</b> fields.</li> <li>▪ <b>J2EE Data Source:</b> specify this option to use a Java EE data source already configured on your target application server. In this case, you must configure the <b>Datasource Name</b> field.</li> </ul> <p>Note: If you select <b>J2EE Datasource</b>, you must use external connection pooling. You cannot use internal connection pools with this <b>Database Driver</b> option (for more information, see <a href="#">Section 97.4, "Configuring External Connection Pooling"</a>).</p>
Driver Class <sup>1</sup>	Configure this field when <b>Database Driver</b> is set to <b>Driver Manager</b> . Select from the menu of options. This menu includes all JDBC drivers in the TopLink classpath.
Driver URL <sup>1</sup>	Configure this field when <b>Database Driver</b> is set to <b>Driver Manager</b> . Select from the menu of options relevant to the selected <b>Driver Class</b> , and edit the URL to suit your data source.
Data Source Name <sup>2</sup>	<p>Configure this field when <b>Database Driver</b> is set to <b>J2EE Datasource</b>. Specify any valid JNDI name that identifies the Java EE data source preconfigured on your target application server (example: <code>jdbc/EmployeeDB</code>).</p> <p>By convention, all such names should resolve to the JDBC subcontext (relative to the standard <code>java:comp/env</code> naming context that is the root of all provided resource factories).</p>
Lookup Type <sup>2</sup>	<p>Configure this field when <b>Database Driver</b> is set to <b>J2EE Datasource</b>. Specify the lookup method for determining the JNDI name:</p> <ul style="list-style-type: none"> <li>▪ Composite Name</li> <li>▪ Compound Name</li> <li>▪ String</li> </ul>

<sup>1</sup> Applicable only when **Database Driver** is set to **Driver Manager**.

<sup>2</sup> Applicable only when **Database Driver** is set to **J2EE Datasource**.

## 98.4 Configuring Sequencing at the Session Level

You configure TopLink sequencing at the session or project level to tell TopLink how to obtain sequence values: that is, what type of sequences to use.

In a CMP project, you do not configure a session directly: in this case, you must configure sequences at the project level (see [Section 20.3, "Configuring Sequencing at the Project Level"](#)). In a POJO project, you can configure a session directly: in this case, you can use session-level sequence configuration to override project-level sequence configuration, on a session-by-session basis, if required.

You can configure sequencing at the session level using Oracle JDeveloper.

Using TopLink Workbench (see [Section 98.4.1, "How to Configure Sequencing at the Session Level Using TopLink Workbench"](#)), you can configure table sequencing (see [Section 18.2.2.1, "Table Sequencing"](#)) and native sequencing ([Section 18.2.2.5, "Native Sequencing with an Oracle Database Platform"](#) and [Section 18.2.2.6, "Native Sequencing with a Non-Oracle Database Platform"](#)), and you can configure a preallocation size that applies to all sequences (see [Section 18.2.3, "Sequencing and Preallocation Size"](#)).

Using Java (see [Section 98.4.2, "How to Configure Sequencing at the Session Level Using Java"](#)), you can configure any sequence type that TopLink supports ([Section 18.2.2, "Sequencing Types"](#)). You can create any number and combination of sequences. You can create a sequence object explicitly or use the default sequence that the platform creates. You can associate the same sequence with more than one descriptor and you can configure a separate preallocation size for each descriptor's sequence.

After configuring the sequence type at the session (or project) level, to enable sequencing, you must configure a descriptor with a sequence field and a sequence name (see [Section 23.3, "Configuring Sequencing at the Descriptor Level"](#)).

For more information about sequencing, see [Section 18.2, "Sequencing in Relational Projects"](#).

### 98.4.1 How to Configure Sequencing at the Session Level Using TopLink Workbench

To specify the sequencing information for a relational server (or database) session, use this procedure:

1. Select the session object in the **Navigator**.
2. Click the **Login** tab in the **Editor**.
3. Click the **Sequencing** subtab. The Sequencing subtab appears.

**Figure 98–4 Login Tab, Sequencing Subtab**

Use the following information to enter data in each field of the Sequencing subtab to configure the persistence type:

Field	Description
<b>Preallocation Size</b>	Select the default preallocation size (see <a href="#">Section 18.2.3, "Sequencing and Preallocation Size"</a> ). Default is 50. The preallocation size you configure applies to all sequences.
<b>Default Sequence Table</b>	Select this option to use table sequencing (see <a href="#">Section 18.2.2.1, "Table Sequencing"</a> ) with default sequence table name <code>SEQUENCE</code> , default sequence name field <code>SEQ_NAME</code> , and default sequence count field <code>SEQ_COUNT</code> .
<b>Native Sequencing</b>	Select this option to use a sequencing object (see <a href="#">Section 18.2.2.5, "Native Sequencing with an Oracle Database Platform"</a> or <a href="#">Section 18.2.2.6, "Native Sequencing with a Non-Oracle Database Platform"</a> ) created by the database platform. This option applies to supported database platforms (see <a href="#">Section 96.1.3.1, "Database Platforms"</a> ).

Field	Description
<b>Custom Sequence Table</b>	Select this option to use table sequencing (see <a href="#">Section 18.2.2.1, "Table Sequencing"</a> ) with a sequence table name, sequence name field, and sequence count field name that you specify.
<b>Name</b>	Select the name of the sequence table.
<b>Name Field</b>	Select the name of the column used to store the sequence name.
<b>Counter Field</b>	Select the name of the column used to store the sequence count.

## 98.4.2 How to Configure Sequencing at the Session Level Using Java

Using Java, you can perform the following sequence configurations:

- [Using the Platform Default Sequence](#)
- [Configuring Multiple Sequences](#)
- [Configuring Query Sequencing](#)

### 98.4.2.1 Using the Platform Default Sequence

After you configure your login with a platform (see [Section 98.2, "Configuring a Relational Database Platform at the Session Level"](#)), you can use the default sequence that the platform provides.

If you associate a descriptor with an unspecified sequence, the TopLink runtime will create an instance of `DefaultSequence` to provide sequencing for that descriptor. For more information, see [Section 23.3.2.3, "Configuring the Platform Default Sequence"](#).

You can access the default platform sequence directly as [Example 98–1](#) shows. For example, by default, a `DatabasePlatform` creates a table sequence using the default table and column names (see [Section 18.2.2.1, "Table Sequencing"](#)).

#### **Example 98–1 Accessing the Platform Default Sequence**

```
// assume that dbLogin owns a DatabasePlatform
TableSequence tableSeq2 = ((TableSequence)dbLogin.getDefaultSequence()).clone();
tableSeq2.setName("EMP_SEQ");
tableSeq2.setPreallocationSize(75);
dbLogin.addSequence(tableSeq2);
```

To avoid having to clone the platform default sequence, you can use the `DefaultSequence` class—a wrapper for the platform default sequence—as [Example 98–2](#) shows. The new sequence named `EMP_SEQ` will be of the same type as the platform default sequence.

#### **Example 98–2 Using the DefaultSequence Class**

```
login.addSequence(
    new DefaultSequence("EMP_SEQ", 75)
);
```

You can override the default platform sequence, as [Example 98–3](#) shows. In this example, `dbLogin` owns a `DatabasePlatform` that provides a default sequence of type `TableSequence`. After setting the default sequence to type `UnaryTableSequence`, when you use the `DefaultSequence` class, it will access the new default sequence type. In this example, the sequence named `EMP_SEQ` will be of type `UnaryTableSequence` and have a preallocation size of 75.

**Example 98–3 Overriding the Platform Default Sequence**

```
// assume that dbLogin owns a DatabasePlatform
Sequence unaryTableSequence = new UnaryTableSequence();
unaryTableSequence.setPreallocationSize(40);
dbLogin.setDefaultSequence(unaryTableSequence);
dbLogin.addSequence(
    new DefaultSequence("EMP_SEQ", 75) // UnaryTableSequence
);
```

**98.4.2.2 Configuring Multiple Sequences**

In addition to using the platform default sequence (see [Section 98.4.2.1, "Using the Platform Default Sequence"](#)), you can explicitly create sequence instances and configure a `Login` with any combination of sequence types, each with their own preallocation size, as [Example 98–4](#) shows. In this example, the sequence named `EMP_SEQ` will provide sequence values exclusively for instances of the `Employee` class and `ADD_SEQ` will provide sequence values exclusively for instances of the `Address` class. The sequence named `PHONE_SEQ` will use the platform default sequence with a preallocation size of 30 to provide sequence values for the `Phone` class.

**Example 98–4 Configuring Multiple Sequences Explicitly**

```
login.addSequence(new TableSequence("EMP_SEQ", 25));
login.addSequence(new DefaultSequence("PHONE_SEQ", 30));
login.addSequence(new UnaryTableSequence("ADD_SEQ", 55));
login.addSequence(new NativeSequence("NAT_SEQ", 10));
```

If `login` owned a `DatabasePlatform` (whose default sequence type is `TableSequence`), you could configure your sequences using the platform default sequence type, as [Example 98–5](#) shows. In this example, sequences `EMP_SEQ` and `PHONE_SEQ` share the same `TableSequence` table: `EMP_SEQ` and `PHONE_SEQ` represent rows in this table.

**Example 98–5 Configuring Multiple Sequences Using the Default Sequence Type**

```
login.addSequence(new DefaultSequence("EMP_SEQ", 25));
login.addSequence(new DefaultSequence("PHONE_SEQ", 30));
login.addSequence(new UnaryTableSequence("ADD_SEQ", 55));
login.addSequence(new NativeSequence("NAT_SEQ", 10));
```

**98.4.2.3 Configuring Query Sequencing**

You can configure the query that `TopLink` uses to update or read a sequence value for any sequence type that extends `QuerySequence`.

In most applications, the queries that `TopLink` automatically uses are sufficient. However, if your application has special sequencing needs—for example, if you want to use stored procedures for sequencing—then you can configure the update and read queries that the `TopLink` sequence uses.

[Example 98–7](#) illustrates how to specify a stored procedure that updates a sequence and returns the new sequence value with a single SQL select query. In this example, the stored procedure is named `UPDATE_SEQ`. It contains one input argument—the name of the sequence to update (`SEQ_NAME`), and one output argument—the value of the sequence after the update (`SEQ_COUNT`). The stored procedure increments the sequence value associated with the sequence named `SEQ_NAME` and returns the new sequence value in the output argument named `SEQ_COUNT`.

**Example 98–6 Using a Stored Procedure for both Sequence Update and Select**

```
ValueReadQuery seqReadQuery = new ValueReadQuery();
StoredProcedureCall spCall = new StoredProcedureCall();
spCall.setProcedureName("UPDATE_SEQ");
seqReadQuery.addNamedArgument("SEQ_NAME");
seqReadQuery.addNamedOutputArgument("SEQ_COUNT");
seqReadQuery.setCall(spCall);
((QuerySequence)login.getDefaultSequence()).setSelectQuery(seqReadQuery);
```

[Example 98–7](#) and [Example 98–8](#) illustrate how to specify separate stored procedures for sequence update and select actions.

In [Example 98–7](#), the stored procedure is named UPDATE\_SEQ and it contains one input argument: the name of the sequence to update (SEQ\_NAME). The stored procedure increments the sequence value associated with the sequence named SEQ\_NAME.

**Example 98–7 Using a Stored Procedure for Sequence Updates Only**

```
DataModifyQuery seqUpdateQuery = new DataModifyQuery();
StoredProcedureCall spCall = new StoredProcedureCall();
spCall.setProcedureName("UPDATE_SEQ");
seqUpdateQuery.addNamedArgument("SEQ_NAME");
seqUpdateQuery.setCall(spCall);
((QuerySequence)login.getDefaultSequence()).setUpdateQuery(seqUpdateQuery);
```

In [Example 98–8](#), the stored procedure is named SELECT\_SEQ and it takes one argument: the name of the sequence to select from (SEQ\_NAME). The stored procedure reads one data value: the current sequence value associated with the sequence name SEQ\_NAME.

**Example 98–8 Using a Stored Procedure for Sequence Selects Only**

```
ValueReadQuery seqReadQuery = new ValueReadQuery();
StoredProcedureCall spCall = new StoredProcedureCall();
spCall.setProcedureName("SELECT_SEQ");
seqReadQuery.addArgument("SEQ_NAME");
seqReadQuery.setCall(spCall);
login.((QuerySequence)getDefaultSequence()).setSelectQuery(seqReadQuery)
```

You can also create a QuerySequence directly and add it to your login, as [Example 98–9](#) shows.

**Example 98–9 Using the QuerySequence Class**

```
// Use the two-argument constructor: pass in sequence name and preallocation size.
// Alternatively, you can use zero- or one-argument (sequence name) constructor
login.addSequence(new QuerySequence("SEQ1", 75));
```

## 98.5 Configuring a Table Qualifier

Some databases (such as Oracle Database and DB2) require that all tables be qualified by an identifier. This can be the creator of the table or database name on which the table exists. When you specify a table qualifier, TopLink uses this qualifier for all of the tables it references. Specify a table qualifier only if required and only if all of the tables have the same qualifier.

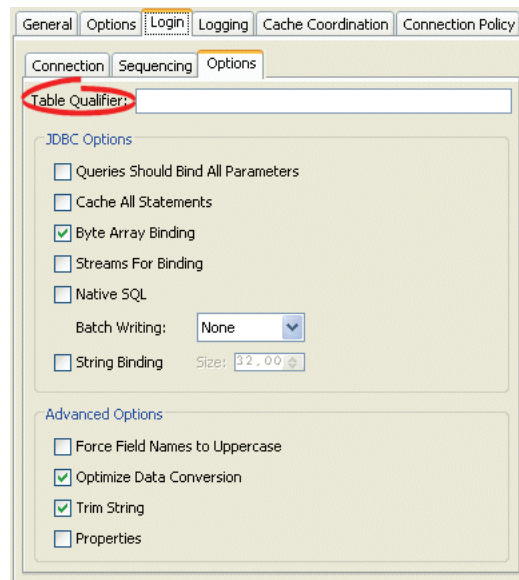
### 98.5.1 How to Configure a Table Qualifier Using TopLink Workbench

To specify a table qualifier, use this procedure:



1. Select a relational server (or database) session in the **Navigator**. Its properties appear in the Editor.
2. Click the **Login** tab. The Login tab appears.
3. Click the **Options** subtab. The Options subtab appears.

**Figure 98–5 Login Tab, Options Subtab, Table Qualifier Field**



In the **Table Qualifier** field enter the identifier used to qualify references to all tables in this database.

## 98.5.2 How to Configure a Table Qualifier Using Java

To set the default qualifier for all tables, use the `DatabaseLogin` method `setTableQualifier`.

## 98.6 Configuring JDBC Options

Most JDBC drivers support the run-time configuration of various options to customize driver operation to meet user needs. TopLink provides direct support (in API, Oracle JDeveloper and TopLink Workbench) for many of the most important options, as this section describes, as well as more advanced options (see [Section 98.7, "Configuring Advanced Options"](#)).

You can also configure additional options by specifying properties (see [Section 97.5, "Configuring Properties"](#)).

---

**Note:** Not all drivers support all JDBC options. Selecting a combination of options may result in different behavior from one driver to another. Before selecting JDBC options, consult your JDBC driver documentation.

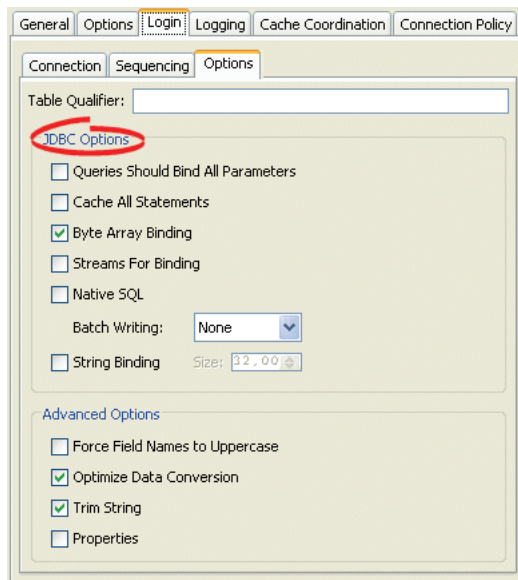
---

### 98.6.1 How to Configure JDBC Options Using TopLink Workbench

To specify the JDBC options for a relational server (or database) session login, use this procedure:

1. Select a relational server (or database) session in the **Navigator**. Its properties appear in the Editor.
2. Click the **Login** tab. The Login tab appears.
3. Click the **Options** subtab. The Options subtab appears.

**Figure 98–6 Login Tab, Options Subtab, JDBC Options**



Use this table to enter data in the fields on the Options subtab to select the JDBC options to use with this session login:

Option	Description
<b>Queries Should Bind All Parameters<sup>1</sup></b>	By default, TopLink binds all of the query’s parameters. Deselect this option if you do not want TopLink to bind parameters.
<b>Cache All Statements<sup>1</sup></b>	When selected, TopLink caches each prepared statement so that when reexecuted, you avoid the SQL preparation time which improves performance.
<b>Byte Array Binding<sup>1</sup></b>	Select this option if you query binary large object (BLOB) data.
<b>Streams for Binding<sup>1</sup></b>	Select this option if you use a JDBC driver that is more efficient at handling BLOB data using <code>java.io.InputStream</code> and <code>java.io.OutputStream</code> .
<b>Native SQL</b>	By default, TopLink generates SQL using JDBC SQL grammar. Select this option if you want TopLink to use database-specific SQL grammar, for example, if your database driver does not support the full JDBC SQL grammar.

Option	Description
<b>Batch Writing</b> <sup>2</sup>	<p>Select this option if you use a JDBC driver that supports sending groups of INSERT, UPDATE, and DELETE statements to the database in a single transaction, rather than individually.</p> <p>Select <b>JDBC</b> to use the batch writing capabilities of your JDBC driver.</p> <p>Select <b>TopLink</b> to use the batch writing capabilities that TopLink provides. Select this option if your JDBC driver does not support batch writing.</p> <p>Note: if you are using Oracle 9 Database platform, and you want to use TopLink batch writing in combination with optimistic locking, then you must enable parameter binding.</p>
<b>String Binding</b> <sup>1</sup>	<p>Select this option if you query large <code>java.lang.String</code> objects.</p> <p>You can configure the maximum <code>String</code> length (default: 32000 characters).</p>

<sup>1</sup> For more information, see Section 12.11.5, "How to Use Parameterized SQL (Parameter Binding) and Prepared Statement Caching for Optimization".

<sup>2</sup> If you are using the MySQL4 database platform (see Section 96.1.3, "Data Source Platform Types"), use **JDBC** batch writing (do not use **TopLink** batch writing). For more information, see Section 12.11.3, "How to Use Batch Writing for Optimization".

## 98.6.2 How to Configure JDBC Options Using Java

To enable prepared statement caching for all queries, configure at the `Login` level, as Example 98–10 shows. For more information, see Section 12.11.5, "How to Use Parameterized SQL (Parameter Binding) and Prepared Statement Caching for Optimization".

### Example 98–10 Prepared Statement Caching at the Login Level

```
databaseLogin.cacheAllStatements();
databaseLogin.setStatementCacheSize(100);
```

Parameter binding is enabled by default in TopLink. To disable binding, configure at the `Login` level, as Example 98–11 shows. For more information, see Section 12.11.5, "How to Use Parameterized SQL (Parameter Binding) and Prepared Statement Caching for Optimization".

### Example 98–11 Disabling Parameter Binding at the Login Level

```
databaseLogin.dontBindAllParameters();
```

To enable JDBC batch writing, use `Login` method `useBatchWriting`, as Example 98–12 shows:

### Example 98–12 Using JDBC Batch Writing

```
project.getLogin().useBatchWriting();
project.getLogin().setMaxBatchWritingSize(100);
```

## 98.7 Configuring Advanced Options

Most JDBC drivers support the run-time configuration of various options to customize driver operation to meet user needs. TopLink provides direct support (in API, Oracle JDeveloper and TopLink Workbench) for many of the most important options (see Section 98.6, "Configuring JDBC Options"), as well as more advanced options, as this section describes.

You can also configure additional options by specifying properties (see [Section 97.5, "Configuring Properties"](#)).

---

**Note:** Not all drivers support all JDBC options. Selecting a combination of options may result in different behavior from one driver to another. Before selecting JDBC options, consult your JDBC driver documentation.

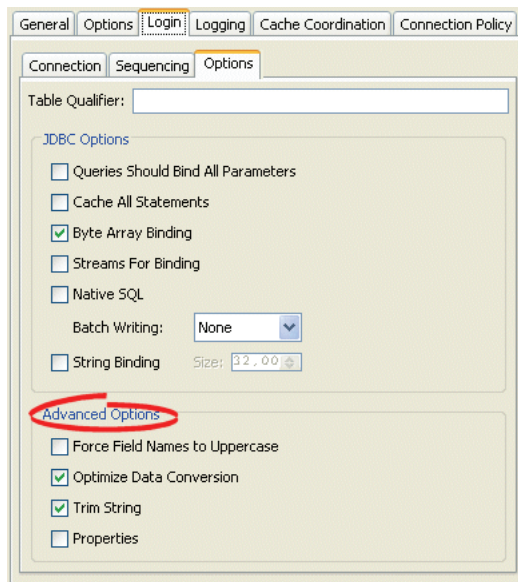
---

### 98.7.1 How to Configure Advanced Options Using TopLink Workbench

To specify the advanced options for a relational server (or database) session login, use this procedure:

1. Select a relational server (or database) session in the **Navigator**. Its properties appear in the Editor.
2. Click the **Login** tab. The Login tab appears.
3. Click the **Options** subtab. The Options subtab appears.

**Figure 98–7 Login Tab, Options Subtab, Advanced Options**



Use this table to enter data in the fields on the Options subtab to select the advanced options to use with this session login:

Option	Description
<b>Force Field Names to Uppercase</b>	By default, TopLink uses the case of field names as returned by the database. If your application expects field names to be uppercase but the database does not return consistent case (for example, if you accessing different databases), enable this option.
<b>Optimize Data Conversion</b>	By default, TopLink optimizes data access by accessing the data from JDBC in the format the application requires. If you are using an older JDBC driver that does not perform data conversion correctly and conflicts with this optimization, disable this optimization.

Option	Description
Trim String	By default, TopLink discards the trailing blanks from CHAR field values. To read and write CHAR field values literally (including any trailing blanks), disable this option.
Properties	Check this option to enable the use of properties for this DatabaseLogin (see <a href="#">Section 97.5, "Configuring Properties"</a> ).

## 98.7.2 How to Configure Advanced Options Using Java

Use the following methods of DatabaseLogin to configure advanced options:

- `setShouldForceFieldNamesToUpperCase`—By default, TopLink uses the case of field names as returned by the database. If your application expects field names to be uppercase but the database does not return consistent case (for example, if you accessing different databases), use this method.
- `setShouldOptimizeDataConversion`—By default, TopLink optimizes data access by accessing the data from JDBC in the format the application requires. If you are using an older JDBC driver that does not perform data conversion correctly and conflicts with this optimization, set this to `false`.
- `setShouldTrimStrings`—By default, TopLink discards the trailing blanks from CHAR field values. To read and write CHAR field values literally (including any trailing blanks), set this to `false`.
- `setProperties`—Set this to `true` to enable the use of properties for this DatabaseLogin (see [Section 97.5, "Configuring Properties"](#)).

## 98.8 Configuring Oracle Database Proxy Authentication

You can configure a database login to use Oracle Database proxy authentication with an Oracle Database platform in JSE applications and JEE applications using OC4J native or managed non-JTA (you set this using `tx-level=local`) data sources with Oracle JDBC driver release 10.1.0.2.0 or later and external connection pools only.

There is no Oracle JDeveloper or TopLink Workbench support for this feature. To configure TopLink to use Oracle Database proxy authentication, you must use Java (see [Section 98.8.1, "How to Configure Oracle Database Proxy Authentication Using Java"](#)).

For more information, see [Section 96.1.4.2, "Oracle Database Proxy Authentication"](#).

You can use TopLink support for Oracle Database proxy authentication by doing the following:

- [Providing Authenticated Reads and Writes of Secured Data Through the Use of an Exclusive Isolated Client Session](#)
- [Providing Authenticated Writes for Database Auditing Purposes with a Client Session](#)
- [Providing Authenticated Reads and Writes with a Database Session](#)

### Providing Authenticated Reads and Writes of Secured Data Through the Use of an Exclusive Isolated Client Session

In this configuration, the client session is an isolated client session (see [Section 87.5, "Isolated Client Sessions"](#)) that uses an exclusive proxy connection. You must acquire the client session using a `ConnectionPolicy` that specifies the proxy authentication user credentials.

Reads and writes of secured data are performed through the proxy-authenticated connection. Reads of nonsecured data occur through nonproxy-authenticated connections.

If you are using Oracle Virtual Private Database (VPD) (see [Section 87.5.1, "Isolated Client Sessions and Oracle Virtual Private Database \(VPD\)"](#)), use this configuration to set up VPD support entirely in the database. That is, rather than making the isolated client session execute SQL (see [Section 92.2, "Using PostAcquireExclusiveConnection Event Handler"](#) and [Section 92.3, "Using PreReleaseExclusiveConnection Event Handler"](#)), the database performs the required set up in an after login trigger using the proxy `session_user`.

### **Providing Authenticated Writes for Database Auditing Purposes with a Client Session**

In this configuration, isolated data or exclusive connections are not required. You must acquire client session using a `ConnectionPolicy` that specifies the proxy authentication user credentials.

Writes are performed through the proxy-authenticated connection. Reads occur through nonproxy-authenticated connections. This enables the database auditing process to access the user that performed the write operations.

### **Providing Authenticated Reads and Writes with a Database Session**

In this configuration, you use a `DatabaseSession` object with a proxy-authenticated login. All reads and writes occur through the proxy-authenticated connection.

---

---

**Note:** Oracle recommends that you exclusively use server and client sessions in a three-tier environment.

Do not use database sessions in a three-tier environment. Ensure that a database session is used by a single user and not accessed concurrently.

---

---

## **98.8.1 How to Configure Oracle Database Proxy Authentication Using Java**

You configure Oracle Database proxy authentication by customizing your session in your Java code, such as through a `SessionCustomizer` when using the `sessions.xml` file. You can wrap a configured `TopLink DataSourceLogin JNDIConnector` with a `TopLink proxy connector instance` (from `oracle.toplink.platform.database.oracle`) appropriate for your JDBC driver and to configure proxy authentication properties.

Regardless of whether you are using the Oracle JDBC Thin driver or OCI driver, use the `OracleJDBC10_1_0_2ProxyConnector` and the property constants defined in `oracle.jdbc.OracleConnection`.

The properties to set are shown in Tables 98–2 through 98–5.

---

**Note:** Property constant names and values are consistent between the two classes except for `PROXYTYPE_` constants (such as `PROXYTYPE_USER_NAME`). In `OracleOCIConnectionPool` these are of type `String` and in `OracleConnection` they are of type `int`. If you are using the Oracle JDBC Thin driver and `OracleJDBC10_1_0_2ProxyConnector`, you must always set these properties as a `String`. For example:

```
login.setProperty(
    "proxytype", Integer.toString(OracleConnection.PROXYTYPE_USER_NAME));
```

---

To configure TopLink to use Oracle Database proxy authentication, do the following:

1. Decide on the proxy type you want to use and create appropriate users and roles.

- a. User Name Authentication:

To authenticate a proxy user `sarah` by user name only, create the user account on Oracle Database using the following:

```
alter user sarah grant connect through dbadminuser
    with roles clerk, reports;
```

In this case, you will need to set the proxy properties shown in [Table 98–2](#).

**Table 98–2 Proxy Properties for User Name Authentication**

Property Name	Property Value
"proxytype"	PROXYTYPE_USER_NAME
PROXY_USER_NAME	"sarah"
PROXY_ROLES	String[] {"role1", "role2", ...}

- b. User Name and Password Authentication:

To authenticate a proxy user `sarah` by user name and password, create the user account on Oracle Database using the following:

```
alter user sarah grant connect through dbadminuser
    authenticated using password
    with roles clerk, reports;
```

In this case, you will need to set the proxy properties shown in [Table 98–3](#).

**Table 98–3 Proxy Properties for User Name and Password Authentication**

Property Name	Property Value
"proxytype"	PROXYTYPE_USER_NAME
PROXY_USER_NAME	"sarah"
PROXY_PASSWORD	"passwordforsarah"
PROXY_ROLES	String[] {"role1", "role2", ...}

- c. Distinguished Name Authentication:

To authenticate a proxy user `sarah` by globally unique distinguished name, create the user account on Oracle Database using the following:

```
create user sarah identified globally as
    'CN=sarah,OU=americas,O=oracle,L=city,ST=ca,C=us';
```

```
alter user sarah grant connect through dbadminuser
    authenticated using distinguished name
    with roles clerk, reports;
```

In this case, you will need to set the proxy properties shown in [Table 98-4](#).

**Table 98-4 Proxy Properties for Distinguished Name Authentication**

Property Name	Property Value
"proxytype"	PROXYTYPE_DISTINGUISHED_NAME
PROXY_DISTINGUISHED_NAME	"CN=sarah,OU=americas,O=oracle,L=city,ST=ca,C=us"
PROXY_ROLES	String[] {"role1", "role2", ...}

**d. Certificate Authentication:**

To authenticate a proxy user sarah by encrypted distinguished name, create the user account on Oracle Database using the following:

```
alter user sarah grant connect through dbadminuser
    authenticated using certificate
    with roles clerk, reports;
```

In this case, you will need to set the proxy properties shown in [Table 98-2](#).

**Table 98-5 Proxy Properties for User Name Authentication**

Property Name	Property Value
"proxytype"	PROXYTYPE_CERTIFICATE
PROXY_CERTIFICATE	byte[] {EncryptedCertificate}
PROXY_ROLES	String[] {"role1", "role2", ...}

**2. Configure your session login using Java code. Do this through a SessionCustomizer when using the sessions.xml file.**

The following example demonstrates how you can wrap the already specified JNDIConnector with the appropriate TopLink proxy authentication connector. You can set the server session's default connection policy to the same proxy authenticated login.

If you use Oracle VPD ([Section 87.5.1, "Isolated Client Sessions and Oracle Virtual Private Database \(VPD\)"](#)), you should set the connection policy to use exclusive connections, and the descriptor for secured data to isolated ([Section 119.13, "Configuring Cache Isolation at the Descriptor Level"](#)).

```
Login login = server.getDatasourceLogin();
// Make sure that external connection pooling is used
login.setUsesExternalConnectionPooling(true);
// Wrap JNDIConnector with OracleJDBC10_1_0_2ProxyConnector
login.setConnector(
    new OracleJDBC10_1_0_2ProxyConnector(
        ((JNDIConnector) login.getConnector()).getName());
ConnectionPolicy policy = server.getDefaultConnectionPolicy();
policy.setPoolName(null);
policy.setLogin(login);
// If using Oracle VPD support, set the connection policy to exclusive
policy.setShouldUseExclusiveConnection(true);
```



Note that you may experience problems when using a data source provided by the application server. In this case, instead of using

```
login.setConnector(new OracleJDBC10_1_0_2ProxyConnector
    (((JNDIConnectorlogin.getConnector()).getName())));
```

create the data source, as follows:

```
oracle.jdbc.pool.OracleDataSource ds = new oracle.jdbc.pool.OracleDataSource();
ds.setUser("MyMainUser");
ds.setPassword("MyPassword");
ds.setUrl("jdbc:oracle:thin:@MyServer:1521:MyDb");
login.setConnector( new OracleJDBC10_1_0_2ProxyConnector(ds));
```

3. Acquire a proxy-authenticated client session through specifying a `ConnectionPolicy` with this user's credentials.

```
ConnectionPolicy policy =
    (ConnectionPolicy)server.getDefaultConnectionPolicy().clone();
Login login = (Login)policy.getLogin().clone();
// Set proxy properties into connection policy's login
login.setProperty("proxytype" , OracleOCIConnectionPool.PROXYTYPE_USER_NAME);
login.setProperty(OracleOCIConnectionPool.PROXY_USER_NAME , "sarah");
policy.setLogin(login);
Session session = server.acquireClientSession(policy);
```



## Configuring an EIS Login

This chapter describes the various components that you must configure to use an EIS login.

This chapter includes the following sections:

- [Introduction to EIS Login Configuration](#)
- [Configuring an EIS Data Source Platform at the Session Level](#)
- [Configuring EIS Connection Specification Options at the Session Level](#)

### 99.1 Introduction to EIS Login Configuration

Table 99–1 lists the configurable options for an EIS login.

**Table 99–1 Configurable Options for EIS Login**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Data source platform at the session level (see <a href="#">Section 99.2, "Configuring an EIS Data Source Platform at the Session Level"</a> )	✓	✓	✓
Connection specification options at the session level (see <a href="#">Section 99.3, "Configuring EIS Connection Specification Options at the Session Level"</a> )	✓	✓	✓
User name and password (see <a href="#">Section 97.2, "Configuring User Name and Password"</a> )	✓	✓	✓
Password encryption (see <a href="#">Section 97.3, "Configuring Password Encryption"</a> )			✓
External connection pooling (see <a href="#">Section 97.4, "Configuring External Connection Pooling"</a> )	✓	✓	✓
Properties (see <a href="#">Section 97.5, "Configuring Properties"</a> )	✓	✓	✓

### 99.2 Configuring an EIS Data Source Platform at the Session Level

For each EIS session, you must specify the platform (such as AQ, for example). This platform configuration overrides the platform at the project level, if configured.

For more information, see the following:

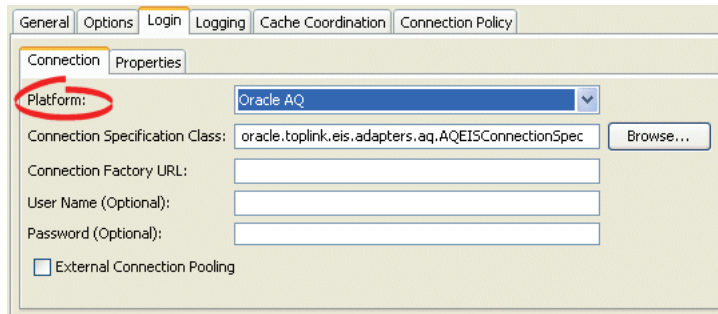
- [Section 20.2, "Configuring Relational Database Platform at the Project Level"](#)
- [Section 96.1.2, "Data Source Login Types"](#)

## 99.2.1 How to Configure an EIS Data Source Platform at the Session Level Using TopLink Workbench

To specify the database platform options for an EIS session login, use this procedure:

1. Select an EIS session in the **Navigator**. Its properties appear in the Editor.
2. Click the **Login** tab. The Login tab appears.
3. Click the **Connection** subtab. The Connection subtab appears.

**Figure 99–1 Login Tab, Connection Subtab, Platform Options**



Use the following information to enter data in the Platform field on the Connection tab to configure the platform:

Field	Description
Platform	The EIS platform for the session. Select from the menu of options. This menu includes all instances of <code>EISPlatform</code> in the TopLink classpath.

## 99.3 Configuring EIS Connection Specification Options at the Session Level

You can configure connection information at the session level for an EIS application. This information is stored in the `sessions.xml` file. The Oracle TopLink runtime uses this information whenever you perform a persistence operation using the session in your EIS application.

This connection configuration overrides the connection information at the project level, if configured. For more information about project-level configuration, see [Section 20.5, "Configuring Development and Deployment Logins"](#) and [Section 73.3, "Configuring EIS Connection Specification Options at the Project Level"](#).

This connection configuration is overridden by the connection information at the connection pool level. For more information about connection pool-level configuration, see [Section 101.5, "Configuring Connection Pool Connection Options"](#).

### 99.3.1 How to Configure EIS Connection Specification Options at the Session Level Using TopLink Workbench

Use this procedure to specify the connection options for an EIS session login.

1. Select an EIS session in the Navigator window. Its properties appear in the Editor window.
2. Click the **Login** tab. The Login tab appears.

3. Click the **Connection** subtab. The Connection tab appears.

**Figure 99–2 Login Tab, Connection Subtab**

The screenshot shows a dialog box with tabs: General, Options, Login, Logging, Cache Coordination, and Connection Policy. The 'Login' tab is active, and the 'Connection' subtab is selected. The 'Platform' dropdown menu is set to 'Oracle AQ'. The 'Connection Specification Class' text box contains 'oracle.toplink.eis.adapters.aq.AQEISConnectionSpec' and has a 'Browse...' button to its right. The 'Connection Factory URL' text box is empty. Below these are 'User Name (Optional):' and 'Password (Optional):' text boxes, and an 'External Connection Pooling' checkbox which is unchecked.

Use the following information to enter data in the connection fields on the tab:

Field	Description
<b>Connection Specification Class</b>	Specify the appropriate connection specification class for the selected <b>Platform</b> . Click <b>Browse</b> to choose from all the classes in the TopLink classpath. (For example: if <b>Platform</b> is <code>oracle.toplink.eis.aq.AQPlatform</code> , use <code>oracle.toplink.eis.aq.AQEISConnectionSpec</code> ).  For more information on platform configuration, see <a href="#">Section 99.2, "Configuring an EIS Data Source Platform at the Session Level"</a> .
<b>Connection Factory URL</b>	Specify the appropriate connection factory URL for the selected <b>Connection Specification Class</b> (For example: <code>jdbc:oracle:thin@localhost:1521:orcl</code> ).



---

## Creating an Internal Connection Pool

This chapter explains how to create TopLink internal connection pools.

This chapter includes the following sections:

- [Introduction to the Internal Connection Pool Creation](#)
- [Creating an Internal Connection Pool](#)

### 100.1 Introduction to the Internal Connection Pool Creation

You can create internal connection pools (see [Section 96.1.6.1, "Internal Connection Pools"](#)) only for server sessions. For more information, see [Section 100.2, "Creating an Internal Connection Pool"](#).

After you create an internal connection pool, you must configure its various options (see [Chapter 101, "Configuring an Internal Connection Pool"](#))

After you create and configure a sequence connection pool (see [Section 96.1.6.4, "Sequence Connection Pools"](#)), TopLink will use this pool whenever it needs to assign an identifier to a new object.

After you create and configure a named connection pool (see [Section 96.1.6.5, "Application-Specific Connection Pools"](#)), you use it in your application by passing in a `ConnectionPolicy` when you acquire a client session (see [Section 90.4.4, "How to Acquire a Client Session that Uses a Named Connection Pool"](#)).

### 100.2 Creating an Internal Connection Pool

You can create an internal connection pool using Oracle JDeveloper, TopLink Workbench, or Java code.

#### 100.2.1 How to Create an Internal Connection Pool Using TopLink Workbench

Before you create a connection pool, you must first create a server session (see [Section 88.4, "Creating a Server Session"](#)).

To create a new TopLink internal connection pool, use this procedure:

1. Select the server session in the **Navigator** in which you want to create a connection pool.
2. Click the appropriate button on the toolbar to create the type of connection pool you want:
  - To create a named connection pool, select **Create a New Named Connection Pool**, enter a name, and then click **OK**.



- To create a sequence connection pool, select **Add the Sequence Connection Pool**.
- To create a write connection pool, select **Add the Write Connection Pool**.

You can also create a new internal connection pool by right-clicking the server session configuration in the **Navigator** and selecting **New > Named Connection Pool, Sequence Connection Pool, or Write Connection Pool** from the menu.

## 100.2.2 How to Create an Internal Connection Pool Using Java

Using Java, you can create read, write, and named connection pools.

[Example 100–1](#) shows how to create connection pools. [Example 100–2](#) shows an optimized connection pool, using the same connection for both read and write operations.

### **Example 100–1 Creating Connection Pools**

```
// Read
ConnectionPool pool = new ConnectionPool();
pool.setName("read");
pool.setLogin(login);
pool.setMaxNumberOfConnections(50);
pool.setMinNumberOfConnections(50);
serverSession.setReadConnectionPool(pool);
```

```
// Write
ConnectionPool pool = new ConnectionPool();
pool.setName("default");
pool.setLogin(login);
pool.setMaxNumberOfConnections(50);
pool.setMinNumberOfConnections(50);
serverSession.addConnectionPool(pool);
```

```
// Named
ConnectionPool pool = new ConnectionPool();
pool.setName("admin");
pool.setLogin(login);
pool.setMaxNumberOfConnections(2);
pool.setMinNumberOfConnections(2);
serverSession.addConnectionPool(pool);
```

You can also use a single connection pools for both read and write operations, when the read and write pools use the same login information. This method requires fewer connections

### **Example 100–2 Using a Single Pool for Read and Write**

```
ConnectionPool pool = new ConnectionPool();
pool.setName("default");
pool.setLogin(login);
pool.setMaxNumberOfConnections(50);
pool.setMinNumberOfConnections(50);
serverSession.setReadConnectionPool(pool);
serverSession.addConnectionPool(pool);
```



## Configuring an Internal Connection Pool

This chapter describes the various components that you must configure to use an internal connection pool.

This chapter includes the following sections:

- [Introduction to the Internal Connection Pool Configuration](#)
- [Configuring Connection Pool Sizes](#)
- [Configuring Properties](#)
- [Configuring a Nontransactional Read Login](#)
- [Configuring Connection Pool Connection Options](#)
- [Configuring Exclusive Read Connections](#)

### 101.1 Introduction to the Internal Connection Pool Configuration

When you are using server sessions, you can configure the default read connection pool and write connection pool. You can also configure the optional named connection pools and sequence connection pool you may have created (see [Section 100.1, "Introduction to the Internal Connection Pool Creation"](#)).

[Table 101–1](#) lists the configurable options for an internal connection pool.

**Table 101–1 Configurable Options for Connection Pool**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Connection pool sizes (see <a href="#">Section 101.2, "Configuring Connection Pool Sizes"</a> )	✓	✓	✓
Exclusive read connections (see <a href="#">Section 101.6, "Configuring Exclusive Read Connections"</a> ) <sup>1</sup>	✓	✓	✓
Nontransactional read login (see <a href="#">Section 101.4, "Configuring a Nontransactional Read Login"</a> ) <sup>1</sup>	✓	✓	✓
Properties (see <a href="#">Section 101.3, "Configuring Properties"</a> )	✓	✓	✓
Connection pool connection options (see <a href="#">Section 101.5, "Configuring Connection Pool Connection Options"</a> ) <sup>2 3</sup>	✓	✓	✓

<sup>1</sup> Read connection pools only.

<sup>2</sup> Not applicable to write connection pools.

<sup>3</sup> Applicable for sequence connection pools.

## 101.2 Configuring Connection Pool Sizes

By default, if using TopLink internal connection pooling, the TopLink write connection pool maintains a minimum of five connections and a maximum of ten. The read connection pool maintains a minimum and maximum of two connections.

Connection pool size can significantly influence the concurrency of your application and should be set to be large enough to handle your expected application load.

---

---

**Tip:** To maintain compatibility with JDBC drivers that do not support many connections, the default number of connections is small. If your JDBC driver supports it, use a larger number of connections for reading and writing.

---

---

The smallest value you can enter is 0. Setting the maximum number of connections to 0 will make it impossible for TopLink to allocate any connections.

The minimum number of connections should always be less than or equal to the maximum number of connections.

If the maximum number of connections is in use, the next connection request will be blocked until a connection is available.

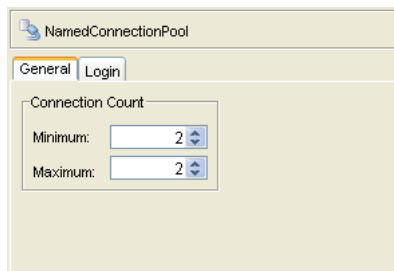
You can configure the connection pool size using Oracle JDeveloper TopLink Editor, TopLink Workbench (see [Section 101.2.1, "How to Configure Connection Pool Size Using TopLink Workbench"](#)), or Java (see [Section 101.2.2, "How to Configure Connection Pool Size Using Java"](#)).

### 101.2.1 How to Configure Connection Pool Size Using TopLink Workbench

To specify the minimum and maximum number of connections in a TopLink internal connection pool, use this procedure:

1. Expand a server session to reveal its connection pools in the **Navigator**.
2. Select a connection pool in the **Navigator**. Its properties appear in the Editor.
3. Click the **General** tab. The General tab appears.

**Figure 101–1** General Tab, Connection Count Options



Enter the desired minimum and maximum number of connections and press **Enter** or use the increment and decrement arrows.

### 101.2.2 How to Configure Connection Pool Size Using Java

Using Java, you can set the connection pool size by using the `setMaxNumberOfConnections` and `setMinNumberOfConnection` method.

[Example 101–1](#) shows how to configure the connection pool size for a read connection.

**Example 101–1 Configuring Connection Pool Size**

```

ConnectionPool pool = new ConnectionPool();
pool.setName("read");
pool.setLogin(login);
pool.setMaxNumberOfConnections(50);
pool.setMinNumberOfConnections(50);
serverSession.setReadConnectionPool(pool);

```

## 101.3 Configuring Properties

For all connection pools, except write connection pools, you can specify arbitrary named values, called properties.

Some data sources require additional, driver-specific properties not supported in the `ConnectionPool` API. Add these properties to the `ConnectionPool` so that TopLink can pass them to the driver.

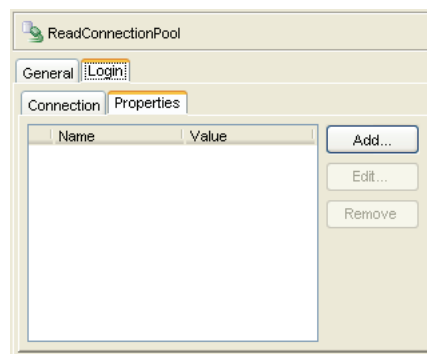
You can configure properties using Oracle JDeveloper TopLink Editor, TopLink Workbench (see [Section 101.3.1, "How to Configure Properties Using TopLink Workbench"](#)), or Java (see [Section 101.3.2, "How to Configure Properties Using Java"](#)).

### 101.3.1 How to Configure Properties Using TopLink Workbench

To specify arbitrary named value pairs that TopLink associates with a `ConnectionPool`, use this procedure:

1. Expand a server session to reveal its connection pools in the **Navigator**.
2. Select a read, named, or sequence connection pool in the **Navigator**. Its properties appear in the Editor.
3. Click the **Login** tab. The Login tab appears.
4. Click the **Properties** subtab. The Properties subtab appears.

**Figure 101–2 Login Tab, Properties Subtab**



Complete the **Add Property** dialog box.

Use the following information to add or edit a login property on the Add Property dialog box to add or edit a login property:

Option	Description
Name	The name by which TopLink retrieves the property value using the <code>DatasourceLogin</code> method <code>getProperty</code> .
Value	The value TopLink retrieves using the <code>DatasourceLogin</code> method <code>getProperty</code> passing in the corresponding property name.  Using TopLink Workbench, you can set only character values which TopLink returns as <code>String</code> objects.

To add (or change) a new **Name/Value** property, click **Add** (or **Edit**).

To delete an existing property, select the **Name/Value** row and click **Remove**.

### 101.3.2 How to Configure Properties Using Java

Using Java, you can set any `Object` value using the `DatasourceLogin` method `setProperty`. To remove a property, use the `DatasourceLogin` method `removeProperty`.

## 101.4 Configuring a Nontransactional Read Login

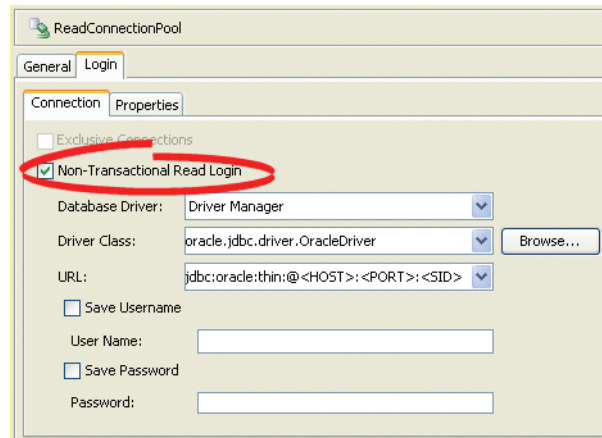
When you use an external transaction controller (see [Section 89.9, "Configuring the Server Platform"](#)), establishing a connection requires not only the usual connection setup overhead, but also transactional overhead. If your application reads data only to display it and only infrequently modifies data, you can configure an internal read connection pool to use its own connection specification that does not use the external transaction controller. This may improve performance by reducing the time it takes to establish a new read connection.

You can configure the nontransactional read login using Oracle JDeveloper TopLink Editor, TopLink Workbench (see [Section 101.4.1, "How to Configure Nontransactional Read Login Using TopLink Workbench"](#)), or Java (see [Section 101.4.2, "How to Configure Nontransactional Read Login Using Java"](#)).

### 101.4.1 How to Configure Nontransactional Read Login Using TopLink Workbench

To enable the configuration of nontransactional connection information for a TopLink read connection pool, use this procedure:

1. Expand a server session to reveal its connection pools in the **Navigator**.
2. Select a read connection pool in the **Navigator**. Its properties appear in the Editor.
3. Click the **Login** tab. The Login tab appears.
4. Click the **Connection** subtab. The Connection subtab appears.

**Figure 101–3 Login Tab, Connection Subtab**

To enable a nontransactional read login, select the **Use Non-Transactional Read Login** option (see [Section 96.1.1, "Externally Managed Transactional Data Sources"](#)). Continue with [Section 101.5, "Configuring Connection Pool Connection Options"](#) to specify the connection information.

### 101.4.2 How to Configure Nontransactional Read Login Using Java

Use the `getLogin` method of your connection pool to obtain a `DatabaseLogin`, and then use the following `DatabaseLogin` methods to configure the nontransactional read login options:

- `useExternalTransactionController`
- `setDriverClass`
- `setDriverClassName`
- `setDriverURLHeader`

## 101.5 Configuring Connection Pool Connection Options

By default, connection pools use the login configuration specified for their session (see [Section 98.3, "Configuring Database Login Connection Options"](#) and [Section 99.3, "Configuring EIS Connection Specification Options at the Session Level"](#)).

For read, named, and sequence connection pools, you can override the session login configuration on a per-connection pool basis.

To configure login configuration for a read connection pool, you must first enable it for a nontransactional read login (see [Section 101.4, "Configuring a Nontransactional Read Login"](#)).

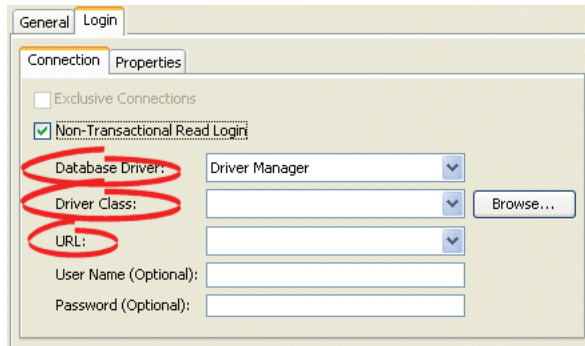
You can configure the connection pool connection options using Oracle JDeveloper TopLink Editor or TopLink Workbench (see [Section 101.5.1, "How to Configure Connection Pool Connection Options Using TopLink Workbench"](#)).

### 101.5.1 How to Configure Connection Pool Connection Options Using TopLink Workbench

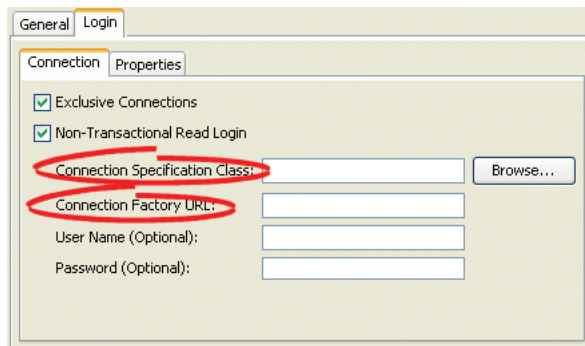
To configure connection information for a TopLink read, named, or sequence connection pool, use this procedure:

1. Expand a server session to reveal its connection pools in the **Navigator**.
2. Select a read, named, or sequence connection pool in the **Navigator**. Its properties appear in the Editor.
3. Click the **Login** tab. The Login tab appears.
4. Click the **Connection** subtab. The Connection subtab appears.

**Figure 101–4 Login Tab, Connection Subtab, Relational Session Connection Pool Options**



**Figure 101–5 Login Tab, Connection Subtab, EIS Session Connection Pool Options**



5. Ensure the **Use Non-Transaction Read Login** option is selected.

Use the following information to complete fields on the Connection subtab:

Field	Description
<b>Database Driver</b> <sup>1</sup>	<p>Specify the appropriate database driver:</p> <ul style="list-style-type: none"> <li>■ <b>Driver Manager:</b> Specify this option to configure the driver class and URL used to connect to the database. In this case, you must configure the <b>Driver Class</b> and <b>Driver URL</b> fields.</li> <li>■ <b>J2EE Datasource:</b> Specify this option to use a Java EE data source already configured on your target application server. In this case, you must configure the <b>Datasource Name</b> field.</li> </ul> <p>Note: If you select <b>J2EE Datasource</b>, you must use external connection pooling. You cannot use internal connection pools with this <b>Database Driver</b> option (for more information, see <a href="#">Section 97.4, "Configuring External Connection Pooling"</a>).</p>

Field	Description
<b>Driver Class</b> <sup>1</sup>	Configure this field when <b>Database Driver</b> is set to <b>Driver Manager</b> . Select from the menu of options. This menu includes all JDBC drivers in the TopLink application classpath.
<b>URL</b> <sup>1</sup>	Configure this field when <b>Database Driver</b> is set to <b>Driver Manager</b> . Select from the menu of options relevant to the selected <b>Driver Class</b> and edit the URL to suit your data source.
<b>Datasource Name</b> <sup>1</sup>	Configure this field when <b>Database Driver</b> is set to <b>J2EE Datasource</b> . Specify any valid JNDI name that identifies the Java EE data source preconfigured on your target application server (For example: jdbc/EmployeeDB).  By convention, all such names should resolve to the JDBC subcontext (relative to the standard java:comp/env naming context that is the root of all provided resource factories).
<b>Connection Specification Class</b> <sup>2</sup>	Specify the appropriate connection specification class for the selected <b>Platform</b> . Click <b>Browse</b> to choose from all the classes in the TopLink classpath. (For example: if <b>Platform</b> is oracle.toplink.eis.aq.AQPlatform, use oracle.toplink.eis.aq.AQEISConnectionSpec).  For more information on platform configuration, see <a href="#">Section 99.2, "Configuring an EIS Data Source Platform at the Session Level"</a> .
<b>Connection Factory URL</b> <sup>2</sup>	Specify the appropriate connection factory URL for the selected <b>Connection Specification Class</b> (For example: jdbc:oracle:thin@:localhost:1521:orcl).

<sup>1</sup> For sessions that contain a DatabaseLogin.

<sup>2</sup> For sessions that contain an EISLogin.

## 101.6 Configuring Exclusive Read Connections

An exclusive connection is one that TopLink allocates specifically to a given session and one that is never used by any other session.

Allowing concurrent reads on the same connection reduces the number of read connections required and reduces the risk of having to wait for an available connection. However, many JDBC drivers do not support concurrent reads.

If you are using internal connection pools (see [Section 96.1.6.1, "Internal Connection Pools"](#)), you can configure TopLink to acquire an exclusive connection from the read connection pool.

By default, TopLink acquires exclusive read connections.

If you are using external connection pools, read connections are always exclusive.

You can configure the connection pool size using Oracle JDeveloper TopLink Editor or TopLink Workbench (see [Section 101.6.1, "How to Configure Exclusive Read Connections Using TopLink Workbench"](#)).

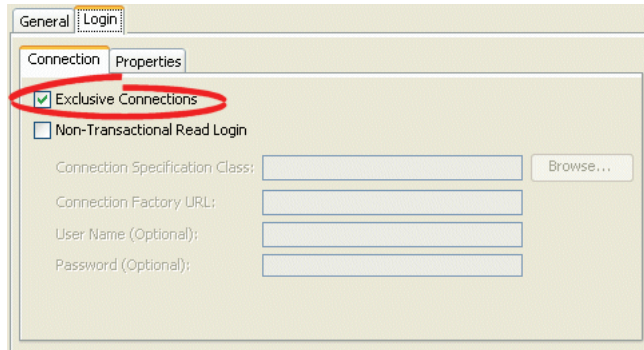
### 101.6.1 How to Configure Exclusive Read Connections Using TopLink Workbench

To configure a TopLink read connection pool to allocate exclusive connections, use this procedure:

1. Expand a server session to reveal its connection pools in the **Navigator**.

2. Select a read connection pool in the **Navigator**. Its properties appear in the Editor.
3. Click the **Login** tab. The Login tab appears.
4. Click the **Connection** subtab. The Connection subtab appears.

**Figure 101–6 Login Tab, Connection Subtab, Exclusive Connections Option**



Select the **Exclusive Connections** option to configure TopLink to acquire an exclusive connection from the read connection pool.

Deselect the **Exclusive Connections** option to configure TopLink to share read connections and allow concurrent reads. Before selecting this option, ensure that your JDBC driver supports concurrent reads.



# Part XXIII

---

## Cache

This part describes using the TopLink object cache in both distributed and nondistributed applications. It contains the following chapters:

- [Chapter 102, "Introduction to Cache"](#)  
This chapter describes each of the different TopLink cache types and important cache concepts.
- [Chapter 103, "Configuring a Coordinated Cache"](#)  
This chapter explains how to configure TopLink coordinated cache options common to two or more coordinated cache types.
- [Chapter 104, "Configuring a JMS Coordinated Cache"](#)  
This chapter explains how to configure a TopLink JMS coordinated cache.
- [Chapter 105, "Configuring an RMI Coordinated Cache"](#)  
This chapter explains how to configure a TopLink RMI coordinated cache.
- [Chapter 106, "Configuring a CORBA Coordinated Cache"](#)  
This chapter explains how to configure a TopLink CORBA coordinated cache.
- [Chapter 107, "Configuring a Custom Coordinated Cache"](#)  
This chapter explains how to configure a custom TopLink coordinated cache.



---

---

## Introduction to Cache

The TopLink cache is an in-memory repository that stores recently read or written objects based on class and primary key values. TopLink uses the cache to do the following:

- Improve performance by holding recently read or written objects and accessing them in-memory to minimize database access.
- Manage locking and isolation level.
- Manage object identity.

This chapter includes the following sections:

- [Cache Architecture](#)
- [Cache Concepts](#)
- [Cache Coordination](#)
- [Cache API](#)

### 102.1 Cache Architecture

TopLink uses two types of cache: the **session cache** maintains objects retrieved from and written to the data source; and the **unit of work cache** holds objects while they participate in transactions. When a unit of work successfully commits to the data source, TopLink updates the session cache accordingly.

---

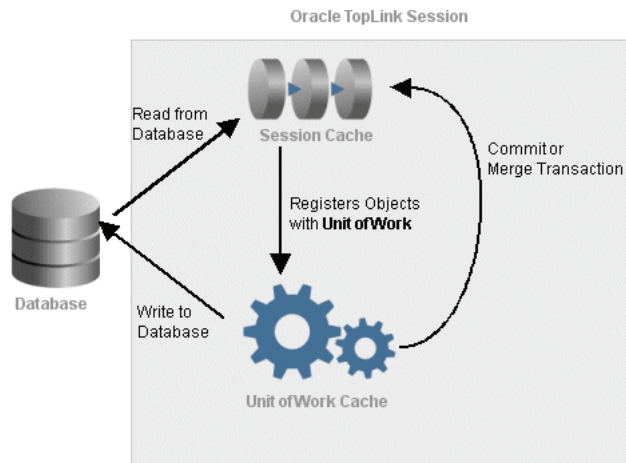
---

**Note:** You can also configure a query to cache its results (see [Section 111.13.1, "How to Cache Results in a ReadQuery"](#))

---

---

As [Figure 102–1](#) shows, the session cache and the unit of work cache work together with the data source connection to manage objects in a TopLink application. The object life cycle relies on these three mechanisms.

**Figure 102–1 Object Life Cycle and the TopLink Caches**

### 102.1.1 Session Cache

The session cache is a shared cache that services clients attached to a given session. When you read objects from or write objects to the data source using a client session, TopLink saves a copy of the objects in the parent server session's cache and makes them accessible to all other processes in the session.

TopLink adds objects to the session cache from the following:

- The data store, when TopLink executes a read operation
- The unit of work cache, when a unit of work successfully commits a transaction

An isolated client session is a special type of client session that provides its own session cache isolated from the shared object cache of its parent server session. The isolated client session cache can be used to improve user-based security or to avoid caching highly volatile data. For more information, see [Section 87.5, "Isolated Client Sessions"](#).

### 102.1.2 Unit of Work Cache

The unit of work cache services operations within the unit of work. It maintains and isolates objects from the session cache, and writes changed or new objects to the session cache after the unit of work commits changes to the data source.

## 102.2 Cache Concepts

This section describes concepts unique to the TopLink cache, including the following:

- [Cache Type and Object Identity](#)
- [Querying and the Cache](#)
- [Handling Stale Data](#)
- [Explicit Query Refreshes](#)
- [Cache Invalidation](#)
- [Cache Coordination](#)
- [Cache Isolation](#)

- [Cache Locking and Transaction Isolation](#)
- [Cache Optimization](#)

## 102.2.1 Cache Type and Object Identity

TopLink preserves object identity through its cache using the primary key attributes of a persistent entity. These attributes may or may not be assigned through sequencing (see [Section 15.2.6, "Projects and Sequencing"](#)). In a Java application, object identity is preserved if each object in memory is represented by one, and only one, object instance. Multiple retrievals of the same object return references to the same object instance—not multiple copies of the same object.

Maintaining object identity is extremely important when the application's object model contains circular references between objects. You must ensure that the two objects are referencing each other directly, rather than copies of each other. Object identity is important when multiple parts of the application may be modifying the same object simultaneously.

Oracle recommends that you always maintain object identity. Disable object identity only if absolutely necessary, for example, for read-only objects (see [Section 119.3, "Configuring Read-Only Descriptors"](#)).

You can configure how object identity is managed on a class-by-class basis. The `ClassDescriptor` object provides the cache and identity map options described in [Table 102-1](#).

**Table 102-1** *Cache and Identity Map Options*

Option (Identity Map)	Caching	Guaranteed Identity	Memory Use
<a href="#">Full Identity Map</a>	Yes	Yes	Very High
<a href="#">Weak Identity Map</a>	Yes	Yes	Low
<a href="#">Soft Identity Map</a>	Yes	Yes	High
<a href="#">Soft Cache Weak Identity Map and Hard Cache Weak Identity Map</a>	Yes	Yes	Medium-high
<a href="#">No Identity Map</a>	No	No	None

For more information, see [Section 102.2.1.6, "Guidelines for Configuring the Cache and Identity Maps"](#).

### 102.2.1.1 Full Identity Map

This option provides full caching and guaranteed identity: objects are never flushed from memory unless they are deleted.

It caches all objects and does not remove them. Cache size doubles whenever the maximum size is reached. This method may be memory-intensive when many objects are read. Do not use this option on batch operations.

Oracle recommends using this identity map when the data set size is small and memory is in large supply.

### 102.2.1.2 Weak Identity Map

This option is similar to the full identity map, except that the map holds the objects by using weak references. This method allows full garbage collection and provides full caching and guaranteed identity.

The weak identity map uses less memory than full identity map but also does not provide a durable caching strategy across client/server transactions. Objects are available for garbage collection when the application no longer references them on the server side (that is, from within the server JVM).

### 102.2.1.3 Soft Identity Map

This option is similar to the weak identity map, except that the map uses soft references instead of weak references. This method allows full garbage collection and provides full caching and guaranteed identity.

The soft identity map allows for optimal caching of the objects, while still allowing the JVM to garbage collect the objects if memory is low.

### 102.2.1.4 Soft Cache Weak Identity Map and Hard Cache Weak Identity Map

These options are similar to the weak identity map, except that they maintain the most frequently used subcache. The subcache uses soft or hard references to ensure that these objects are garbage-collected only if the system is low on memory.

The soft cache weak identity map and hard cache weak identity map provide more efficient memory use. They release objects as they are garbage-collected, except for a fixed number of most recently used objects. Note that weakly cached objects might be flushed if the transaction spans multiple client-server invocations. The size of the subcache is proportional to the size of the identity map, as specified by the `ClassDescriptor` method `setIdentityMapSize`. You should set this cache size to be as large as the maximum number of objects (of the same type) referenced within a transaction (see [Section 119.12, "Configuring Cache Type and Size at the Descriptor Level"](#)).

Oracle recommends using this identity map in most circumstances as a means to control memory used by the cache.

For more information, see [Section 102.2.1.7, "What You May Need to Know About the Internals of Weak, Soft, and Hard Identity Maps"](#).

### 102.2.1.5 No Identity Map

This option does not preserve object identity and does not cache objects.

Oracle does not recommend using the no identity map option. Instead, review the alternatives of cache invalidation and isolated caching.

### 102.2.1.6 Guidelines for Configuring the Cache and Identity Maps

You can configure the cache at the project ([Section 117.10, "Configuring Cache Type and Size at the Project Level"](#)) or descriptor ([Section 119.12, "Configuring Cache Type and Size at the Descriptor Level"](#)) level.

Use the following guidelines when configuring your cache and identity map:

- If objects with a long life span and object identity are important, use a `SoftIdentityMap`, `SoftCacheWeakIdentityMap` or `HardCacheWeakIdentityMap` policy. For more information on when to choose one or the other, see [Section 102.2.1.7, "What You May Need to Know About the Internals of Weak, Soft, and Hard Identity Maps"](#).
- If object identity is important, but caching is not, use a `WeakIdentityMap` policy.
- If an object has a long life span or requires frequent access, or object identity is important, use a `FullIdentityMap` policy.

---

---

**WARNING:** Use the `FullIdentityMap` only if the class has a small number of finite instances. Otherwise, a memory leak will occur.

---

---

- If an object has a short life span or requires frequent access, and identity is not important, use a `CacheIdentityMap` policy.
- If objects are discarded immediately after being read from the database, such as in a batch operation, use a `NoIdentityMap` policy. The `NoIdentityMap` does not preserve object identity.

---

---

**Note:** Oracle does not recommend the use of `CacheIdentityMap` and `NoIdentityMap` policies.

---

---

### 102.2.1.7 What You May Need to Know About the Internals of Weak, Soft, and Hard Identity Maps

The `WeakIdentityMap` and `SoftIdentityMap` use JVM weak and soft references to ensure that any object referenced by the application is held in the cache. Once the application releases its reference to the object, the JVM is free to garbage-collect the objects. The timing of a weak and soft reference garbage collection is determined by the JVM. In general, you could expect a weak reference to be garbage-collected on each JVM garbage collection, and a soft reference to be garbage-collected when the JVM determines memory is low.

The `SoftCacheWeakIdentityMap` and `HardCacheWeakIdentityMap` types of identity map contain the following two caches:

- Reference cache: implemented as a `LinkedList` that contains soft or hard references, respectively.
- Weak cache: implemented as a `HashMap` that contains weak references.

When you create a `SoftCacheWeakIdentityMap` or `HardCacheWeakIdentityMap` with a specified size, the reference cache `LinkedList` is exactly this size. The weak cache `HashMap` is initialized to 100 percent of the specified size: the weak cache will grow when more objects than the specified size are read in. Because `TopLink` does not control garbage collection, the JVM can reap the weakly held objects whenever it sees fit.

Because the reference cache is implemented as a `LinkedList`, new objects are added to the end of the list. Because of this, it is by nature a least recently used (LRU) cache: fixed size, object at the top of the list is deleted, provided the maximum size has been reached.

The `SoftCacheWeakIdentityMap` and `HardCacheWeakIdentityMap` are essentially the same type of identity map, with the former being the subclass of the latter. The `HardCacheWeakIdentityMap` was constructed to work around an issue with some JVMs; the `SoftCacheWeakIdentityMap` inherits this feature.

If your application reaches a low system memory condition frequently enough, or if your platform's JVM treats weak and soft references the same, the objects in the reference cache may be garbage-collected so often that you will not benefit from the performance improvement provided by it. If this is the case, Oracle recommends that you use the `HardCacheWeakIdentityMap`. It is identical to the `SoftCacheWeakIdentityMap` except that it uses hard references in the reference cache. This guarantees that your application will benefit from the performance improvement provided by it.

When an object in a `HardCacheWeakIdentityMap` or `SoftCacheWeakIdentityMap` is pushed out of the reference cache, it gets put in the weak cache. Although it is still cached, TopLink cannot guarantee that it will be there for any length of time because the JVM can decide to garbage-collect weak references at anytime.

## 102.2.2 Querying and the Cache

A query that is run against the shared session cache is known as an **in-memory query**. Careful configuration of in-memory querying can improve performance (see [Section 108.16.2, "How to Use In-Memory Queries"](#)).

By default, a query that looks for a single object based on primary key attempts to retrieve the required object from the cache first, searches the data source only if the object is not in the cache. All other query types search the database first, by default. You can specify whether a given query runs against the in-memory cache, the database, or both.

For more information, see [Section 108.16, "Queries and the Cache"](#).

## 102.2.3 Handling Stale Data

**Stale data** is an artifact of caching, in which an object in the cache is not the most recent version committed to the data source. To avoid stale data, implement an appropriate cache locking strategy.

By default, TopLink optimizes concurrency to minimize cache locking during read or write operations. Use the default TopLink isolation level, unless you have a very specific reason to change it. For more information on isolation levels in TopLink, see [Section 102.2.7, "Cache Isolation"](#).

Cache locking regulates when processes read or write an object. Depending on how you configure it, cache locking determines whether a process can read or write an object that is in use within another process.

A well-managed cache makes your application more efficient. There are very few cases in which you turn the cache off entirely, because the cache reduces database access, and is an important part of managing object identity.

To make the most of your cache strategy and to minimize your application's exposure to stale data, Oracle recommends the following:

- [Configuring a Locking Policy](#)
- [Configuring the Cache on a Per-Class Basis](#)
- [Forcing a Cache Refresh when Required on a Per-Query Basis](#)
- [Configuring Cache Invalidation](#)
- [Configuring Cache Coordination](#)

### 102.2.3.1 Configuring a Locking Policy

Make sure you configure a locking policy so that you can prevent or at least identify when values have already changed on an object you are modifying. Typically, this is done using optimistic locking. TopLink offers several locking policies such as numeric version field, time-stamp version field, and some or all fields.

For more information, see [Section 119.26, "Configuring Locking Policy"](#).



### 102.2.3.2 Configuring the Cache on a Per-Class Basis

If other applications can modify the data used by a particular class, use a weaker style of cache for the class. For example, the `SoftCacheWeakIdentityMap` or `WeakIdentityMap` minimizes the length of time the cache maintains an object whose reference has been removed.

For more information, see [Section 119.12, "Configuring Cache Type and Size at the Descriptor Level"](#).

### 102.2.3.3 Forcing a Cache Refresh when Required on a Per-Query Basis

Any query can include a flag that forces TopLink to go to the data source for the most up-to-date version of selected objects and update the cache with this information.

For more information, see the following:

- [Section 102.4.2, "Cache Refresh API"](#)
- [Section 109.2, "Using DatabaseQuery Queries"](#)
- [Section 109.3, "Using Named Queries"](#)

### 102.2.3.4 Configuring Cache Invalidation

Using descriptor API, you can designate an object as invalid: when any query attempts to read an invalid object, TopLink will go to the data source for the most up to date version of that object and update the cache with this information. You can manually designate an object as invalid or use a `CacheInvalidationPolicy` to control the conditions under which an object is designated invalid.

For more information, see [Section 102.2.5, "Cache Invalidation"](#).

### 102.2.3.5 Configuring Cache Coordination

If your application is primarily read-based and the changes are all being performed by the same Java application operating with multiple, distributed sessions, you may consider using the TopLink cache coordination feature. Although this will not prevent stale data, it should greatly minimize it.

For more information, see [Section 102.2.6, "Cache Coordination"](#).

## 102.2.4 Explicit Query Refreshes

Some distributed systems require only a small number of objects to be consistent across the servers in the system. Conversely, other systems require that several specific objects must always be guaranteed to be up-to-date, regardless of the cost. If you build such a system, you can explicitly refresh selected objects from the database at appropriate intervals, without incurring the full cost of distributed cache coordination.

To implement this type of strategy, do the following:

1. Configure a set of queries that refresh the required objects.
2. Establish an appropriate refresh policy.
3. Invoke the queries as required to refresh the objects.

### 102.2.4.1 Refresh Policy

When you execute a query, if the required objects are in the cache, TopLink returns the cached objects without checking the database for a more recent version. This reduces the number of objects that TopLink must build from database results, and is optimal

for noncoordinated cache environments. However, this may not always be the best strategy for a coordinated cache environment.

To override this behavior, set a refresh policy that specifies that the objects from the database always take precedence over objects in the cache. This updates the cached objects with the data from the database.

You can implement this type of refresh policy on each TopLink descriptor, or just on certain queries, depending upon the nature of the application.

For more information, see the following:

- [Section 119.9, "Configuring Cache Refreshing"](#)
- [Section 108.16.5, "How to Refresh the Cache"](#)

---

---

**Note:** Refreshing does not prevent phantom reads from occurring. See [Section 108.16.8.3, "Refreshing Finder Results"](#).

---

---

#### 102.2.4.2 EJB 2.n CMP Finders and Refresh Policy

When you invoke a `findByPrimaryKey` finder, if the object exists in the cache, TopLink returns that copy. This is the default behavior, regardless of the refresh policy. To force a database query, you can configure the query to refresh by calling `refreshIdentityMapResult` method on it.

For more information, see the following:

- [Section 108.16, "Queries and the Cache"](#)
- [Section 119.7.1.9, "Configuring Named Query Options"](#)

### 102.2.5 Cache Invalidation

By default, objects remain in the session cache until they are explicitly deleted (see [Section 114.7, "Deleting Objects"](#)) or garbage collected when using a weak identity map (see [Section 117.10, "Configuring Cache Type and Size at the Project Level"](#)).

Alternatively, you can configure any object with a `CacheInvalidationPolicy` that lets you specify, either automatically or manually, under what circumstances a cached object is invalid: when any query attempts to read an invalid object, TopLink will go to the data source for the most up-to-date version of that object, and update the cache with this information.

You can use any of the following `CacheInvalidationPolicy` instances:

- `DailyCacheInvalidationPolicy`: the object is automatically flagged as invalid at a specified time of day.
- `NoExpiryCacheInvalidationPolicy`: the object can only be flagged as invalid by explicitly calling `oracle.toplink.sessions.IdentityMapAccessor.invalidateObject`.
- `TimeToLiveCacheInvalidationPolicy`: the object is automatically flagged as invalid after a specified time period has elapsed since the object was read.

You can configure a cache invalidation policy in the following ways:

- At the project level that applies to all objects ([Section 117.13, "Configuring Cache Expiration at the Project Level"](#))

- At the descriptor level to override the project level configuration on a per-object basis (Section 119.16, "Configuring Cache Expiration at the Descriptor Level")
- At the query level that applies to the results returned by the query (Section 111.13.2, "How to Configure Cache Expiration at the Query Level")

If you configure a query to cache results in its own internal cache (see Section 108.16.7, "How to Cache Query Results in the Query Cache"), the cache invalidation policy you configure at the query level applies to the query's internal cache in the same way it would apply to the session cache.

If you are using a coordinated cache (see Section 102.2.6, "Cache Coordination"), you can customize how TopLink communicates the fact that an object has been declared invalid. For more information, see Section 119.15, "Configuring Cache Coordination Change Propagation at the Descriptor Level".

## 102.2.6 Cache Coordination

The need to maintain up-to-date data for all applications is a key design challenge for building a distributed application. The difficulty of this increases as the number of servers within an environment increases. TopLink provides a distributed cache coordination feature that ensures data in distributed applications remains current.

Cache coordination reduces the number of optimistic lock exceptions encountered in a distributed architecture, and decreases the number of failed or repeated transactions in an application. However, cache coordination in no way eliminates the need for an effective locking policy. To effectively ensure working with up-to-date data, cache coordination must be used with optimistic or pessimistic locking. Oracle recommends that you use cache coordination with an optimistic locking policy (see Section 119.26, "Configuring Locking Policy").

You can use cache invalidation to improve cache coordination efficiency. For more information, see Section 102.2.5, "Cache Invalidation".

For more information, see Section 102.3, "Cache Coordination".

## 102.2.7 Cache Isolation

Isolated client sessions provide a mechanism for disabling the shared server session cache. Any classes marked as isolated only cache objects relative to the life cycle of their client session. These classes never utilize the shared server session cache. This is the best mechanism to prevent caching as it is configured on a per-class basis allowing caching for some classes, and denying for others.

For more information, see Section 87.5, "Isolated Client Sessions".

## 102.2.8 Cache Locking and Transaction Isolation

By default, TopLink optimizes concurrency to minimize cache locking during read or write operations. Use the default TopLink transaction isolation configuration unless you have a very specific reason to change it.

For more information, see Section 115.15, "Database Transaction Isolation Levels".

## 102.2.9 Cache Optimization

Tune the TopLink cache for each class to help eliminate the need for distributed cache coordination. Always tune these settings before implementing cache coordination.

For more information, see Section 12.10, "Optimizing Cache".

## 102.3 Cache Coordination

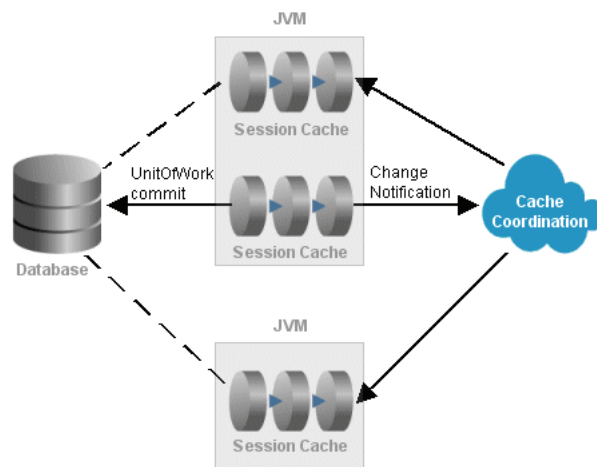
As [Figure 102–2](#) shows, cache coordination is a session feature that allows multiple, possibly distributed, instances of a session to broadcast object changes among each other so that each session's cache is either kept up-to-date or notified that the cache must update an object from the data source the next time it is read.

---

**Note:** You cannot use isolated client sessions (see [Section 87.5](#), "Isolated Client Sessions") with cache coordination.

---

**Figure 102–2** Cache Coordination



When sessions are distributed, that is, when an application contains multiple sessions (in the same JVM, in multiple JVMs, possibly on different servers), as long as the servers hosting the sessions are interconnected on the network, sessions can participate in cache coordination. Coordinated cache types that require discovery services also require the servers to support User Datagram Protocol (UDP) communication and multicast configuration (for more information, see [Section 102.3.2](#), "Coordinated Cache Architecture").

This section describes the following:

- [When to Use Cache Coordination](#)
- [Coordinated Cache Architecture](#)
- [Coordinated Cache Types](#)
- [Custom Coordinated Cache](#)

For more information, see [Section 103](#), "Configuring a Coordinated Cache".

### 102.3.1 When to Use Cache Coordination

Cache coordination can enhance performance and reduce the likelihood of stale data for applications that have the following characteristics:

- Changes are all being performed by the same Java application operating with multiple, distributed sessions
- Primarily read-based
- Regularly requests and updates the same objects

To maximize performance, avoid cache coordination for applications that do not have these characteristics. For more information about alternatives to cache coordination, see [Section 12.10, "Optimizing Cache"](#).

Cache coordination enhances performance mainly by avoiding data source access.

Cache coordination reduces the occurrence of stale data by increasing the likelihood that distributed caches are kept up-to-date with changes and are notified when one of the distributed caches must update an object from the data source the next time it is read.

Cache coordination reduces the number of optimistic lock exceptions encountered in a distributed architecture, and decreases the number of failed or repeated transactions in an application. However, cache coordination in no way eliminates the need for an effective locking policy. To effectively ensure working with up-to-date data, cache coordination must be used with optimistic or pessimistic locking. Oracle recommends that you use cache coordination with an optimistic locking policy (see [Section 119.26, "Configuring Locking Policy"](#)).

For other options to reduce the likelihood of stale data, see [Section 102.2.3, "Handling Stale Data"](#).

## 102.3.2 Coordinated Cache Architecture

TopLink provides coordinated cache implementations that perform discovery and message transport services using various technologies including the following:

- Java Message Service (JMS)—See [Section 102.3.3.1, "JMS Coordinated Cache"](#)
- Remote Method Invocation (RMI)—See [Section 102.3.3.2, "RMI Coordinated Cache"](#)
- Common Object Request Broker Architecture (CORBA)—See [Section 102.3.3.3, "CORBA Coordinated Cache"](#)

Regardless of the type of discovery and message transport you choose to use, the following are the principal objects that provide coordinated cache functionality:

- [Session](#)
- [Descriptor](#)
- [Unit of Work](#)

### 102.3.2.1 Session

When you enable a session for change propagation, the session provides discovery and message transport services using either JMS, RMI, CORBA, or Oracle Application Server Cluster.

Discovery services ensure that sessions announce themselves to other sessions participating in cache coordination. Discovery services use UDP communication and multicast configuration to monitor sessions as they join and leave the coordinated cache. All coordinated cache types (except JMS) require discovery services.

Message transport services allow the session to broadcast object change notifications to other sessions participating in cache coordination when a unit of work from this session commits a change.

### 102.3.2.2 Descriptor

You can configure how object changes are broadcast on a descriptor-by-descriptor basis. This lets you fine-tune the type of notification to make.

For example, for an object with few attributes, you can configure its descriptor to send object changes. For an object with many attributes, it may be more efficient to configure its descriptor so that the object is flagged as invalid (so that other sessions will know to update the object from the data source the next time it is read).

### 102.3.2.3 Unit of Work

Only changes committed by a unit of work are subject to propagation when cache coordination is enabled. The unit of work computes the appropriate change set based on the descriptor configuration of affected objects.

## 102.3.3 Coordinated Cache Types

You can create the following types of coordinated cache:

- [JMS Coordinated Cache](#)
- [RMI Coordinated Cache](#)
- [CORBA Coordinated Cache](#)

### 102.3.3.1 JMS Coordinated Cache

For a JMS coordinated cache, when a particular session's coordinated cache starts up, it uses its JNDI naming service information to locate and create a connection to the JMS server. The coordinated cache is ready when all participating sessions are connected to the same topic on the same JMS server. At this point, sessions can start sending and receiving object change messages. You can then configure all sessions that are participating in the same coordinated cache with the same JMS and JNDI naming service information.

Because you must supply the necessary information to connect to the JMS Topic, a JMS coordinated cache does not use a discovery service.

If you do use cache coordination, Oracle recommends that you use JMS cache coordination: JMS is robust, easy to configure, and provides efficient support for asynchronous change propagation.

For more information, see [Chapter 104, "Configuring a JMS Coordinated Cache"](#).

For more information on configuring JMS, see *Oracle Fusion Middleware Services Guide for Oracle Containers for Java EE*.

### 102.3.3.2 RMI Coordinated Cache

For an RMI coordinated cache, when a particular session's coordinated cache starts up, the session binds its connection in its naming service (either an RMI registry or JNDI), creates an announcement message (that includes its own naming service information), and broadcasts the announcement to its multicast group (see [Section 103.4, "Configuring a Multicast Group Address"](#) and [Section 103.5, "Configuring a Multicast Port"](#)). When a session that belongs to the same multicast group receives this announcement, it uses the naming service information in the announcement message to establish bidirectional connections with the newly announced session's coordinated cache. The coordinated cache is ready when all participating sessions are interconnected in this way, at which point sessions can start sending and receiving object change messages. You can then configure each session with naming information that identifies the host on which the session is deployed.

If you do use cache coordination, Oracle recommends that you use RMI cache coordination only if you require synchronous change propagation (see [Section 103.2, "Configuring the Synchronous Change Propagation Mode"](#)).

TopLink also supports cache coordination using RMI over the Internet Inter-ORB Protocol (IIOP). An RMI/IIOP coordinated cache uses RMI (and a JNDI naming service) for discovery and message transport services.

---

---

**Note:** If you use an RMI coordinated cache, Oracle recommends that you use RMI/IIOP only if absolutely necessary.

---

---

For more information, see [Chapter 105, "Configuring an RMI Coordinated Cache"](#).

### 102.3.3.3 CORBA Coordinated Cache

For a CORBA coordinated cache, when a particular session's coordinated cache starts up, the session binds its connection in JNDI, creates an announcement message (that includes its own JNDI naming service information), and broadcasts the announcement to its multicast group (see [Section 103.4, "Configuring a Multicast Group Address"](#) and [Section 103.5, "Configuring a Multicast Port"](#)). When a session that belongs to the same multicast group receives this announcement, it uses the naming service information in the announcement message to establish bidirectional connections with the newly announced session's coordinated cache. The coordinated cache is ready when all participating sessions are interconnected in this way, at which point, sessions can start sending and receiving object change messages. You can then configure each session with naming information that identifies the host on which the session is deployed.

Currently, TopLink provides support for the Sun Object Request Broker.

For more information on configuring a CORBA coordinated cache, see [Chapter 106, "Configuring a CORBA Coordinated Cache"](#).

## 102.3.4 Custom Coordinated Cache

Using the classes in `oracle.toplink.remotecommand` package, you can define your own coordinated cache for custom solutions. For more information, contact your TopLink support representative.

Once you have created the required cache coordination classes, for more information on configuring a user-defined coordinated cache, see [Chapter 107, "Configuring a Custom Coordinated Cache"](#).

## 102.4 Cache API

To configure the TopLink cache, you use the appropriate API in the following objects:

- [Object Identity API](#)
- [Cache Refresh API](#)
- [Cache Invalidation API](#)
- [Cache Coordination API](#)

### 102.4.1 Object Identity API

You configure object identity using the `ClassDescriptor` API summarized in [Example 102-1](#).

For more information, see [Section 119.12, "Configuring Cache Type and Size at the Descriptor Level"](#).

**Example 102–1 Object Identity ClassDescriptor API**

```
useCacheIdentityMap
useFullIdentityMap
useHardCacheWeakIdentityMap
useNoIdentityMap
useSoftCacheWeakIdentityMap
useWeakIdentityMap
useSoftIdentityMap
```

## 102.4.2 Cache Refresh API

You configure cache refresh using the `ClassDescriptor` API summarized in [Example 102–2](#).

**Example 102–2 Cache Refresh ClassDescriptor API**

```
alwaysRefreshCache
alwaysRefreshCacheOnRemote
disableCacheHits
disableCacheHitsOnRemote
onlyRefreshCacheIfNewerVersion
```

You can also configure cache refresh using the following API calls:

- `Session`: `refreshObject` method
- `DatabaseSession` and `UnitOfWork`: `refreshAndLockObject` methods
- `ObjectLevelReadQuery`: `refreshIdentityMapResult` and `refreshRemoteIdentityMapResult` methods

For more information, see [Section 119.9, "Configuring Cache Refreshing"](#).

## 102.4.3 Cache Invalidation API

You configure cache invalidation using `ClassDescriptor` methods `getCacheInvalidationPolicy` and `setCacheInvalidationPolicy` to configure an `oracle.toplink.descriptors.invalidation.CacheInvalidationPolicy`.

You can use any of the following `CacheInvalidationPolicy` instances:

- `DailyCacheInvalidationPolicy`: The object is automatically flagged as invalid at a specified time of day.
- `NoExpiryCacheInvalidationPolicy`: The object can only be flagged as invalid by explicitly calling `oracle.toplink.sessions.IdentityMapAccessor.invalidateObject`.
- `TimeToLiveCacheInvalidationPolicy`: The object is automatically flagged as invalid after a specified time period has elapsed since the object was read.

You can also configure cache invalidation using a variety of API calls accessible through the `Session`. The `oracle.toplink.sessions.IdentityMapAccessor` provides the following methods:

- `getRemainingValidTime`: Returns the remaining life of the specified object. This time represents the difference between the next expiry time of the object and its read time.
- `invalidateAll`: Sets all objects for all classes to be invalid in `TopLink` identity maps.



- `invalidateClass(Class klass)` and `invalidateClass(Class klass, boolean recurse)`: Set all objects of a specified class to be invalid in TopLink identity maps.
- `invalidateObject(Object object)`, `invalidateObject(Record rowWithPrimarykey, Class klass)` and `invalidateObject(Vector primaryKey, Class klass)`: Set an object to be invalid in TopLink identity maps.
- `invalidateObjects(Expression selectionCriteria)` and `invalidateObjects(Vector collection)`: Set all objects from the specified Expression/collection to be invalid in TopLink identity maps.
- `isValid(Record recordContainingPrimarykey, Class theClass)`, `isValid(Object object)` and `isValid(java.util.Vector primaryKey, Class theClass)`: Return true if the object is valid in TopLink identity maps.

For more information, see the following:

- [Section 117.13, "Configuring Cache Expiration at the Project Level"](#)
- [Section 119.16, "Configuring Cache Expiration at the Descriptor Level"](#)
- [Section 111.13.2, "How to Configure Cache Expiration at the Query Level"](#)

## 102.4.4 Cache Coordination API

You configure cache coordination using the `Session` methods summarized in [Example 102-3](#).

You configure how object changes are propagated using the `ClassDescriptor` methods summarized in [Example 102-4](#).

For more information, see [Section 103.1, "Configuring Common Coordinated Cache Options"](#).

### **Example 102-3 Cache Coordination Session API**

```

Session.getCommandManager().
    setShouldPropagateAsynchronously(boolean)
Session.getCommandManager().getDiscoveryManager().
    setAnnouncementDelay()
    setMulticastGroupAddress()
    setMulticastPort()
    setPacketTimeToLive()
Session.getCommandManager().getTransportManager().
    setEncryptedPassword()
    setInitialContextFactoryName()
    setLocalContextProperties(Hashtable)
    setNamingServiceType() passing in one of:
        TransportManager.JNDI_NAMING_SERVICE
        TransportManager.REGISTRY_NAMING_SERVICE
    setPassword()
    setRemoteContextProperties(Hashtable)
    setShouldRemoveConnectionOnError()
    setUsername()

```

### **Example 102-4 Cache Coordination ClassDescriptor API**

```

setCacheSynchronizationType() passing in one of:
    ClassDescriptor.DO_NOT_SEND_CHANGES

```

```
ClassDescriptor.INVALIDATE_CHANGED_OBJECTS  
ClassDescriptor.SEND_NEW_OBJECTS_WITH_CHANGES  
ClassDescriptor.SEND_OBJECT_CHANGES
```

---



---

## Configuring a Coordinated Cache

This chapter describes how to configure a TopLink coordinated cache.

This chapter includes the following sections:

- [Configuring Common Coordinated Cache Options](#)
- [Configuring the Synchronous Change Propagation Mode](#)
- [Configuring a Service Channel](#)
- [Configuring a Multicast Group Address](#)
- [Configuring a Multicast Port](#)
- [Configuring a Naming Service Type](#)
- [Configuring JNDI Naming Service Information](#)
- [Configuring RMI Registry Naming Service Information](#)
- [Configuring an Announcement Delay](#)
- [Configuring Connection Handling](#)
- [Configuring Context Properties](#)
- [Configuring a Packet Time-to-Live](#)

**Table 103–1** *Configuring TopLink Coordinated Caches*

If you are configuring a...	See...
JMS Coordinated Cache	Chapter 104, "Configuring a JMS Coordinated Cache"
RMI Coordinated Cache	Chapter 105, "Configuring an RMI Coordinated Cache"
CORBA Coordinated Cache	Chapter 106, "Configuring a CORBA Coordinated Cache"
Custom Coordinated Cache	Chapter 107, "Configuring a Custom Coordinated Cache"

For more information, see [Section 102.3, "Cache Coordination"](#).

### 103.1 Configuring Common Coordinated Cache Options

[Table 103–1](#) lists the configurable options shared by two or more TopLink coordinated cache types. In addition to the configurable options described here, you must also configure the options described for the specific [Coordinated Cache Types](#), as shown in [Table 103–2](#).

**Table 103–2 Common Coordinated Cache Options**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Cache coordination change propagation at the descriptor level (see Section 119.15, "Configuring Cache Coordination Change Propagation at the Descriptor Level")	✓	✓	✓
Synchronous change propagation mode (see Section 103.2, "Configuring the Synchronous Change Propagation Mode")	✓	✓	✓
Service channel (see Section 103.3, "Configuring a Service Channel")	✓	✓	✓
Multicast group address (see Section 103.4, "Configuring a Multicast Group Address")	✓	✓	✓
Multicast port (see Section 103.5, "Configuring a Multicast Port")	✓	✓	✓
Naming service type (see Section 103.6, "Configuring a Naming Service Type")	✓	✓	✓
Announcement delay (see Section 103.9, "Configuring an Announcement Delay")	✓	✓	✓
Connection handling (see Section 103.10, "Configuring Connection Handling")	✓	✓	✓
Context properties (see Section 103.11, "Configuring Context Properties")	✓	✓	✓
Packet time-to-live (see Section 103.12, "Configuring a Packet Time-to-Live")	✓	✓	✓

## 103.2 Configuring the Synchronous Change Propagation Mode

You can configure whether the coordinated cache should propagate object changes asynchronously or synchronously.

Table 103–3 summarizes which coordinated caches support propagation mode configuration.

**Table 103–3 Coordinated Cache Support for Propagation Mode Configuration**

Coordinated Cache	How to Use Oracle JDeveloper	How to Configure the Synchronous Change Propagation Mode Using TopLink Workbench	How to Configure the Synchronous Change Propagation Mode Using Java
JMS Coordinated Cache (asynchronous only)			
RMI Coordinated Cache	✓	✓	✓
CORBA Coordinated Cache	✓	✓	✓
Custom Coordinated Cache			

Synchronous propagation mode forces the session to wait for an acknowledgement before sending the next object change notification: this reduces the likelihood of stale data at the expense of performance.

Asynchronous propagation mode allows the session to create separate threads to propagate changes to remote servers. TopLink returns control to the client immediately after the local commit operation, whether or not the changes merge

successfully on the remote servers. This offers superior performance for applications that are somewhat tolerant of stale data.

For more information, [Section 102.2.3, "Handling Stale Data"](#).

### 103.2.1 How to Configure the Synchronous Change Propagation Mode Using TopLink Workbench

To specify the change propagation mode, use this procedure:

1. Select a session or session broker in the **Navigator**. Its properties appear in the Editor.
2. Click the **Cache Coordination** tab. The Cache Coordination tab appears.
3. Ensure the **Enable Cache Coordination** option is selected, then select the appropriate coordinated cache **Type** (see [Table 103–3](#)). The cache coordination options appear on the tab.

**Figure 103–1** Cache Coordination Tab, Synchronous Field

The screenshot shows the 'Cache Coordination' tab in the TopLink Workbench. The 'Enable Cache Coordination' checkbox is checked. The 'Type' dropdown menu is set to 'RMI'. The 'Channel' text field contains 'TopLinkCommandChannel'. The 'Multicast Group Address' text field contains '226.10.12.64'. The 'Multicast Port' spinner is set to '3,121'. The 'Packet Time to Live' spinner is set to '2'. The 'Announcement Delay' spinner is set to '1,000'. Below these fields, the 'Remove Connection On Error' checkbox is checked. The 'Synchronous' radio button is selected and circled in red. The 'Registry Naming Service' radio button is unselected. The 'JNDI Naming Service' radio button is selected. The 'URL' text field is empty. The 'User Name' text field contains 'admin'. The 'Password' text field contains 'password'. The 'Initial Context Factory' text field contains 'com.evermind.server.rmi.RMIInitialContextFactory'. There is a 'Browse...' button next to the 'Initial Context Factory' field. A 'Properties...' button is at the bottom left.

4. Select the **Synchronous** option to use synchronous change propagation. Do not select this option to use asynchronous change propagation.

### 103.2.2 How to Configure the Synchronous Change Propagation Mode Using Java

Use the `oracle.toplink.remotecommand.RemoteCommandManager` method `setShouldPropagateAsynchronously` to define whether changes should be propagated synchronously or asynchronously for this coordinated cache.

For more information, see [Section 102.4.4, "Cache Coordination API"](#).

## 103.3 Configuring a Service Channel

The **service channel** is the name of the TopLink coordinated cache channel to which sessions subscribe in order to participate in the same coordinated cache. Such sessions

use the service channel to exchange messages with each other. Messages sent on other service channels will not be exchanged with this coordinated cache.

Table 103–4 summarizes which coordinated caches support service channel configuration.

**Table 103–4 Coordinated Cache Support for Service Channel Configuration**

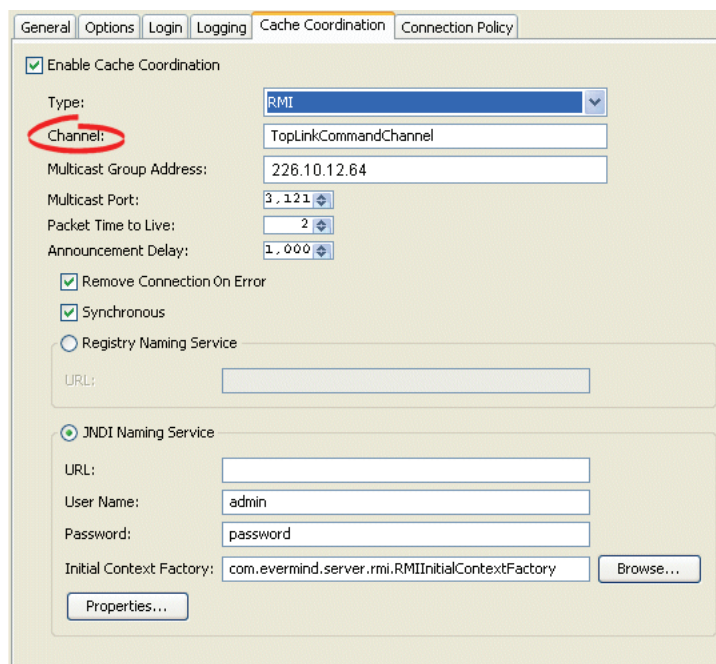
Coordinated Cache	How to Use Oracle JDeveloper	How to Configure a Service Channel Using TopLink Workbench	How to Configure a Service Channel Using Java
JMS Coordinated Cache			
RMI Coordinated Cache	✓	✓	✓
CORBA Coordinated Cache	✓	✓	✓
Custom Coordinated Cache	✓	✓	✓

### 103.3.1 How to Configure a Service Channel Using TopLink Workbench

To specify the service channel, use this procedure:

1. Select a session or session broker in the **Navigator**. Its properties appear in the Editor.
2. Click the **Cache Coordination** tab. The Cache Coordination tab appears.
3. Ensure the **Enable Cache Coordination** option is selected, then select the appropriate coordinated cache **Type** (see Table 103–4). The cache coordination options appear on the tab.

**Figure 103–2 Cache Coordination Tab, Channel Field**



4. In the **Channel** field, enter the name of the service channel for this coordinated cache.

### 103.3.2 How to Configure a Service Channel Using Java

Use the `oracle.toplink.remotecommand.RemoteCommandManager` method `setChannel` to set the name of the service channel for this coordinated cache.

For more information, see [Section 102.4.4, "Cache Coordination API"](#).

## 103.4 Configuring a Multicast Group Address

A multicast group address is an Internet Protocol (IP) address in the range 224.0.0.0 to 239.255.255.255 that identifies the members of an IP multicast group. To efficiently broadcast the same message to all members of an IP multicast group, you configure each recipient with the same multicast group address and send the message to that address.

[Table 103–5](#) summarizes which coordinated caches support multicast group address configuration.

**Table 103–5** *Coordinated Cache Support for Multicast Group Address Configuration*

Coordinated Cache	How to Use Oracle JDeveloper	How to Configure a Multicast Group Address Using TopLink Workbench	How to Configure a Multicast Group Address Using Java
<a href="#">JMS Coordinated Cache</a>			
<a href="#">RMI Coordinated Cache</a>	✓	✓	✓
<a href="#">CORBA Coordinated Cache</a>	✓	✓	✓
<a href="#">Custom Coordinated Cache</a>			

---

**Note:** Ensure your host and network are configured to support multicast operation before configuring this option.

---

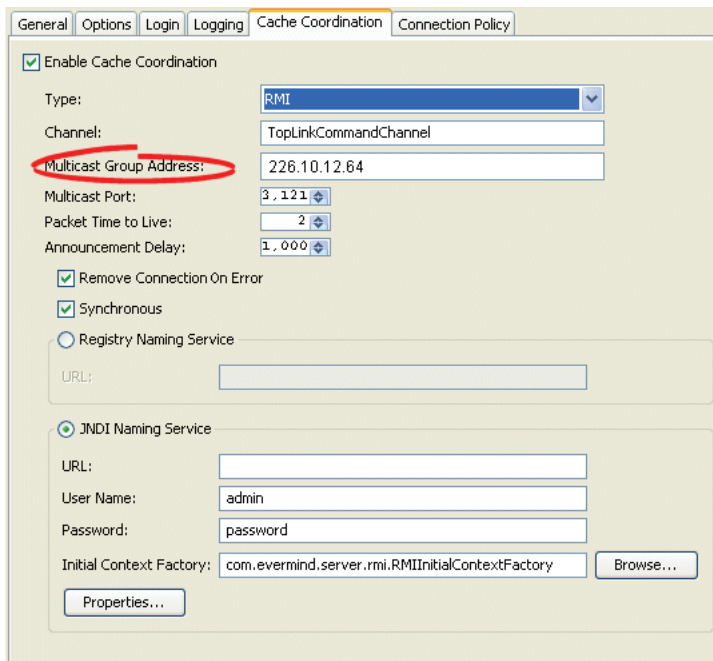
In addition to configuring the multicast group address, you must also configure the multicast port (see [Section 103.5, "Configuring a Multicast Port"](#)) for the coordinated cache types shown in [Table 103–5](#).

### 103.4.1 How to Configure a Multicast Group Address Using TopLink Workbench

To specify the multicast group address, use this procedure:

1. Select a session or session broker in the **Navigator**. Its properties appear in the Editor.
2. Click the **Cache Coordination** tab. The Cache Coordination tab appears.
3. Ensure the **Enable Cache Coordination** option is selected, then select the appropriate coordinated cache **Type** (see [Table 103–5](#)). The cache coordination options appear on the tab.

**Figure 103–3 Cache Coordination Tab, Multicast Group Address Field**



4. Enter the multicast group address in the range 224.0.0.0 to 239.255.255.255 to subscribe this session to a given coordinated cache.

### 103.4.2 How to Configure a Multicast Group Address Using Java

Use the `oracle.toplink.remotecommand.DiscoveryManager` method `setMulticastGroupAddress` to subscribe this session to a given coordinated cache.

---

**Note:** Ensure that the address falls in the range of 224.0.0.0 to 239.255.255.255

---

For more information, see [Section 102.4.4, "Cache Coordination API"](#).

## 103.5 Configuring a Multicast Port

The multicast port is the port on which multicast messages are received. Members of a multicast group (see [Section 103.4, "Configuring a Multicast Group Address"](#)) rely on messages broadcast to their multicast group address to communicate with one another.

[Table 103–6](#) summarizes which coordinated caches support multicast port configuration.

**Table 103–6 Coordinated Cache Support for Multicast Port Configuration**

Coordinated Cache	How to Use Oracle JDeveloper	How to Configure a Multicast Port Using TopLink Workbench	How to Configure a Multicast Port Using Java
<a href="#">JMS Coordinated Cache</a>			



**Table 103–6 (Cont.) Coordinated Cache Support for Multicast Port Configuration**

Coordinated Cache	How to Use Oracle JDeveloper	How to Configure a Multicast Port Using TopLink Workbench	How to Configure a Multicast Port Using Java
RMI Coordinated Cache	✓	✓	✓
CORBA Coordinated Cache	✓	✓	✓
Custom Coordinated Cache			

---

**Note:** Ensure your host and network are configured to support multicast operation before configuring this option

---

### 103.5.1 How to Configure a Multicast Port Using TopLink Workbench

To specify the multicast port, use this procedure:

1. Select a session or session broker in the **Navigator**. Its properties appear in the Editor.
2. Click the **Cache Coordination** tab. The Cache Coordination tab appears.
3. Ensure the **Enable Cache Coordination** option is selected, then select the appropriate coordinated cache **Type** (see [Table 103–6](#)). The cache coordination options appear on the tab.

**Figure 103–4 Cache Coordination Tab, Multicast Port Field**

The screenshot shows the 'Cache Coordination' tab in the TopLink Workbench. The 'Enable Cache Coordination' checkbox is checked. The 'Type' dropdown is set to 'RMI'. The 'Channel' is 'TopLinkCommandChannel'. The 'Multicast Group Address' is '226.10.12.64'. The 'Multicast Port' is '3,121', which is circled in red. Other settings include 'Packet Time to Live' at 2 and 'Announcement Delay' at 1,000. There are also checkboxes for 'Remove Connection On Error' and 'Synchronous', and radio buttons for 'Registry Naming Service' and 'JNDI Naming Service'. The 'JNDI Naming Service' section includes fields for 'URL', 'User Name' (admin), 'Password' (password), and 'Initial Context Factory' (com.evermind.server.rmi.RMIInitialContextFactory).

4. Enter the multicast port on which messages broadcast to the multicast group address are received.

## 103.5.2 How to Configure a Multicast Port Using Java

Use the `oracle.toplink.remotecommand.DiscoveryManager` method `setMulticastPort` to define the multicast port on which messages broadcast to the multicast group address are to be received.

For more information, see [Section 102.4.4, "Cache Coordination API"](#).

## 103.6 Configuring a Naming Service Type

The session's message transport service uses a naming service when it looks up connections to other sessions in the coordinated cache. You can configure the message transport service to look up remote objects using an RMI registry or Java Naming and Directory Interface (JNDI). By default, JNDI is used.

[Table 103–7](#) summarizes which coordinated caches support naming service configuration.

**Table 103–7 Coordinated Cache Support for Naming Service Configuration**

Coordinated Cache	JNDI Naming Service	RMI Registry Naming Service
<a href="#">JMS Coordinated Cache</a>	✓	
<a href="#">RMI Coordinated Cache</a>	✓	✓
<a href="#">CORBA Coordinated Cache</a>	✓	
<a href="#">Custom Coordinated Cache</a>		

For information, see the following:

- [Section 103.8, "Configuring RMI Registry Naming Service Information"](#)
- [Section 103.7, "Configuring JNDI Naming Service Information"](#)

## 103.7 Configuring JNDI Naming Service Information

The session's message transport service uses a naming service when it looks up connections to other sessions in the coordinated cache. If you choose to use a JNDI naming service, you must configure JNDI naming service information.

[Table 103–8](#) summarizes which coordinated caches support JNDI naming service configuration.

**Table 103–8 Coordinated Cache Support for JNDI Naming Service Configuration**

Coordinated Cache	How to Use Oracle JDeveloper	How to Configure JNDI Naming Service Information Using TopLink Workbench	How to Configure JNDI Naming Service Information Using Java
<a href="#">JMS Coordinated Cache</a>	✓	✓	✓
<a href="#">RMI Coordinated Cache</a>	✓	✓	✓
<a href="#">CORBA Coordinated Cache</a>	✓	✓	✓
<a href="#">Custom Coordinated Cache</a>			

TopLink uses JNDI naming service information differently, depending on the type of coordinated cache.

For a JMS coordinated cache, when a particular session's coordinated cache starts up, it uses its JNDI naming service information to locate and create a connection to the JMS server. The coordinated cache is ready when all participating sessions are connected to the JMS server. At this point, sessions can start sending and receiving object change messages. You can then configure all sessions that are participating in the same coordinated cache with the same JNDI naming service information.

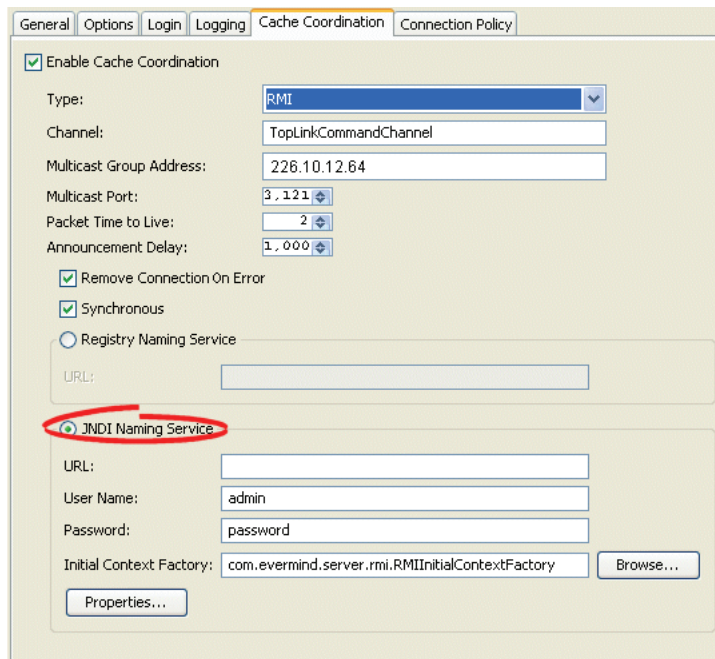
For an RMI or CORBA coordinated cache, when a particular session's coordinated cache starts up, the session binds its connection in JNDI, creates an announcement message (that includes its own JNDI naming service information), and broadcasts the announcement to its multicast group (see [Section 103.4, "Configuring a Multicast Group Address"](#) and [Section 103.5, "Configuring a Multicast Port"](#)). When a session that belongs to the same multicast group receives this announcement, it uses the JNDI naming service information in the announcement message to establish bidirectional connections with the newly announced session's coordinated cache. The coordinated cache is ready when all participating sessions are interconnected in this way, at which point, sessions can start sending and receiving object change messages. You can then configure each session with JNDI naming information that identifies the host on which the session is deployed.

### 103.7.1 How to Configure JNDI Naming Service Information Using TopLink Workbench

To specify the sessions's JNDI naming service, use this procedure:

1. Select a session or session broker in the **Navigator**. Its properties appear in the Editor.
2. Click the **Cache Coordination** tab. The Cache Coordination tab appears.
3. Ensure the **Enable Cache Coordination** option is selected, then select the appropriate coordinated cache **Type** (see [Table 103-8](#)). The cache coordination options appear on the tab.

**Figure 103–5 Cache Coordination Tab, JNDI Naming Service Options**



Use the following information to enter data in the fields of the Cache Coordination tab to configure the naming service options:

Field	Description
URL	<p>The location of the JNDI naming service.</p> <p><b>For a JMS coordinated cache:</b> If you are using the Oracle Containers for Java EE (OC4J) JNDI naming service and all the hosts in your coordinated cache can communicate using the OC4J proprietary RMI protocol ORMI, use a URL similar to the following:</p> <pre>ormi://&lt;JMS-host-IP&gt;:&lt;JMS-host-port&gt;</pre> <p>where <code>JMS-host-IP</code> is the IP address of the host on which the JMS service provider is running, and <code>JMS-host-port</code> is the port on which the JMS service provider is listening for JMS requests.</p> <p><b>For an RMI or CORBA coordinated cache:</b> If you are using the OC4J JNDI naming service and all the hosts in your coordinated cache can communicate using the OC4J proprietary RMI protocol ORMI on OC4J default port 23791, use a URL similar to the following:</p> <pre>ormi://&lt;session-host-IP&gt;:23791</pre> <p>where <code>session-host-IP</code> is the IP address of the host on which this session is deployed.</p>
Username	<p>The user name required to log in to the JNDI naming service.</p> <p>The value you enter defines the <code>Context.SECURITY_PRINCIPAL</code> environment property.</p>

Field	Description
<b>Password</b>	<p>The plain text (unencrypted) password required to log in to the JNDI naming service. The password appears in plain text in TopLink Workbench, but it is encrypted when written to the <code>sessions.xml</code> file.</p> <p>The value you enter defines the <code>Context.SECURITY_CREDENTIALS</code> environment property.</p>
<b>Initial Context Factory</b>	<p>The name of the factory class, provided by your JNDI naming service provider, that implements the <code>javax.naming.spi.InitialContextFactory</code> interface. This factory class is used to create a <code>javax.naming.Context</code> instance that can access the JNDI naming service provider's context implementation.</p> <p>The value you enter defines the <code>Context.INITIAL_CONTEXT_FACTORY</code> environment property.</p>
<b>Properties</b>	<p>The JNDI context properties.</p> <p>Click <b>Properties</b> to configure custom JNDI context properties (see <a href="#">Section 103.11, "Configuring Context Properties"</a>).</p>

## 103.7.2 How to Configure JNDI Naming Service Information Using Java

Use the `oracle.toplink.remotecommand.TransportManager` method `setNamingServiceType` as follows:

```
setNamingServiceType(TransportManager.JNDI_NAMING_SERVICE)
```

Then use the following `TransportManager` methods to configure the JNDI naming service options:

- `setUserName`—Set the user name required to log in to the JNDI naming service. The value you enter defines the `Context.SECURITY_PRINCIPAL` environment property.
- `setPassword`—Set the unencrypted password required to log in to the JNDI naming service. The value you enter defines the `Context.SECURITY_CREDENTIALS` in the cached context properties.
- `setEncryptedPassword`—Set the encrypted password required to log in to the JNDI naming service. The value you enter defines the `Context.SECURITY_CREDENTIALS` in the cached context properties.
- `setInitialContextFactoryName`—The name of the factory class, provided by your JNDI naming service provider, that implements the `javax.naming.spi.InitialContextFactory` interface. This factory class is used to create a `javax.naming.Context` instance that can access the JNDI naming service provider's context implementation. The value you enter defines the `Context.INITIAL_CONTEXT_FACTORY` in the cached context properties.
- `setLocalContextProperties`—Set the properties that will be used to create the initial context for local JNDI access.

Do not forget to specify the location of the JNDI naming service by providing its URL. Consider the following:

- For a JMS coordinated cache, if you are using the OC4J JNDI naming service and all the hosts in your coordinated cache can communicate using the OC4J proprietary RMI protocol ORMI, use a URL similar to the following:

```
ormi://<JMS-host-IP>:<JMS-host-port>
```

where `JMS-host-IP` is the IP address of the host on which the JMS service provider is running, and `JMS-host-port` is the port on which the JMS service provider is listening for JMS requests.

- For an RMI or CORBA coordinated cache, if you are using the OC4J JNDI naming service and all the hosts in your coordinated cache can communicate using the OC4J proprietary RMI protocol ORMI on OC4J default port 23791, use a URL similar to the following:

```
ormi://<session-host-IP>:23791
```

where `session-host-IP` is the IP address of the host on which this session is deployed.

Note that the default protocol value is "ormi", and the default port value is "23791". You can also use the `TransportManager.DEFAULT_URL_PROTOCOL` and `DEFAULT_URL_PORT`.

For more information, see [Section 102.4.4, "Cache Coordination API"](#).

## 103.8 Configuring RMI Registry Naming Service Information

The session's message transport service uses a naming service when it looks up connections to other sessions in the coordinated cache. If you choose to use an RMI registry naming service, you can configure RMI registry naming service options.

[Table 103–8](#) summarizes which coordinated caches support RMI registry naming service configuration.

**Table 103–9 Coordinated Cache Support for RMI Registry Naming Service Configuration**

Coordinated Cache	How to Use Oracle JDeveloper	How to Configure RMI Registry Naming Service Information Using TopLink Workbench	How to Configure RMI Registry Naming Service Information Using Java
<a href="#">JMS Coordinated Cache</a>			
<a href="#">RMI Coordinated Cache</a>	✓	✓	✓
<a href="#">CORBA Coordinated Cache</a>			
<a href="#">Custom Coordinated Cache</a>			

For an RMI coordinated cache, when a particular session's coordinated cache starts up, the session binds its connection in its RMI registry, creates an announcement message (that includes its own naming service information), and broadcasts the announcement to its multicast group (see [Section 103.4, "Configuring a Multicast Group Address"](#) and [Section 103.5, "Configuring a Multicast Port"](#)). When a session that belongs to the same multicast group receives this announcement, it uses the JNDI naming service information in the announcement message to establish bidirectional connections with the newly announced session's coordinated cache. The coordinated cache is ready when all participating sessions are interconnected in this way, at which point, sessions can start sending and receiving object change messages. You can then configure each session with RMI registry naming information that identifies the host on which the session is deployed.

## 103.8.1 How to Configure RMI Registry Naming Service Information Using TopLink Workbench

To specify the sessions's registry naming service, use this procedure:

1. Select a session or session broker in the **Navigator**. Its properties appear in the Editor window.
2. Click the **Cache Coordination** tab. The Cache Coordination tab appears.
3. Ensure the **Enable Cache Coordination** option is selected, then select the appropriate coordinated cache **Type** (see [Table 103-9](#)). The cache coordination options appear on the tab.

**Figure 103-6 Cache Coordination Tab, Naming Service Options**

The screenshot shows the 'Cache Coordination' tab in the TopLink Workbench. The 'Enable Cache Coordination' checkbox is checked. The 'Type' dropdown is set to 'RMI'. The 'Channel' is 'TopLinkCommandChannel'. The 'Multicast Group Address' is '226.10.12.64'. The 'Multicast Port' is '3,121'. The 'Packet Time to Live' is '2'. The 'Announcement Delay' is '1,000'. The 'Remove Connection On Error' and 'Synchronous' checkboxes are checked. The 'Registry Naming Service' radio button is selected and circled in red. Below it is a 'URL:' field. The 'JNDI Naming Service' radio button is unselected. Below it are fields for 'URL:', 'User Name:' (admin), 'Password:' (password), and 'Initial Context Factory:' (com.evermind.server.rmi.RMIInitialContextFactory) with a 'Browse...' button. A 'Properties...' button is at the bottom.

Use the following information to configure the naming service options:

Field	Description
URL	Assuming that you are using the OC4J JNDI naming service and that all the hosts in your coordinated cache can communicate using the OC4J proprietary RMI protocol ORMI on OC4J default port 23791, use a URL similar to the following:  <code>ormi://&lt;session-host-IP&gt;:23791</code>  where <code>session-host-IP</code> is the IP address of the host on which this session is deployed.

## 103.8.2 How to Configure RMI Registry Naming Service Information Using Java

Use the `oracle.toplink.remotecommand.TransportManager` method `setNamingServiceType` as follows:

```
setNamingServiceType(TransportManager.REGISTRY_NAMING_SERVICE)
```

Then specify the location of the JNDI naming service by providing its URL. Consider the following:

For an RMI or CORBA coordinated cache, if you are using the OC4J JNDI naming service and all the hosts in your coordinated cache can communicate using the OC4J proprietary RMI protocol ORMI on OC4J default port 23791, use a URL similar to the following:

```
ormi://<session-host-IP>:23791
```

where `session-host-IP` is the IP address of the host on which this session is deployed.

Note that the default protocol value is "ormi", and the default port value is "23791". You can also use the `TransportManager.DEFAULT_URL_PROTOCOL` and `DEFAULT_URL_PORT` constants.

For more information, see [Section 102.4.4, "Cache Coordination API"](#).

## 103.9 Configuring an Announcement Delay

Use the announcement delay option to set the amount of time (in milliseconds) that a session should wait between the time that it is available and the time that it broadcasts its announcement message to the members of the coordinated cache. This additional delay may be necessary to give some systems more time to post their connections into the naming service (see [Section 103.6, "Configuring a Naming Service Type"](#)).

[Table 103–10](#) summarizes which coordinated caches support announcement delay configuration.

**Table 103–10 Coordinated Cache Support for Announcement Delay Configuration**

Coordinated Cache	How to Use Oracle JDeveloper	How to Configure an Announcement Delay Using TopLink Workbench	How to Use Java
JMS Coordinated Cache			
RMI Coordinated Cache	✓	✓	✓
CORBA Coordinated Cache	✓	✓	✓
Custom Coordinated Cache			

In addition to announcement delay, you may also need to consider packet time-to-live configuration (see [Section 103.12, "Configuring a Packet Time-to-Live"](#)).

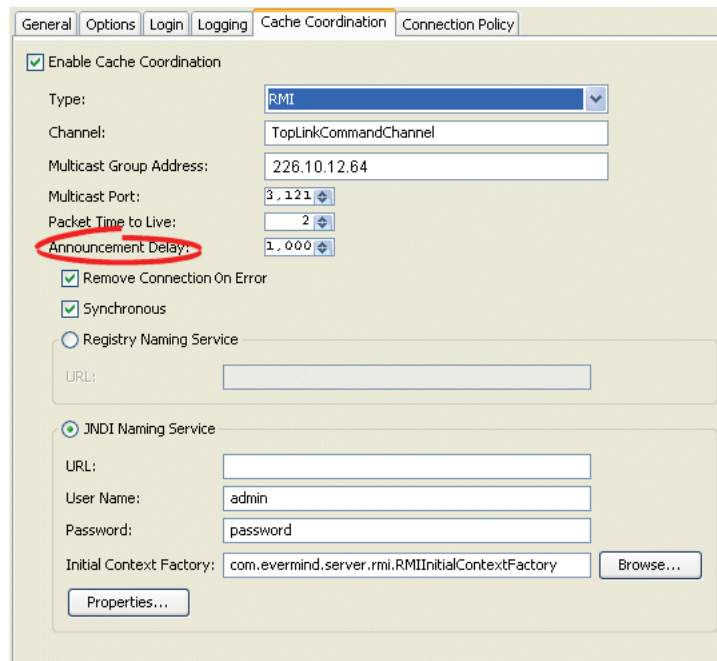
### 103.9.1 How to Configure an Announcement Delay Using TopLink Workbench

To specify the announcement delay (in milliseconds) for an RMI coordinated cache, use this procedure:

1. Select a session or session broker in the **Navigator**. Its properties appear in the Editor.
2. Click the **Cache Coordination** tab. The Cache Coordination tab appears.
3. Ensure the **Enable Cache Coordination** option is selected, then select the appropriate coordinated cache **Type** (see [Table 103–10](#)). The cache coordination options appear on the tab.



**Figure 103–7 Cache Coordination Tab, Announcement Delay Field**



4. Select the amount of time (in milliseconds) that this session should wait between the time that it is available and the time that it broadcasts its announcement message to the members of the coordinated cache.

### 103.9.2 How to Configure an Announcement Delay Using Java

Use the `oracle.toplink.remotecommand.DiscoveryManager` method `setAnnouncementDelay` to select the amount of time (in milliseconds) that this session should wait between the time that it is available and the time that it broadcasts its announcement message to the members of the coordinated cache

For more information, see [Section 102.4.4, "Cache Coordination API"](#).

## 103.10 Configuring Connection Handling

The session’s transport manager creates connections to the various members of the coordinated cache. If a communication error occurs on one of these connections, you can configure the session to either ignore the error or remove the connection.

[Table 103–11](#) summarizes which coordinated caches support connection handling configuration.

**Table 103–11 Coordinated Cache Support for Connection Handling Configuration**

Coordinated Cache	How to Use Oracle JDeveloper	How to Configure Connection Handling Using TopLink Workbench	How to Configure Connection Handling Using Java
JMS Coordinated Cache	✓	✓	✓
RMI Coordinated Cache	✓	✓	✓

**Table 103–11 (Cont.) Coordinated Cache Support for Connection Handling**

Coordinated Cache	How to Use Oracle JDeveloper	How to Configure Connection Handling Using TopLink Workbench	How to Configure Connection Handling Using Java
CORBA Coordinated Cache	✓	✓	✓
Custom Coordinated Cache	✓	✓	✓

If you configure the session to remove the connection on error, the next time the session tries to communicate with that coordinated cache member, it will construct a new connection.

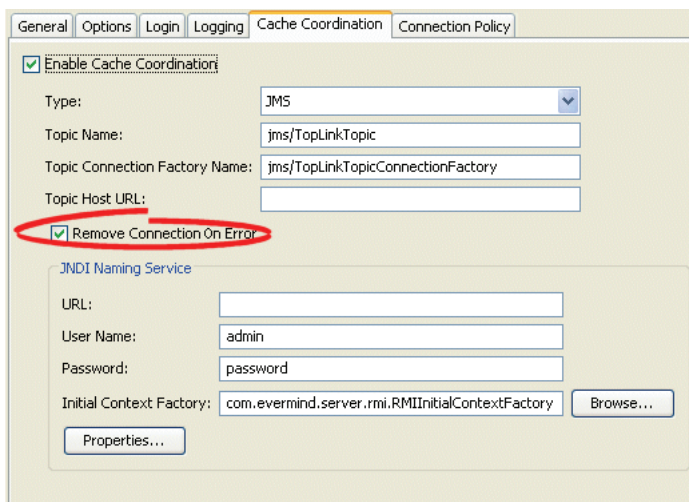
If you configure the session to ignore the error, the next time the session tries to communicate with that coordinated cache member, it will continue to use the same connection.

### 103.10.1 How to Configure Connection Handling Using TopLink Workbench

To specify how TopLink handles session connections in the event of an error, use this procedure:

1. Select a session or session broker in the **Navigator**. Its properties appear in the Editor.
2. Click the **Cache Coordination** tab. The Cache Coordination tab appears.
3. Ensure the **Enable Cache Coordination** option is selected, then select the appropriate coordinated cache **Type** (see [Table 103–11](#)). The cache coordination options appear on the tab.

**Figure 103–8 Cache Coordination Tab, Remove Connection on Error Option**



4. Select the **Remove Connection on Error** option to configure the session to remove the data source connection in the event of an error.

## 103.10.2 How to Configure Connection Handling Using Java

Use the `oracle.toplink.remotecommand.TransportManager` method `setShouldRemoveConnectionOnError` to configure the session to remove the data source connection if an error occurs.

For more information, see [Section 102.4.4, "Cache Coordination API"](#).

## 103.11 Configuring Context Properties

When you configure a coordinated cache to use a JNDI naming service (see [Section 103.6, "Configuring a Naming Service Type"](#)), you can add new environment properties to the environment of the initial JNDI context.

[Table 103–12](#) summarizes which coordinated caches support context properties.

**Table 103–12 Coordinated Cache Support for Context Properties**

Coordinated Cache	How to Use Oracle JDeveloper	How to Configure Context Properties Using TopLink Workbench	How to Configure Context Properties Using Java
JMS Coordinated Cache	✓	✓	✓
RMI Coordinated Cache <sup>1</sup>	✓	✓	✓
CORBA Coordinated Cache <sup>1</sup>	✓	✓	✓
Custom Coordinated Cache			

<sup>1</sup> When JNDI naming service is selected (see [Section 103.6, "Configuring a Naming Service Type"](#)).

Using TopLink Workbench, TopLink uses the new environment properties you add to create the initial context for both local and remote JNDI access.

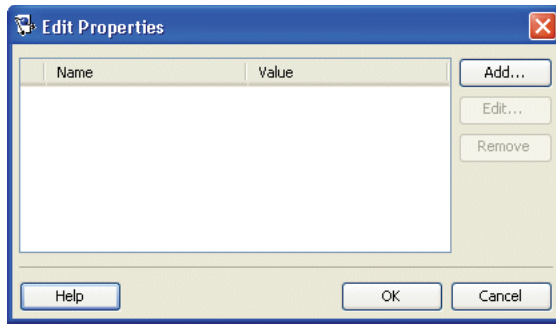
Using Java, you can configure different properties for local and remote JNDI access using a session customizer class to call `TransportManager` methods `setLocalContextProperties` and `setRemoteContextProperties` (for more information, see [Section 89.8, "Configuring a Session Customizer Class"](#)).

### 103.11.1 How to Configure Context Properties Using TopLink Workbench

To define JNDI context properties, use this procedure:

1. Select a session or session broker in the **Navigator**. Its properties appear in the Editor.
2. Ensure the **Enable Cache Coordination** option is selected, then select the appropriate coordinated cache **Type** (see [Table 103–12](#)). The cache coordination options appear on the tab.
3. Ensure the **JNDI Naming Service** option is selected. See [Section 103.6, "Configuring a Naming Service Type"](#).
4. In the JNDI Naming Service area, click **Properties**. The Edit Properties dialog box appears.

**Figure 103–9 Edit Properties Dialog Box**



Use this table to enter data in the following fields on the dialog box.

Field	Description
Name	The name of the property.
Value	The value of the property.

To change (or delete) an existing property, select the property and click **Edit** (or **Remove**).

### 103.11.2 How to Configure Context Properties Using Java

Use the `oracle.toplink.remotecommand.TransportManager` method `setLocalContextProperties` to define a `Hashtable` of the JNDI context properties that will be used to create the initial context for the local JNDI access. Note that the "dedicated.connection" is the default key with the default value of "true".

For more information, see [Section 102.4.4, "Cache Coordination API"](#).

## 103.12 Configuring a Packet Time-to-Live

The **packet time-to-live** is the number of hops that session data **packets** can take before expiring. The default is 2. This allows for a **hub** and an interface card, and prevents the data packets from leaving the local network. If sessions are hosted on different local area networks (LANs) that are part of wide area network (WAN), or if a firewall configuration prevents it, the announcement sent by one session may not reach the other sessions in the coordinated cache. In this case, consult your network administrator for the correct time-to-live value.

[Table 103–13](#) summarizes which coordinated caches support packet time-to-live configuration.

**Table 103–13 Coordinated Cache Support for Packet Time-to-Live Configuration**

Coordinated Cache	How to Use Oracle JDeveloper	How to Configure a Packet Time-to-Live Using TopLink Workbench	How to Use Java
<a href="#">JMS Coordinated Cache</a>			
<a href="#">RMI Coordinated Cache</a>	✓	✓	✓

**Table 103–13 (Cont.) Coordinated Cache Support for Packet Time-to-Live Configuration**

Coordinated Cache	How to Use Oracle JDeveloper	How to Configure a Packet Time-to-Live Using TopLink Workbench	How to Use Java
CORBA Coordinated Cache	✓	✓	✓
Custom Coordinated Cache			

In addition to configuring packet time-to-live, you may also need to configure announcement delay (see [Section 103.9, "Configuring an Announcement Delay"](#)).

### 103.12.1 How to Configure a Packet Time-to-Live Using TopLink Workbench

To specify the number of hops that session data packets can take before expiring, use this procedure:

1. Select a session or session broker in the **Navigator**. Its properties appear in the Editor.
2. Ensure the **Enable Cache Coordination** option is selected, then select the appropriate coordinated cache **Type** (see [Table 103–12](#)). The cache coordination options appear on the tab.

**Figure 103–10 Cache Coordination Tab, Packet Time to Live Field**

The screenshot shows the 'Cache Coordination' tab in the TopLink Workbench configuration editor. The 'Enable Cache Coordination' checkbox is checked. The 'Type' dropdown is set to 'RMI'. The 'Channel' is 'TopLinkCommandChannel'. The 'Multicast Group Address' is '226.10.12.64'. The 'Multicast Port' is '3,121'. The 'Packet Time to Live' field is highlighted with a red circle and contains the value '2'. The 'Announcement Delay' is '1,000'. There are also checkboxes for 'Remove Connection On Error' and 'Synchronous', and radio buttons for 'Registry Naming Service' and 'JNDI Naming Service'. The 'JNDI Naming Service' section includes fields for 'URL', 'User Name' (admin), 'Password' (password), and 'Initial Context Factory' (com.evermind.server.rmi.RMIInitialContextFactory).

In the **Packet Time to Live** field, specify the number of hops (default = 2) that session data packets can take before expiring.

## 103.12.2 How to Configure a Packet Time-to-Live Using Java

Use the `oracle.toplink.remotecommand.DiscoveryManager` method `setPacketTimeToLive` to specify the number of hops (default = 2) that session data packets can take before expiring.

For more information, see [Section 102.4.4, "Cache Coordination API"](#).

## Configuring a JMS Coordinated Cache

This chapter describes the various components that you must configure in order to use a JMS coordinated cache.

This chapter includes the following sections:

- [Introduction to JMS Coordinated Cache Configuration](#)
- [Configuring a Topic Name](#)
- [Configuring a Topic Connection Factory Name](#)
- [Configuring a Topic Host URL](#)
- [Configuring Connection Handling](#)

### 104.1 Introduction to JMS Coordinated Cache Configuration

Table 104–1 lists the configurable options for a JMS coordinated cache.

**Table 104–1 Configurable Options for a JMS Coordinated Cache**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Cache coordination change propagation at the descriptor level (see Section 119.15, "Configuring Cache Coordination Change Propagation at the Descriptor Level")	✓	✓	✓
Synchronous change propagation mode (see Section 103.2, "Configuring the Synchronous Change Propagation Mode")	✓	✓	✓
JNDI naming service (see Section 103.7, "Configuring JNDI Naming Service Information")	✓	✓	✓
Topic name (see Section 104.2, "Configuring a Topic Name")	✓	✓	✓
Topic connection factory name (see Section 104.3, "Configuring a Topic Connection Factory Name")	✓	✓	✓
Topic host URL (see Section 104.4, "Configuring a Topic Host URL")	✓	✓	✓
Connection handling (see Section 104.5, "Configuring Connection Handling")	✓	✓	✓
Context properties (see Section 103.11, "Configuring Context Properties")	✓	✓	✓
Packet time-to-live (see Section 103.12, "Configuring a Packet Time-to-Live")			✓

## 104.2 Configuring a Topic Name

A JMS topic identifies a publish/subscribe destination for a JMS server. JMS users who wish to share messages subscribe to the same JMS topic.

The topic name you configure is the name that TopLink uses to look up the `javax.jms.Topic` instance from the JNDI service. You must provide a fully qualified JNDI name, such as `jms/<topic_name>`.

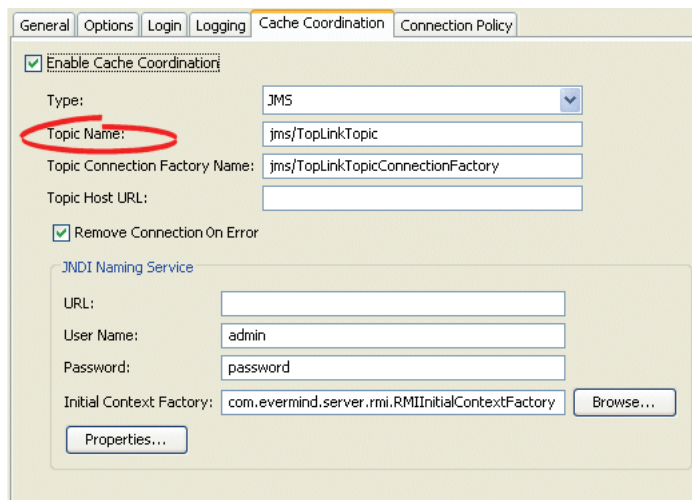
All the members of the same JMS coordinated cache must use the same JMS topic.

### 104.2.1 How to Configure a Topic Name Using TopLink Workbench

To specify the topic name for JMS cache coordination, use this procedure:

1. Select a server session in the **Navigator**. Its properties appear in the Editor.
2. Click the **Cache Coordination** tab. The Cache Coordination tab appears.
3. Ensure **Enable Cache Coordination** is selected and the **Type** is **JMS** (see [Section 102.3, "Cache Coordination"](#) for more information).

**Figure 104–1** Cache Coordination Tab, Topic Name Field, JMS



4. Enter the topic name to use with the JMS coordinated cache for this session. This must be a fully qualified JNDI name, such as `jms/<topic_name>`.

Enter the topic name to use with the JMS coordinated cache for this session. This must be a fully qualified JNDI name, such as `jms/<topic_name>`.

### 104.2.2 How to Configure a Topic Name Java

User the `oracle.toplink.remotecommand.broadcast.BroadcastTransportManager` method `setTopicName` to configure the Topic name for the Topic to which this transport manager will be connecting.

You obtain the `BroadcastTransportManager` using the following `Session` API:

```
Session.getCommandManager().getTransportManager()
```



## 104.3 Configuring a Topic Connection Factory Name

A JMS topic connection factory creates connections with the JMS provider for a specific JMS destination. Each connection factory contains the specific configuration information to create a connection to a JMS destination.

The topic connection factory name you configure is the name that TopLink uses to look up the `javax.jms.TopicConnectionFactory` instance from the JNDI service. This must be a fully qualified JNDI name, such as `jms/<resource_name>`.

### 104.3.1 How to Configure a Topic Connection Factory Name Using TopLink Workbench

To specify the topic connection factory for a JMS coordinated cache, use this procedure:

1. Select a server session in the **Navigator**. Its properties appear in the Editor.
2. Click the **Cache Coordination** tab. The Cache Coordination tab appears.
3. Ensure **Enable Cache Coordination** is selected and the **Type** is **JMS** (see [Section 102.3, "Cache Coordination"](#) for more information).

**Figure 104–2 Cache Coordination Tab, Topic Connection Factory Name Field**

4. Enter the topic connection factory name to use with the JMS coordinated cache for this session. This must be a fully qualified JNDI name, such as `jms/<resource_name>`.

Enter the topic connection factory name to use with the JMS coordinated cache for this session. This must be a fully qualified JNDI name, such as `jms/<resource_name>`.

### 104.3.2 How to Configure a Topic Connection Factory Name Using Java

Use the `oracle.toplink.remotecommand.jms.JMSTopicTransportManager` method `setTopicConnectionFactoryName` to configure the JMS Topic connection factory name for the JMS Topic connections.

You obtain the `JMSTopicTransportManager` using the following `Session` API:

```
Session.getCommandManager().getTransportManager()
```

## 104.4 Configuring a Topic Host URL

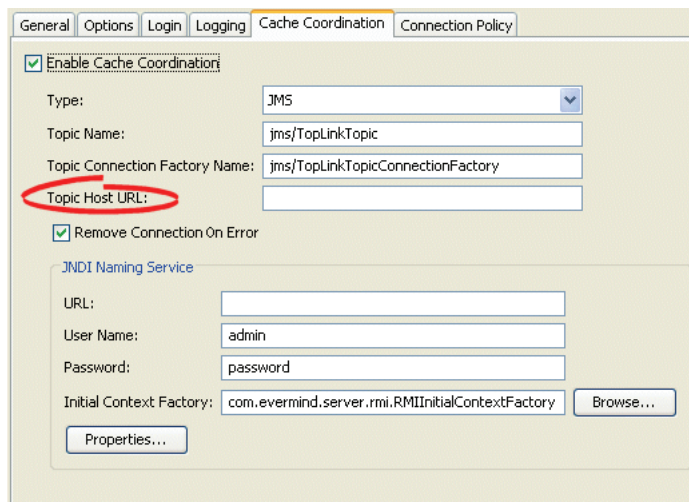
The JMS topic host URL is the URL of the machine on the network that hosts the JMS topic (see [Section 104.2, "Configuring a Topic Name"](#)).

### 104.4.1 How to Configure a Topic Host URL Using TopLink Workbench

To specify the topic host URL for a JMS coordinated cache, use this procedure:

1. Select a server session in the **Navigator**. Its properties appear in the Editor.
2. Click the **Cache Coordination** tab. The Cache Coordination tab appears.
3. Ensure **Enable Cache Coordination** is selected and the **Type** is **JMS** (see [Section 102.3, "Cache Coordination"](#) for more information).

**Figure 104–3** Cache Coordination Tab, Topic Host URL Field



Enter the URL of the machine on the network that hosts the JMS topic (see [Section 104.2, "Configuring a Topic Name"](#)) to use with the JMS coordinated cache for this session.

### 104.4.2 How to Configure a Topic Host URL Using Java

Use the `oracle.toplink.remotecommand.jms.JMSTopicTransportManager` method `setTopicHostURL` to configure the URL of the computer on the network that hosts the JMS Topic.

You obtain the `JMSTopicTransportManager` using the following `Session` API:

```
Session.getCommandManager().getTransportManager()
```

## 104.5 Configuring Connection Handling

The session's transport manager creates connections to the various members of the coordinated cache. If a communication error occurs on one of these connections, you can configure the session to either ignore the error or remove the connection.

If you configure the session to remove the connection on error, the next time the session tries to communicate with that coordinated cache member, it will construct a new connection. If an error occurs during the connection creation phase, TopLink will either throw a `RemoteCommandManagerException.ERROR_CREATING_JMS_`

CONNECTION (if the error occurred while sending a message) or a `RemoteCommandManagerException.ERROR_CREATING_LOCAL_JMS_CONNECTION` (if the error occurred while receiving a message). If you want to recover from this failure, consider the following options:

- You may choose to take no action: messages will not be sent or received.
- You may choose to handle the exception. You may do so by changing some of the `oracle.toplink.remotecommand.jms.JMSTopicTransportManager` settings and calling the `createExternalConnection` or `createInternalConnection` method of the `JMSTopicTransportManager`.

If you configure the session to ignore the error, the next time the session tries to communicate with that coordinated cache member, it will continue to use the same connection. In this case, if the listening (local) connection gets a `RemoteCommandManagerException.ERROR_RECEIVING_JMS_MESSAGE` exception, the coordinated cache waits for 10 seconds before resuming listening. If you want to recover from this failure, consider the following options:

- You may choose to take no action (wait for the connection recovery).
- You may choose to handle the `RemoteCommandManagerException.ERROR_PROPAGATING_COMMAND` or `RemoteCommandManagerException.ERROR_RECEIVING_JMS_MESSAGE` exception. You may do so by shutting down the remote command manager.

In either case, if the coordinated cache receives a null JMS message, it will throw a `RemoteCommandManagerException.ERROR_RECEIVED_JMS_MESSAGE_IS_NULL` exception.

### 104.5.1 How to Configure Connection Handling Using TopLink Workbench

To specify how TopLink handles session connections in the event of an error, use this procedure:

1. Select a session or session broker in the **Navigator**. Its properties appear in the Editor.
2. Click the **Cache Coordination** tab. The Cache Coordination tab appears.
3. Ensure the **Enable Cache Coordination** option is selected, then select the appropriate coordinated cache **Type** (JMS). The cache coordination options appear on the tab.

**Figure 104–4 Cache Coordination Tab, Remove Connection on Error Option**

The screenshot shows a configuration window with several tabs: General, Options, Login, Logging, Cache Coordination (selected), and Connection Policy. Under the 'Cache Coordination' tab, the 'Enable Cache Coordination' checkbox is checked. Below it, there are fields for 'Type' (set to JMS), 'Topic Name' (jms/TopLinkTopic), 'Topic Connection Factory Name' (jms/TopLinkTopicConnectionFactory), and 'Topic Host URL'. The 'Remove Connection on Error' checkbox is checked and circled in red. Below this is a 'JNDI Naming Service' section with fields for 'URL', 'User Name' (admin), 'Password' (password), and 'Initial Context Factory' (com.evermind.server.rmi.RMIInitialContextFactory). There are also 'Browse...' and 'Properties...' buttons.

4. Select the **Remove Connection on Error** option to configure the session to remove the data source connection in the event of an error.

Select the **Remove Connection on Error** option to configure the session to remove the data source connection in the event of an error.

## 104.5.2 How to Configure Connection Handling Using Java

Use the `oracle.toplink.remotecommand.TransportManager` method `setShouldRemoveConnectionOnError` to define whether connections to remote services should be disconnected when an error occurs.

You obtain the `TransportManager` using the following `Session` API:

```
Session.getCommandManager().getTransportManager()
```

## Configuring an RMI Coordinated Cache

This chapter describes the various components that you must configure in order to use an RMI coordinated cache.

This chapter includes the following sections:

- [Introduction to RMI Coordinated Cache Configuration](#)

### 105.1 Introduction to RMI Coordinated Cache Configuration

Table 105–1 lists the configurable options for an RMI coordinated cache.

**Table 105–1** Configurable Options for an RMI Coordinated Cache

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Cache coordination change propagation at the descriptor level (see Section 119.15, "Configuring Cache Coordination Change Propagation at the Descriptor Level")		✓	✓
Synchronous change propagation mode (see Section 103.2, "Configuring the Synchronous Change Propagation Mode")		✓	✓
Service channel (see Section 103.3, "Configuring a Service Channel")		✓	✓
Multicast group address (see Section 103.4, "Configuring a Multicast Group Address")		✓	✓
Multicast port (see Section 103.5, "Configuring a Multicast Port")		✓	✓
Naming service type (see Section 103.6, "Configuring a Naming Service Type")		✓	✓
Announcement delay (see Section 103.9, "Configuring an Announcement Delay")		✓	✓
Connection handling (see Section 103.10, "Configuring Connection Handling")		✓	✓
Context properties (see Section 103.11, "Configuring Context Properties")		✓	✓
Packet time-to-live (see Section 103.12, "Configuring a Packet Time-to-Live")		✓	✓



## Configuring a CORBA Coordinated Cache

This chapter describes the various components that you must configure to use a CORBA coordinated cache.

This chapter includes the following sections:

- [Introduction to CORBA Coordinated Cache Configuration](#)

### 106.1 Introduction to CORBA Coordinated Cache Configuration

Table 106–1 lists the configurable options for a CORBA coordinated cache.

**Table 106–1** Configurable Options for a CORBA Coordinated Cache

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Cache coordination change propagation at the descriptor level (see Section 119.15, "Configuring Cache Coordination Change Propagation at the Descriptor Level")	✓	✓	✓
Synchronous change propagation mode (see Section 103.2, "Configuring the Synchronous Change Propagation Mode")	✓	✓	✓
Service channel (see Section 103.3, "Configuring a Service Channel")	✓	✓	✓
Multicast group address (see Section 103.4, "Configuring a Multicast Group Address")	✓	✓	✓
Multicast port (see Section 103.5, "Configuring a Multicast Port")	✓	✓	✓
Naming service type (see Section 103.6, "Configuring a Naming Service Type")	✓	✓	✓
Announcement delay (see Section 103.9, "Configuring an Announcement Delay")	✓	✓	✓
Connection handling (see Section 103.10, "Configuring Connection Handling")	✓	✓	✓
Context properties (see Section 103.11, "Configuring Context Properties")	✓	✓	✓
Packet time-to-live (see Section 103.12, "Configuring a Packet Time-to-Live")			✓





## Configuring a Custom Coordinated Cache

This chapter describes the various components that you must configure to use a custom, user-defined coordinated cache. For more information, see [Section 102.3.4, "Custom Coordinated Cache"](#).

This chapter includes the following sections:

- [Introduction to Custom Coordinated Cache Configuration](#)
- [Configuring Transport Class](#)

### 107.1 Introduction to Custom Coordinated Cache Configuration

[Table 107–1](#) lists the configurable options for a custom coordinated cache.

**Table 107–1** Configurable Options for a Custom Coordinated Cache

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Cache coordination change propagation at the descriptor level (see <a href="#">Section 119.15, "Configuring Cache Coordination Change Propagation at the Descriptor Level"</a> )	✓	✓	✓
Service channel (see <a href="#">Section 103.3, "Configuring a Service Channel"</a> )	✓	✓	✓
Transport class (see <a href="#">Section 107.2, "Configuring Transport Class"</a> )	✓	✓	✓
Connection handling (see <a href="#">Section 103.10, "Configuring Connection Handling"</a> )	✓	✓	✓

### 107.2 Configuring Transport Class

To configure a custom coordinated cache, you must specify your custom instance of `oracle.toplink.remotecommand.TransportManager`.

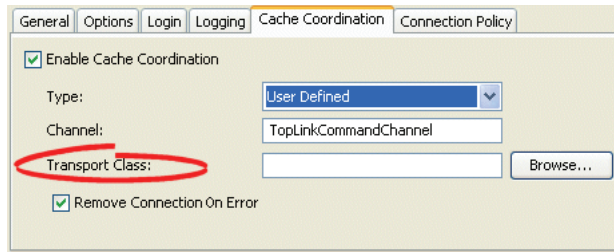
This section describes [How to Configure Transport Class Using TopLink Workbench](#).

#### 107.2.1 How to Configure Transport Class Using TopLink Workbench

To select the transport class for the user defined coordinated cache, use this procedure:

1. Select a session or session broker in the **Navigator**. Its properties appear in the Editor.
2. Click the **Cache Coordination** tab. The Cache Coordination tab appears.
3. Ensure the **Enable Cache Coordination** option is selected and the **Type** is **User Defined** (see [Section 102.3, "Cache Coordination"](#)).

**Figure 107–1 Cache Coordination, Transport Class Option**



4. Click **Browse** and select the transport class for the user-defined coordinated cache.  
Click **Browse** and select the transport class for the user-defined coordinated cache.

## 107.2.2 How to Configure Transport Class Using Java

Create a custom instance of the `oracle.toplink.remotecommand.TransportManager` that you use as a transport class for your coordinated cache.

You obtain the `TransportManager` using the following `Session` API:

```
Session.getCommandManager().getTransportManager()
```

# Part XXIV

---

## Queries

This part describes building TopLink queries and using them to create, read, update, and delete objects. It contains the following chapters.:

- [Chapter 108, "Introduction to TopLink Queries"](#)  
This chapter describes each of the different TopLink query types and important query concepts.
- [Chapter 109, "Using Basic Query API"](#)  
This chapter explains how to use basic TopLink query options.
- [Chapter 110, "Introduction to TopLink Expressions"](#)  
This chapter describes the TopLink expressions framework and how to use it with TopLink queries.
- [Chapter 111, "Using Advanced Query API"](#)  
This chapter explains how to use advanced TopLink query options.
- [Chapter 112, "Introduction to TopLink Support for Oracle Spatial"](#)  
This chapter explains how to extend TopLink to support the mapping and querying of Oracle Spatial.



---

---

## Introduction to TopLink Queries

TopLink enables you to create, read, update, and delete persistent objects or data using queries in both Java EE and non-Java EE applications for both relational and nonrelational data sources.

This chapter includes the following sections:

- [Query Types](#)
- [Query Concepts](#)
- [Building Queries](#)
- [Executing Queries](#)
- [Handling Query Results](#)
- [Session Queries](#)
- [Database Queries](#)
- [Named Queries](#)
- [Call Queries](#)
- [Redirect Queries](#)
- [Historical Queries](#)
- [Interface and Inheritance Queries](#)
- [Descriptor Query Manager Queries](#)
- [Oracle Extensions](#)
- [EJB 2.n CMP Finders](#)
- [Queries and the Cache](#)
- [Query API](#)

### 108.1 Query Types

Table 108–1 lists the query types that you can build in TopLink.

**Table 108–1 TopLink Query Types**

Query Type	Description	Oracle JDeveloper	TopLink Workbench	Java
Session Queries	A query implicitly constructed and executed by a <code>Session</code> based on input parameters used to perform the most common data source actions on objects.			✓
Database Queries	A query also known as a <i>query object query</i> . An instance of <code>DatabaseQuery</code> that you create and then execute to perform any data source action on either objects or data. You can further refine a <code>DatabaseQuery</code> by also creating and configuring its <code>Call</code> (see <a href="#">Section 108.9, "Call Queries"</a> ).			✓
Named Queries	An instance of <code>DatabaseQuery</code> stored by name in a <code>Session</code> or a descriptor's <code>DescriptorQueryManager</code> where it is constructed and prepared once. Such a query can then be repeatedly executed by name.	✓	✓	✓
Call Queries	An instance of <code>Call</code> that you create and then either execute directly, using a special <code>Session</code> API to perform limited data source actions on data only, or execute indirectly in the context of a <code>DatabaseQuery</code> . TopLink supports <code>Call</code> instances for custom SQL, stored procedures, and EIS interactions.	✓	✓	✓
Redirect Queries	An instance of <code>MethodBasedQueryRedirector</code> (taking the name of a static method and the <code>Class</code> in which it is defined as parameters) set on a named query. When the query is executed, the static method is invoked.			✓
Historical Queries	Any query executed in the context of a historical session using the time-aware features of the TopLink <code>Expression</code> framework.			✓
Interface and Inheritance Queries	Any query that references an interface type or super and subclasses of an inheritance hierarchy.			✓
Descriptor Query Manager Queries	The <code>DescriptorQueryManager</code> defines a default <code>DatabaseQuery</code> for each basic data source operation (create, read, update, and delete), and provides an API with which you can customize either the <code>DatabaseQuery</code> or its <code>Call</code> .	✓	✓	✓
EJB 2.n CMP Finders	A query defined on the home interface of an enterprise bean that returns enterprise beans. You can implement finders using any TopLink query type, including <code>JPAQLCall</code> and <code>EJBQLCall</code> , a <code>Call</code> that takes JPA/EJB QL.	✓	✓	✓

For more information, see the following:

- [Chapter 109, "Using Basic Query API"](#)
- [Chapter 111, "Using Advanced Query API"](#)
- [Section 108.2, "Query Concepts"](#)

## 108.2 Query Concepts

In general, querying a data source means performing an action on or interacting with the contents of the data source. To do this, you must be able to perform the following:

- Define an action in a syntax native to the data source being queried.
- Apply the action in a controlled fashion.
- Manage the results returned by the action (if any).

Specific to TopLink, you must also consider how the query affects the TopLink cache. For more information, see [Section 108.16, "Queries and the Cache"](#).

This section introduces query concepts unique to TopLink, including the following:

- [Call](#)
- [DatabaseQuery](#)
- [Data-Level and Object-Level Queries](#)
- [Summary Queries](#)
- [Descriptor Query Manager](#)
- [TopLink Expressions](#)
- [Query Keys](#)
- [Query Languages](#)

## 108.2.1 Call

In TopLink, the `Call` object encapsulates an operation or action on a data source. TopLink provides a variety of `Call` types such as structured query language (SQL), Enterprise Java Beans Query Language (EJB QL), Java Persistence Query Language (JP QL), Extensible Markup Language (XML), and enterprise information system (EIS).

You can execute a `Call` directly or in the context of a `DatabaseQuery`.

## 108.2.2 DatabaseQuery

A `DatabaseQuery` object is an abstraction that associates additional customization and optimization options with the action encapsulated by a `Call`. By separating these options from the `Call`, TopLink can provide sophisticated query capabilities across all `Call` types.

For more information, see [Section 108.7, "Database Queries"](#).

## 108.2.3 Data-Level and Object-Level Queries

In TopLink, queries can be defined for objects or data, as follows:

- **Object-level** queries (see [Section 108.7.1, "Object-Level Read Query"](#) and [Section 108.7.3, "Object-Level Modify Query"](#)) are object-specific and return data as objects in your domain model. They are the preferred type of query for mapped data. By far, object-level `DatabaseQuery` queries are the most common query used in TopLink.
- **Data-level** queries (see [Section 108.7.2, "Data-Level Read Query"](#) and [Section 108.7.4, "Data-Level Modify Query"](#)) are used to query database tables directly, and are an appropriate way to work with unmapped data.

## 108.2.4 Summary Queries

While data-level queries return raw data and object-level queries return objects in your domain model, summary queries return data about objects. TopLink provides partial object queries (see [Section 108.7.1.3, "Partial Object Queries"](#)) to return a set of objects with only specific attributes populated, and report queries (see [Section 108.7.5, "Report Query"](#)) to return summarized (or rolled-up) data for specific attributes of a set of objects.

## 108.2.5 Descriptor Query Manager

In addition to storing named queries applicable to a particular class (see [Section 108.8, "Named Queries"](#)), you can also use the `DescriptorQueryManager` to override the

default action that TopLink defines for common data source operations. For more information, see [Section 108.13, "Descriptor Query Manager Queries"](#).

## 108.2.6 TopLink Expressions

TopLink expressions let you specify query search criteria based on your domain object model. When you execute the query, TopLink translates these search criteria into the appropriate query language for your platform.

TopLink provides the following two public classes to support expressions:

- The `Expression` class represents an expression that can be anything from a simple constant to a complex clause with boolean logic. You can manipulate, group, and integrate expressions.
- The `ExpressionBuilder` class is the factory for constructing new expressions.

You can specify a selection criterion as an `Expression` with `DatabaseQuery` method `setSelectionCriteria` (see [Section 108.7, "Database Queries"](#)), and in a finder that takes an `Expression` (see [Section 108.15.7, "Expression Finders"](#)).

For more information about using TopLink expressions, see [Chapter 110, "Introduction to TopLink Expressions"](#).

## 108.2.7 Query Keys

A query key is a schema-independent alias for a database field name. Using a query key, you can refer to a field using a schema-independent alias. In relational projects only, TopLink automatically creates query keys for all mapped attributes. The name of the query key is the name of the class attribute specified in your object model.

You can configure query keys in a class descriptor (see [Section 119.10, "Configuring Query Keys"](#)) or interface descriptor (see [Section 119.11, "Configuring Interface Query Keys"](#)).

You can use query keys in expressions (see [Section 110.4, "Query Keys and Expressions"](#)) and to query variable one-to-one mappings (see [Section 111.8, "Using Queries on Variable One-to-One Mappings"](#)).

## 108.2.8 Query Languages

Using TopLink, you can express a query using any of the following query languages:

- [SQL Queries](#)
- [EJB QL Queries](#)
- [JP QL Queries](#)
- [XML Queries](#)
- [EIS Interactions](#)
- [Query-by-Example](#)
- [TopLink Expressions](#) (see [Chapter 110, "Introduction to TopLink Expressions"](#))

In most cases, you can compose a query directly in a given query language or, preferably, you can construct a `DatabaseQuery` with an appropriate `Call` and specify selection criteria using a `TopLink Expression`. Although composing a query directly in SQL appears to be the simplest approach (and for simple operations or operations on unmapped data, it is), using the `DatabaseQuery` approach offers the



compelling advantage of confining your query to your domain object model and avoiding dependence on data source schema implementation details.

Oracle recommends that you compose your queries using JP QL or `Expression`.

### 108.2.8.1 SQL Queries

SQL is the most common query language for applications that use a relational database data source.

You can execute custom SQL directly using `Session` methods `executeSelectingCall` and `executeNonSelectingCall`, or you can construct a `DatabaseQuery` with an appropriate `Call`.

`TopLink` provides a variety of SQL `Call` objects for use with stored procedures and, with Oracle Database, stored functions. For more information, see [Section 108.9.1, "SQL Calls"](#).

`TopLink` also supports PL/SQL call for Oracle stored procedures with PL/SQL data types. For more information, see [Section 109.5, "Using a StoredProcedureCall"](#).

### 108.2.8.2 EJB QL Queries

Like SQL, EJB QL is a query language; but unlike SQL, it presents queries from an object model perspective, allowing you to declare queries using the attributes of each abstract entity bean in your object model. It also includes path expressions that enable navigation over the relationships defined between entity beans and dependent objects.

Using EJB QL offers the following advantages:

- You do not need to know the database structure (such as tables and fields).
- You can construct queries using the attributes of the entity beans instead of using database tables and fields.
- You can use relationships in a query to provide navigation from attribute to attribute.
- EJB QL queries are portable because they are database-independent.
- You can specify the reference class in the `SELECT` clause.

The disadvantage of EJB QL queries is that it is difficult to use when you construct complex queries.

`TopLink` provides the full support for the EJB QL specification.

---

---

**Note:** `TopLink` supports the `LOCATE` string function and will generate the correct SQL with it. However, not all data sources support `LOCATE`. Before using the `LOCATE` string function, consult your data source documentation.

---

---

EJB QL is the standard query language first defined in the EJB 2.0 specification. Consequently, `TopLink` lets you specify selection criteria using EJB QL in an EJB finder (see [Section 108.15.8, "EJB QL Finders"](#)).

Although EJB QL is usually associated with EJB, `TopLink` also lets you specify selection criteria using EJB QL in queries for regular Java objects as well. `TopLink` provides an EJB QL `Call` that you can execute directly or in the context of a `DatabaseQuery`. For more information, see [Section 108.9.2, "EJB QL Calls"](#) and [Section 108.2.2, "DatabaseQuery"](#).

### 108.2.8.3 JP QL Queries

See "What You May Need to Know About Querying with Java Persistence Query Language" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Developing\\_Applications\\_Using\\_EclipseLink\\_JPA\\_%28ELUG%29#What\\_You\\_May\\_Need\\_to\\_Know\\_About\\_Querying\\_with\\_Java\\_Persistence\\_Query\\_Language](http://wiki.eclipse.org/Developing_Applications_Using_EclipseLink_JPA_%28ELUG%29#What_You_May_Need_to_Know_About_Querying_with_Java_Persistence_Query_Language) for more information.

### 108.2.8.4 XML Queries

You can use TopLink XML to query XML data stored in Oracle Database XMLType field. For more information, see [Section 27.4, "Direct-to-XMLType Mapping"](#) and [Section 110.2.4, "XMLType Functions"](#).

### 108.2.8.5 EIS Interactions

When you execute a TopLink query using an EIS Call (see [Section 108.9.3, "Enterprise Information System \(EIS\) Interactions"](#)), TopLink converts your selection criteria into an XML format appropriate for your JCA adapter.

If supported by your JCA adapter, you can use the XQuery language by executing an XQuery interaction (see [Section 108.9.3.4, "XQueryInteraction"](#)) either directly or in the context of a DatabaseQuery.

### 108.2.8.6 Query-by-Example

Query-by-example is a simple and intuitive way to express a query. To specify a query-by-example, provide a sample instance of the persistent object to query, and set appropriate values on only the data members on which you wish to query.

Query-by-example lets you query for an object based on any attribute that uses a direct mapping or a one-to-one relationship (including those with nesting).

---

---

**Note:** Query-by-example does not support any other relationship mapping types, nor does it support EJB 2.1 CMP beans.

---

---

Set only the attributes on which you base the query; set all other attributes to null. By default, TopLink ignores attributes in the sample instance that contain null, zero (0), empty strings, and FALSE. You can modify this list of values (and define other query by example options) by specifying a QueryByExamplePolicy (see [Defining a QueryByExamplePolicy](#)).

Query-by-example uses the AND operator to tie the attribute comparisons together. By default, attribute values in the sample instance are compared against corresponding values of candidate objects using EQUALS operator. You can modify this behaviour using the QueryByExamplePolicy.

Both ReadAllQuery and ReadObjectQuery provide a setExampleObject method and setQueryByExamplePolicy method that you can use to specify selection criteria based on an example object instance.

For more information and examples, see [Section 109.2.1.4, "Reading Objects Using Query-By-Example"](#).

## 108.3 Building Queries

You can build queries using Oracle JDeveloper, TopLink Workbench, or Java using the TopLink API.

Some queries are implicitly constructed for you based on passed in arguments and executed in one step (for example, session queries, as described in [Section 108.6](#), "Session Queries") and others you explicitly create, configure, and then execute (for example, [Section 108.7](#), "Database Queries").

For more information, see the following:

- [Chapter 109](#), "Using Basic Query API"
- [Chapter 111](#), "Using Advanced Query API"

## 108.4 Executing Queries

In TopLink, you execute most queries using the `Session` API summarized in [Table 108–2](#).

**Table 108–2** *Session Methods for Executing a Query*

Query Type	Session Method	Advantages and Disadvantages
Session Queries	<code>readObject</code>	Advantages: the most convenient way to perform common data source operations on objects.
	<code>readAllObjects</code>	
	<code>writeObject</code>	
	<code>writeAllObjects</code>	
	<code>deleteObject</code>	Disadvantages: less control over query execution and results; less efficient for frequently executed queries.
	<code>deleteAllObjects</code>	
	<code>insertObject</code>	
	<code>updateObject</code>	
Database Queries Named Queries Redirect Queries	<code>executeQuery</code>	Advantages: greatest configuration and execution flexibility; can take advantage of named queries for efficiency.  Disadvantages: you must explicitly create and configure <code>DatabaseQuery</code> and possibly <code>Call</code> objects.
Call Queries	<code>executeSelectingCall</code>	Advantages: convenient way to directly apply an action to unmapped data.  Disadvantages: least control over query execution and results; your application must do more work to handle raw data results.
	<code>executeNonSelectingCall</code>	

---

**Note:** Oracle recommends that you perform all data source operations using a unit of work: doing so is the most efficient way to manage transactions, concurrency, and referential constraints. For more information, see [Chapter 113](#), "Introduction to TopLink Transactions".

Alternatively, you can execute queries outside of a unit of work using a session API directly, but doing so places greater responsibility on your application to manage transactions, concurrency, and referential constraints.

---

TopLink executes `DescriptorQueryManager` queries when you execute a session query. For more information, see [Section 108.13](#), "Descriptor Query Manager Queries".

You execute EJB 2.1 CMP finders when you call the appropriate finder method on an EJB 2.1 CMP bean. For more information, see [Section 108.15](#), "EJB 2.1 CMP Finders".

---

---

**WARNING:** Allowing an unverified SQL string to be passed into methods (for example: `setSQLString(String sql)`, `readAllObjects(Class class, String sql)` methods) makes your application vulnerable to SQL injection attacks.

---

---

For more information, see the following:

- [Chapter 109, "Using Basic Query API"](#)
- [Chapter 111, "Using Advanced Query API"](#)

## 108.5 Handling Query Results

TopLink queries generally return Java objects as their result set. TopLink queries can return any of the following:

- Entire objects, with all attributes populated and the object reflected in the cache.
- Collections of objects (see [Section 108.5.1, "Collection Query Results"](#)).
- Collections of records.
- Report summaries.
- Partial objects, with only the attributes you specify populated and without the object reflected in the cache (see [Section 108.5.2, "Report Query Results"](#)).
- Streams of objects (see [Section 108.5.3, "Stream and Cursor Query Results"](#)).
- EJB (see [Section 108.15, "EJB 2.n CMP Finders"](#)).

### 108.5.1 Collection Query Results

A collection is a group of Java objects contained by an instance of `Collection` or `Map`.

By default, queries that return more than one object return their results in a `Vector`.

You can configure TopLink to return query results in any concrete instance of `Collection` or `Map`.

Collection results are supported by all TopLink query types.

For information and examples on how to configure and handle collection query results, see [Section 109.10, "Handling Collection Query Results"](#).

### 108.5.2 Report Query Results

A `ReportQuery` (a type of partial object query) returns summary data for selected objects using the database reporting functions and features supported by your platform. Although the report query returns data (not objects), it does enable you to query the returned data and specify it at the object level.

By default, a `ReportQuery` returns a collection (see [Section 108.5.1, "Collection Query Results"](#)) of `ReportQueryResult` objects, one collection per database row returned.

You can use the `ReportQuery` API to configure how a `ReportQuery` returns its results. For more information see [Section 109.11, "Handling Report Query Results"](#).

For more information, see the following:

- [Section 108.7.5, "Report Query"](#)
- [Section 109.11, "Handling Report Query Results"](#)

- [Section 108.7.1.3, "Partial Object Queries"](#).

### 108.5.3 Stream and Cursor Query Results

A stream is a view of a collection, which can be a file, a device, or a `Vector`. A stream provides access to the collection, one element at a time in sequence. This makes it possible to implement stream classes in which the stream does not contain all the objects of a collection at the same time.

Large result sets can be resource-intensive to collect and process. To improve performance and give the client more control over the returned results, configure `TopLink` queries to use a cursor or stream.

Cursors & streams are supported by all subclasses of `DataReadQuery` and `ReadAllQuery`.

For more information, see [Section 111.11, "Handling Cursor and Stream Query Results"](#).

## 108.6 Session Queries

Sessions provide query methods that lets you perform the object operations listed in [Table 108–3](#).

**Table 108–3** *Session Object Query Summary*

Session Type	Create	Read	Update	Delete
UnitOfWork	registerObject	readObject readAllObjects	NA	deleteObject deleteAllObjects
Server	NA	NA	NA	NA
ClientSession	NA	readObject readAllObjects	NA	NA
DatabaseSession	insertObject	readObject readAllObjects	updateObject writeObject writeAllObjects	deleteObject deleteAllObjects

---

**Note:** Oracle recommends that you perform all data source operations using a unit of work: doing so is the most efficient way to manage transactions, concurrency, and referential constraints. For more information, see [Chapter 113, "Introduction to TopLink Transactions"](#).

---

These methods implicitly construct and execute a `DatabaseQuery` based on any of the following input parameters and return `Object` or `Object` collection:

- Reference Class (the Class of objects that the query accesses)
- Reference Class and Call
- Reference Class and Expression
- Example object with primary key set

These methods are a convenient way to perform the most common data source operations on objects.

---

---

**WARNING:** Allowing an unverified SQL string to be passed into these methods makes your application vulnerable to SQL injection attacks.

---

---

To access all configurable options to further refine and optimize a query, consider using a corresponding `DatabaseQuery` directly. For more information, see [Section 108.7, "Database Queries"](#).

For more information, see [Section 109.1, "Using Session Queries"](#).

### 108.6.1 Read-Object Session Queries

Read-object queries return the first instance of an `Object` that matches the specified selection criteria, and read-all object queries return all such instances.

You can also pass in a domain `Object` with its primary key set and `TopLink` will construct and execute a read-object query to select that object. This is one form of query by example. For more information on query by example, see [Section 108.2.8.6, "Query-by-Example"](#).

For more information, see [Section 109.1.1, "How to Read Objects with a Session Query"](#).

### 108.6.2 Create, Update, and Delete Object Session Queries

Oracle recommends that you create and update objects using a unit of work: doing so is the most efficient way to manage transactions, concurrency, and referential constraints. For more information, see [Chapter 113, "Introduction to TopLink Transactions"](#).

However, you can also create and update objects using a session query. These session queries are a convenient way to modify objects directly on the database when you manage simple, nonbusiness object data that has no relationships (for example, user preferences).

If you know an object is new, you can use an `insertObject` method to avoid having `TopLink` perform an existence check. If you do not know if an object is new, use the `updateObject`, `writeObject`, or `writeAllObject` methods: `TopLink` performs an existence check if necessary.

When you execute a write session query, it writes both the object and its privately owned parts to the database. To manage this behavior, use a corresponding `DatabaseQuery` (see [Section 108.7.3.7, "Object-Level Modify Queries and Privately Owned Parts"](#)).

Using the `Session` method `deleteObject`, you can delete a specific object. Using the `Session` method `deleteAllObjects`, you can delete a collection of objects. Each specified object and all its privately owned parts are deleted. In the case of `deleteAllObjects`, all deletions are performed within a single transaction.

For more information, see [Section 109.1.2, "How to Create, Update, and Delete Objects with a Session Query"](#).

## 108.7 Database Queries

All session types provide an `executeQuery` method that takes any of the following types of `DatabaseQuery`:

- [Object-Level Read Query](#)
- [Data-Level Read Query](#)
- [Object-Level Modify Query](#)
- [Data-Level Modify Query](#)
- [Report Query](#)

Using `DatabaseQuery` method `setCall`, you can define your own `Call` to accommodate a variety of data source options such as SQL (including stored procedures and stored functions), EJB QL queries, and EIS interactions. For more information, see [Section 108.9, "Call Queries"](#).

Using `DatabaseQuery` method `setSelectionCriteria`, you can specify your selection criteria using a `TopLink` Expression. For more information, see [Section 108.2.6, "TopLink Expressions"](#).

For more information, see [Section 109.2, "Using DatabaseQuery Queries"](#).

## 108.7.1 Object-Level Read Query

Using an `ObjectLevelReadQuery`, you can query your data source and return `Object` instances that match the specified selection criteria. This section describes the following:

- [ReadObjectQuery](#)
- [ReadAllQuery](#)
- [Partial Object Queries](#)
- [Read-Only Query](#)
- [Join Reading and Object-Level Read Queries](#)
- [Fetch Groups and Object-Level Read Queries](#)

For more information, see [Section 109.2.1, "How to Read Objects Using a DatabaseQuery"](#).

### 108.7.1.1 ReadObjectQuery

Using a `ReadObjectQuery`, you can query your data source and return the first object that matches the specified selection criteria.

### 108.7.1.2 ReadAllQuery

Using a `ReadAllQuery`, you can query your data source and return a `Collection` of all the objects that match the specified selection criteria.

### 108.7.1.3 Partial Object Queries

By default, an `ObjectLevelReadQuery` returns all attributes of the objects read.

If you require only certain attributes from selected objects, you can create a partial object query by using `ObjectLevelReadQuery` method `addPartialAttributes`. Using this method, you can improve query performance by making `TopLink` return objects with only specified attributes populated.

Applications frequently use partial object queries to compile a list for further selection. For example, a query to find the names and addresses of all employees over the age of 40 returns a list of data (the names and addresses) that partially represents objects (the employees). A common next step is to present this list so the user can select the

required object or objects from the list. Later retrieval of a complete object is simplified because TopLink always includes the primary key attribute (even if you do not add it as a partial attribute).

Consider the following when you use partial object queries:

- You cannot edit or cache partial objects.
- Unspecified attributes will be left null.
- You cannot have two partial attributes of the same type.
- You cannot add a partial attribute which is of the same type as the class being queried.

If you require only summary information for certain attributes from selected objects, it is more efficient to use a `ReportQuery` (see [Section 108.7.5, "Report Query"](#)).

For more information, see [Section 109.2.1.2, "Reading Objects Using Partial Object Queries"](#).

#### 108.7.1.4 Read-Only Query

In cases where you know that data is read-only, you can improve performance by specifying a query as read-only: this tells TopLink that any object returned by the query is immutable.

For more information, see the following:

- [Section 119.3, "Configuring Read-Only Descriptors"](#)
- [Section 111.4, "Using Read-Only Queries"](#)
- [Section 12.12.5, "How to Use Read-Only Queries for Optimization"](#)

#### 108.7.1.5 Join Reading and Object-Level Read Queries

Join reading is a query optimization feature that allows a single query for a class to return the data to build the instances of that class and its related objects. Use this feature to improve query performance by reducing database access. By default, relationships are not join-read: each relationship is fetched separately when accessed if you are using indirection (lazy loading) or as a separate database query if you are not using indirection. For more information, see [Section 17.2.4, "Indirection \(Lazy Loading\)"](#).

You can use join reading with `ReadObjectQuery` and `ReadAllQuery` to join the mapped relationships that [Table 108–4](#) lists. Join reading is not currently supported for any other relationship mappings.

**Table 108–4 Join Reading by Mapping Type**

Query	Mapping Type
<code>ReadObjectQuery</code>	<ul style="list-style-type: none"> <li>■ One-to-one (see <a href="#">Section 27.5, "One-to-One Mapping"</a>)</li> <li>■ One-to-many (see <a href="#">Section 27.7, "One-to-Many Mapping"</a>)</li> </ul>
<code>ReadAllQuery</code>	<ul style="list-style-type: none"> <li>■ Many-to-many (see <a href="#">Section 27.8, "Many-to-Many Mapping"</a>)</li> <li>■ Direct collection (see <a href="#">Section 27.10, "Direct Collection Mapping"</a>)</li> <li>■ Direct map (see <a href="#">Section 27.11, "Direct Map Mapping"</a>)</li> <li>■ Aggregate collection (see <a href="#">Section 27.9, "Aggregate Collection Mapping"</a>)</li> </ul>



Join reading can specify multiple and nested relationships to be joined. Nested joins are expressed through using expressions (see [Section 110.2.7, "Expressions for Joining and Complex Relationships"](#)).

Outer joins can also be used with join reading through using the expression `outer join` API. If an outer join is not used, objects with missing one-to-one relationships or empty one-to-many relationships will be filtered from the result set. You can also configure an object-level read query to allow inherited subclasses to be outer-joined to avoid the cost of a single query per class. You can also specify inner or outer joins using the `useInnerJoinFetch` or `useOuterJoinFetch` method of any of the mappings listed in [Table 108-4](#).

You can use join reading with custom SQL or stored procedures, but the query must ensure that all of the required data to build all of the join-read objects is returned. If the result set includes the same tables or fields, they must be returned in the same table order as `TopLink` would have generated.

For more information, see the following:

- [Section 109.2.1.10, "Using Join Reading with ObjectLevelReadQuery"](#)

**108.7.1.5.1 Avoiding Join-Reading Duplicate Data** Join reading can result in returning duplicate data if a one-to-many or a shared one-to-one relationship is joined. Although `TopLink` correctly filters the duplicate results from the object result, the duplicate data still must be fetched from the database and can degrade performance, especially if multiple one-to-many relationships are joined. In general, batch reading can be used as a better alternative to join reading, as it does not require fetching duplicate data.

Oracle recommends that you use one-to-many joining with caution, because it does not scale well in many situations.

Because the main cost of a `ReadObjectQuery` is SQL execution, the performance of a one-to-many join in this case is usually better than a query without joining.

However, because the main cost of a `ReadAllObjectQuery` is row-fetching, which the duplicate data of a join increases, the performance of a one-to-many join in this case is less efficient than batch reading in many scenarios (even though one-to-many joining is more efficient than reading the objects one-by-one).

This is mainly due to the fact that a one-to-many join reads in duplicate data: the data for each source object will be duplicated for each target object. Depending on the size of the one-to-many relationship and the size of the source object's row, this can become very inefficient, especially if the source object has a Large Object (LOB).

If you use multiple or nested one-to-many joins in the same query, the problem is compounded: the source object's row is duplicated  $n*m$  times, and each target object  $n$  and  $m$  times respectively. This can become a major performance issue.

To handle empty collections, you must use outer joins, so the queries can easily become very database intensive. Batch reading has the advantage of only returning the required data, and does not require outer joins.

Oracle recommends that you use batch reading to optimize querying relationships in read-all applications.

For more information, see the following:

- [Section 12.12.3, "How to Use Batch and Join Reading for Optimization"](#)
- [Section 12.12.9.2, "Reading Case 2: Batch Reading Objects"](#)

### 108.7.1.6 Fetch Groups and Object-Level Read Queries

You can use a fetch group with a `ReadObjectQuery` or `ReadAllQuery`. When you execute the query, TopLink retrieves only the attributes in the fetch group. TopLink automatically executes a query to fetch all the attributes excluded from this subset when and if you call a getter method on any one of the excluded attributes.

For more information, see the following:

- [Section 16.2.4, "Fetch Groups"](#)
- [Section 111.3, "Using Queries with Fetch Groups"](#)

## 108.7.2 Data-Level Read Query

Using a `DataLevelReadQuery`, you can query your data source and return `Object` instances that match the specified selection criteria. This section describes the following:

- [DataReadQuery](#)
- [DirectReadQuery](#)
- [ValueReadQuery](#)

For more information, see [Section 109.2.4, "How to Read Data with a DatabaseQuery"](#).

---

---

**WARNING:** Allowing an unverified SQL string to be passed into constructors of such objects as `DataReadQuery`, `DirectReadQuery` and `ValueReadQuery` makes your application vulnerable to SQL injection attacks.

---

---

### 108.7.2.1 DataReadQuery

Use a `DataReadQuery` to execute a selecting SQL string that returns a `Collection` of the `Record` objects representing the result set.

### 108.7.2.2 DirectReadQuery

Use a `DirectReadQuery` to read a single column of data (that is, one field) that returns a `Collection` of values representing the result set.

### 108.7.2.3 ValueReadQuery

Use a `ValueReadQuery` to read a single data value (that is, one field). A single data value is returned, or null if no rows are returned.

## 108.7.3 Object-Level Modify Query

Using an `ObjectLevelModifyQuery`, you can query your data source to create, update, and delete objects. This section describes the following:

- [WriteObjectQuery](#)
- [UpdateObjectQuery](#)
- [InsertObjectQuery](#)
- [DeleteObjectQuery](#)
- [UpdateAllQuery](#)
- [DeleteAllQuery](#)

- [Object-Level Modify Queries and Privately Owned Parts](#)

For more information, see [Section 109.2.2, "How to Create, Update, and Delete Objects with a DatabaseQuery"](#).

---

---

**Note:** Oracle recommends that you create and update objects using a `TopLink UnitOfWork`: doing so is the most efficient way to manage transactions, concurrency, and referential constraints. For more information, see [Chapter 113, "Introduction to TopLink Transactions"](#).

---

---

### 108.7.3.1 WriteObjectQuery

If you do not know whether or not an object is new, use a `WriteObjectQuery`: `TopLink` performs an existence check if necessary to determine whether to perform an insert or an update.

If you do know whether or not an object exists, you can avoid the existence check by using an `UpdateObjectQuery` (see [Section 108.7.3.2, "UpdateObjectQuery"](#)) or `InsertObjectQuery` (see [Section 108.7.3.3, "InsertObjectQuery"](#)).

### 108.7.3.2 UpdateObjectQuery

If you know that the object you want to modify exists, use an `UpdateObjectQuery` to avoid having `TopLink` perform an existence check.

### 108.7.3.3 InsertObjectQuery

If you know an object is new, you can use an `InsertObjectQuery` to avoid having `TopLink` perform an existence check.

### 108.7.3.4 DeleteObjectQuery

To delete a specific object, construct a `DeleteObjectQuery` with a single specific object as an argument.

### 108.7.3.5 UpdateAllQuery

The `UpdateAllQuery` allows you to take an expression and update a set of objects (at the object level) without loading the objects into memory. You can updated to either a specific or relative value. For example, you can set the value to 5 or to increase by 5 percent.

For more information, see [Section 109.2.3.1, "Using UpdateAll Queries"](#).

### 108.7.3.6 DeleteAllQuery

To delete multiple objects, construct a `DeleteAllQuery` and use its `setObjects` method to configure the collection of specific objects to delete. Use `DeleteAllQuery` method `setReferenceClass` to configure the reference class of the objects to delete. Each specified object is deleted, but its privately owned parts are not.

In the case of a `DeleteAllQuery`, all deletions are performed within a single transaction.

For more information, see [Section 109.2.3.2, "Using DeleteAll Queries"](#).

### 108.7.3.7 Object-Level Modify Queries and Privately Owned Parts

When you execute a create or update object `DatabaseQuery`, it writes both the object and its privately owned parts to the database by default. To create a query that does

not update privately owned parts, use the `DatabaseQuery` method `dontCascadeParts`. Use this method to do the following:

- Increase performance when you know that only the object's direct attributes have changed.
- Manually resolve referential integrity dependencies when you write large groups of new, independent objects.

---

---

**Note:** Because the unit of work resolves referential integrity internally, this method is not required if you use the unit of work to write to the data source. For more information, see [Chapter 113, "Introduction to TopLink Transactions"](#).

---

---

### 108.7.4 Data-Level Modify Query

Using a `DataModifyQuery`, you can query your data source to execute a nonselecting SQL statement. It is equivalent to `Session` method `executeNonSelectingCall`.

For more information, see [Section 109.2.5, "How to Update Data with a DatabaseQuery"](#).

### 108.7.5 Report Query

If you want to summarize (or roll up) certain attributes of a set of objects, you can use a `ReportQuery`.

A `ReportQuery` returns summary data from a set of objects and their related objects. That is, it returns data about objects. It can also return multiple objects. A `ReportQuery` lets you query and specify the data at the object level. To build a report query, you specify the search criteria, the data you require about the objects, and how that data should be summarized.

For example, you can create a report query to compute the average age of all employees in your company. The report query is not interested in the specific objects (the employees), but rather, summary information about them (their average age).

A `ReportQuery` lets you do the following:

- Specify a subset of the object's attributes and its related object's attributes, which allows you to query for lightweight information.
- Build complex object-level expressions for the selection criteria and ordering criteria.
- Use data source aggregation functions (supported by your platform), such as `SUM`, `MIN`, `MAX`, `AVG`, and `COUNT`.
- Use expressions to group data.
- Request primary key attributes with each `ReportQueryResult`. This makes it easy to request the real object from a lightweight result.

A `ReportQuery` is the most efficient form of partial object query (see [Section 108.7.1.3, "Partial Object Queries"](#)) because it takes advantage of the reporting capabilities of your data source (if available). Oracle recommends that you use `ReportQuery` to do partial object queries.

The `ReportQuery` API returns a collection of `ReportQueryResult` objects, similar in structure and behavior to a `Record` or a `Map`. For more information, see [Section 108.5.2, "Report Query Results"](#).

For more information, see the following:

- [Section 12.12.9.1, "Reading Case 1: Displaying Names in a List"](#)
- [Section 109.2.1.3, "Reading Objects Using Report Queries"](#)
- [Section 119.7, "Configuring Named Queries at the Descriptor Level"](#)

## 108.8 Named Queries

When you use a session query method like `readAllObjects` (see [Section 108.6, "Session Queries"](#)), TopLink creates a corresponding `ReadAllQuery`, which builds other objects it needs to perform its task. When TopLink finishes execution of the `readAllObjects` method, these objects are discarded. Each time you call this session method, TopLink creates these related objects again, uses them once, and then discards them.

Alternatively, you can create a `DatabaseQuery` (see [Section 108.7, "Database Queries"](#)) and store it by name at the descriptor-level (see [Section 119.7, "Configuring Named Queries at the Descriptor Level"](#)) or session-level (see [Section 89.13, "Configuring Named Queries at the Session Level"](#)).

TopLink prepares a named query once, and it (and all its associated supporting objects) can be efficiently reused thereafter making a named query well suited for frequently executed operations.

Using the `Session` API (see [Section 109.3, "Using Named Queries"](#)), you can execute these queries by name, passing in any required arguments.

### When to Use Named Queries

For a reasonably complex query that you execute frequently, you should consider making the query a named query.

If a query is global to a project, configure the named query at the session level ([Section 89.13, "Configuring Named Queries at the Session Level"](#)).

If a query is global to a `Class` or you want to configure CMP finders, configure the named query at the descriptor level (see [Section 119.7, "Configuring Named Queries at the Descriptor Level"](#)). For more information about descriptor level query configuration, see [Section 108.13, "Descriptor Query Manager Queries"](#).

For a very complex query, you can delegate query execution to your own static method using a special form of a named query called a redirect query. For more information about redirect queries, see [Section 108.10, "Redirect Queries"](#).

### When Not to Use Named Queries

Rarely used queries may be more efficient when built on an as-needed basis. If you seldom use a given query, it may not be worthwhile to build and store that query when you invoke a session.

## 108.9 Call Queries

All session types provide `executeSelectingCall` and `executeNonSelectingCall` methods that take any of the following `Call` types:

- [SQL Calls](#)
- [EJB QL Calls](#)
- [Enterprise Information System \(EIS\) Interactions](#)

You can also execute a `Call` in the context of a `DatabaseQuery`. For more information on `DatabaseQuery`, see [Section 108.7, "Database Queries"](#).

---

---

**WARNING:** Allowing an unverified SQL string to be passed into methods (for example: `executeSelectingCall(String sql)` method) makes your application vulnerable to SQL injection attacks.

---

---

## 108.9.1 SQL Calls

SQL calls access fields in a relational database. TopLink supports the following SQL calls:

- [SQLCall](#)
- [StoredProcedureCall](#), including `PLSQLStoreProcedureCall`
- [StoredFunctionCall](#)

Using the `Call` API (or SQL string conventions), you can specify input, output, and input-output parameters and assign values for input and input/output parameters.

Using a descriptor `ReturningPolicy`, you can control whether or not TopLink writes a parameter out, retrieves a value generated by the database, or both. For more information, see [Section 119.27, "Configuring Returning Policy"](#).

### 108.9.1.1 SQLCall

Using a `SQLCall`, you can specify any arbitrary SQL statement and execute it on a data source.

---

---

**WARNING:** Allowing an unverified SQL string to be passed into methods makes your application vulnerable to SQL injection attacks.

---

---

For more information, see [Section 109.4, "Using a SQLCall"](#).

### 108.9.1.2 StoredProcedureCall

A stored procedure is composed of one or more procedural language statements, such as Procedural Language/Structured Query Language (PL/SQL), stored by name in the database. Most relational databases support stored procedures.

You invoke a stored procedure to execute logic and access data from the data source.

Using a `StoredProcedureCall`, you can detect execution errors, specify input parameters, output parameters, and input/output parameters. However, stored procedures do not provide a return value.

For more information, see [Section 109.5, "Using a StoredProcedureCall"](#).

### 108.9.1.3 StoredFunctionCall

A stored function is Oracle Database feature that provides all the functionality of a stored procedure as well as the ability to return a value.

Using a `StoredFunctionCall`, you can specify all the features of a `StoredProcedureCall` as well as the field name of the return value.

For more information, see [Section 109.6, "Using a StoredFunctionCall"](#).

## 108.9.2 EJB QL Calls

In TopLink, EJB QL calls represent EJB QL strings. An `EJBQLCall` object is an abstraction of a database invocation. You can execute an EJB QL call directly from a session or in the context of a `DatabaseQuery`.

For more information, see the following:

- [Section 109.7, "Using Java Persistence Query Language \(JPQL\) Calls"](#)
- [Section 109.2.8, "How to Specify a Custom EJB QL String in a DatabaseQuery"](#)
- [Section 108.2.8.2, "EJB QL Queries"](#)

## 108.9.3 Enterprise Information System (EIS) Interactions

To invoke a query through a Java EE Connector Architecture (JCA) adapter to a remote EIS, you use an `EISInteraction`, an instance of `Call`. TopLink supports the following `EISInteraction` types:

- [IndexedInteraction](#)
- [MappedInteraction](#)
- [XMLInteraction](#)
- [XQueryInteraction](#)
- [QueryStringInteraction](#)

In each of these interactions, you specify a functional interface (similar to a stored procedure) that identifies the function to invoke on the EIS. This functional interface contains the following:

- the function name;
- the record name (if different than the function name);
- a list of input arguments;
- a list of output arguments.

For more information, see the following:

- [Chapter 71, "Introduction to EIS Projects"](#)
- [Section 109.8, "Using EIS Interactions"](#)

### 108.9.3.1 IndexedInteraction

In an `IndexedInteraction`, you exchange data with the EIS using indexed records. The order of the specification of the arguments must match the order of the values defined in the indexed record.

### 108.9.3.2 MappedInteraction

In a `MappedInteraction`, you exchange data with the EIS using mapped records. The arguments you specify map by name to fields in the mapped record.

### 108.9.3.3 XMLInteraction

An `XMLInteraction` is a `MappedInteraction` that maps data to an XML record. For an `XMLInteraction`, you may also provide an optional root element name.

#### 108.9.3.4 XQueryInteraction

If your JCA adapter supports the XQuery dynamic query language, you can use an `XQueryInteraction`, which is an `XMLInteraction` that lets you specify your XQuery string.

#### 108.9.3.5 QueryStringInteraction

If your JCA adapter supports a query string based dynamic query language, you can use a `QueryStringInteraction`, which is a `MappedInteraction` that lets you specify the dynamic query string.

## 108.10 Redirect Queries

To accommodate complex query logic, you can implement a **redirect query**: a named query that delegates query execution control to your application. For more information, see [Section 108.8, "Named Queries"](#).

Redirect queries lets you define the query implementation in code as a static method. When you invoke the query, the call redirects to the specified static method. Redirect queries accept any arbitrary parameters passed into them packaged in a `Vector`.

Although most `TopLink` queries search for objects directly, a redirect query generally invokes a method that exists on another class and waits for the results. Redirect queries let you build and use complex operations, including operations that might not otherwise be possible within the query framework.

By delegating query invocation to a method you provide, redirect queries let you dynamically make decisions about how a query should be executed based on argument values.

Using a redirect query, you can do the following:

- Dynamically configure the query options based on the arguments (for example, ordering and query optimization).
- Dynamically define the selection criteria based on the arguments.
- Pass query-by-example objects or expressions as the arguments.
- Post-process the query results.
- Perform multiple queries or special operations.

If you execute the query on a `UnitOfWork`, the results register with that instance of `UnitOfWork`, so any objects you attempt to retrieve with the `invoke` method must come from the `Session` cache.

To create a redirect query, you implement the `QueryRedirector` interface and set your implementation on a named query.

Oracle recommends that you take advantage of the `MethodBasedQueryRedirector`, an instance of `QueryRedirector` that `TopLink` provides. It takes the name of a static method and the `Class` in which it is defined as parameters. When you set a `MethodBasedQueryRedirector` on a named query, whenever `invokeQuery` method is called on this instance, `TopLink` uses reflection to invoke your static method instead.

The advantages of using a `MethodBasedQueryRedirector` are as follows:

- You can specify the static method and its `Class` dynamically.
- The class that provides the static method does not need to implement `QueryRedirector`.



- Your static method can have any name.
- You can restrict the parameters to your static method to only a `Session` and a `Vector` of arguments.

For more information, see [Section 111.1, "Using Redirect Queries"](#).

## 108.11 Historical Queries

By default, a session represents a view of the most current version of objects and when you execute a query in that session, it returns the most current version of selected objects.

If your data source maintains past or historical versions of objects, you can configure `TopLink` to access this historical data (see [Section 87.6, "Historical Sessions"](#)).

Once you configure `TopLink` to take advantage of this historical data, you can access historical versions using the historical queries that [Table 108–5](#) summarizes.

---



---

**Note:** Flashback queries do not support view selects. This means you cannot use a flashback query on objects with an inheritance policy for read-all-subclasses views. For more information, see [Section 16.3, "Descriptors and Inheritance"](#).

---



---

**Table 108–5** Historical Queries

Historical Query Type	Session	Cache	Must set <code>maintainCache</code> to false?	Query both current and historical versions?
<a href="#">Using an <code>ObjectLevelReadQuery</code> with an <code>AsOfClause</code></a>	Regular <sup>1</sup>	<ul style="list-style-type: none"> <li>■ Global</li> <li>■ Read-only</li> <li>■ Contains current versions</li> </ul>	Yes	No
<a href="#">Using an <code>ObjectLevelReadQuery</code> with <code>Expression Operator asOf</code></a>	Regular <sup>1</sup>	<ul style="list-style-type: none"> <li>■ Global</li> <li>■ Read and write</li> <li>■ Contains current versions</li> </ul>	No	Yes
<a href="#">Using an <code>ObjectLevelReadQuery</code> in a Historical Session</a>	Historical <sup>2</sup>	<ul style="list-style-type: none"> <li>■ Isolated</li> <li>■ Read-only</li> <li>■ Contains static snapshot as of specified time</li> </ul>	No	No

<sup>1</sup> A server or database session based on an `OraclePlatform` for an Oracle9i Database (or later), or based on `TopLink HistoryPolicy`.

<sup>2</sup> A session returned by a server or database session based on an `OraclePlatform` or `TopLink HistoryPolicy` using the `acquireHistoricalSession` method passing in an `AsOfClause`.

### 108.11.1 Using an `ObjectLevelReadQuery` with an `AsOfClause`

You can query historical versions of objects using an `ObjectLevelReadQuery` configured with an `AsOfClause` (set by `ObjectLevelReadQuery` method `setAsOfClause`) that specifies a point in time that applies to every `Expression` used in the query.

This type of historical query lets you query a static snapshot of object versions as of the specified time.

---

---

**Note:** To prevent corrupting the global shared cache with old versions of objects, you must set `ObjectLevelReadQuery` method `maintainCache` to `false` in this historical query. If you do not, `TopLink` will throw an exception when you execute the query.

---

---

For more information and examples of using an `ObjectLevelReadQuery` with an `AsOfClause`, see [Section 111.2, "Using Historical Queries"](#).

### 108.11.2 Using an `ObjectLevelReadQuery` with Expression Operator `asOf`

You can query historical versions of objects using an `ObjectLevelReadQuery` (such as `ReadObjectQuery` or `ReadAllQuery`) containing one or more expressions that use Expression operator `asOf` to specify a point in time on an Expression-by-Expression basis.

This type of historical query lets you combine both current and historical versions of objects in the same query.

If you configure the `ObjectLevelReadQuery` with an `AsOfClause`, that point in time overrides the point in time specified in any Expression in the query (see [Section 108.11.1, "Using an `ObjectLevelReadQuery` with an `AsOfClause`"](#)).

For more information and examples of using an `ObjectLevelReadQuery` with Expression operator `asOf`, see [Section 111.2, "Using Historical Queries"](#).

### 108.11.3 Using an `ObjectLevelReadQuery` in a Historical Session

Given a session that maintains historical versions of objects (based on an appropriate `OraclePlatform` or `TopLinkHistoryPolicy`), you can use `Session` method `acquireHistoricalSession` passing in an `AsOfClause` that specifies a point in time that applies to all queries and expressions.

This method returns a lightweight, read-only snapshot of object versions as of the specified time. The cache used in this type of session is isolated from the global shared cache. You do not need to set `ObjectLevelReadQuery` method `maintainCache` to `false` in this case.

For more information and examples of using an `ObjectLevelReadQuery` with a historical session, see [Section 111.2, "Using Historical Queries"](#).

## 108.12 Interface and Inheritance Queries

When you define an interface descriptor (see [Section 22.2.1.3, "Creating Relational Interface Descriptors"](#)), you can perform queries on interfaces and inheritance hierarchies.

For more information, see the following:

- [Section 111.5, "Querying on Interfaces"](#)
- [Section 111.6, "Querying on an Inheritance Hierarchy"](#)

## 108.13 Descriptor Query Manager Queries

Each `Descriptor` owns an instance of `DescriptorQueryManager` that you can use for the following:

- Configuring named queries (see [Section 108.13.1, "How to Configure Named Queries"](#))
- Configuring default query implementation (see [Section 108.13.2, "How to Configure Default Query Implementations"](#))
- Configuring additional join expressions (see [Section 108.13.3, "How to Configure Additional Join Expressions"](#))

### 108.13.1 How to Configure Named Queries

The `DescriptorQueryManager` provides API for storing and retrieving frequently used queries by name.

For more information, see [Section 108.8, "Named Queries"](#).

### 108.13.2 How to Configure Default Query Implementations

The `DescriptorQueryManager` of each `Descriptor` lets you customize the query implementation that TopLink uses for the following data source operations:

- insert object
- update object
- read object
- read all objects
- delete object

For example, if you need to insert an object using a stored procedure, you can override the default `SQLCall` used by the `DescriptorQueryManager` insert object query.

Whenever you execute a query on a given `Class`, TopLink consults the `DescriptorQueryManager` to determine how to perform the given data source operation.

You can use this capability for a variety of purposes such as to extend TopLink behavior, access nonrelational data, or use stored procedures or customized SQL calls.

---

---

**WARNING:** Allowing an unverified SQL string to be passed into methods makes your application vulnerable to SQL injection attacks.

---

---

For information and examples on customizing these default query implementations, see the following:

- [Section 23.4, "Configuring Custom SQL Queries for Basic Persistence Operations"](#)
- [Section 76.5, "Configuring Custom EIS Interactions for Basic Persistence Operations"](#)

### 108.13.3 How to Configure Additional Join Expressions

You can configure the `DescriptorQueryManager` to automatically append an expression to every query it performs on a class. For example, you can add an expression that filters the data source for the valid instances of a given class.

For more information, see [Section 111.7, "Appending Additional Join Expressions"](#).

## 108.14 Oracle Extensions

When you use TopLink with Oracle Database, you can make use of the following Oracle specific query features from within your TopLink applications:

- [Hints](#)
- [Hierarchical Queries](#)
- [Flashback Queries](#)
- [Stored Functions](#)

### 108.14.1 Hints

Oracle lets you specify SQL query additions called hints that can influence how the database server SQL optimizer works. This lets you influence decisions usually reserved for the optimizer. You use hints to specify things such as join order for a join statement, or the optimization approach for a SQL call.

You specify hints using the `DatabaseQuery` method `setHintString`.

For more information, see the following:

- [Section 108.7, "Database Queries"](#)
- [Section 111.9.1, "How to Use Oracle Hints"](#)
- Your database *Performance Tuning Guide and Reference*.

### 108.14.2 Hierarchical Queries

Oracle Database Hierarchical Queries mechanism lets you select database rows based on hierarchical order. For example, you can design a query that reads the row of a given employee, followed by the rows of people the employee manages, followed by their managed employees, and so on.

You specify a hierarchical query clause using `DatabaseQuery` subclass `ReadAllQuery` method `setHierarchicalQueryClause`. For more information on `DatabaseQuery` queries, see [Section 108.7, "Database Queries"](#).

For more information on configuring a `ReadAllQuery` with an Oracle hierarchical query clause, see [Section 111.9.2, "How to Use Hierarchical Queries"](#).

### 108.14.3 Flashback Queries

When using TopLink with Oracle9i Database (or later), you can acquire a special historical session where all objects are read as of a past time, and then you can express read queries depending on how your objects are changing over time.

For more information, see [Section 108.11, "Historical Queries"](#).

### 108.14.4 Stored Functions

A stored function is Oracle Database mechanism that provides all the capabilities of a stored procedure in addition to returning a value.

For more information, see [Section 108.9.1.3, "StoredFunctionCall"](#).

## 108.15 EJB 2.n CMP Finders

An EJB finder is a query as defined by the EJB specification. It returns EJB, collections, and enumerations. The difference between a finder and a query is that queries return Java objects, but finders return EJB. The TopLink query framework lets you create and execute complex finders that retrieve entity beans.

Finders contain finder methods that define search criteria. The work involved in creating these methods depends on whether you are building container-managed persistence (CMP) bean finders or bean-managed persistence (BMP) bean finders:

- CMP finders require you to define the finder API method signature on the bean Home interface. The CMP provider generates the actual code mechanisms for the finder from the API definition.
- BMP finders require you to provide the code required to execute the finder methods.

In either case, you define finders in the Home interface of the bean.

You can implement finders using any TopLink query feature and you can take advantage of predefined finder implementations that TopLink provides for both CMP and BMP entity beans.

This section describes the following:

- [Predefined Finders](#)
- [Default Finders](#)
- [Call Finders](#)
- [DatabaseQuery Finders](#)
- [Named Query Finders](#)
- [Primary Key Finders](#)
- [Expression Finders](#)
- [EJB QL Finders](#)
- [SQL Finders](#)
- [Redirect Finders](#)
- [The ejbSelect Method](#)

For more information, see [Section 111.10, "Using EJB 2.n CMP Finders"](#).

### 108.15.1 Predefined Finders

TopLink provides predefined finder implementations that provide a rich API that lets you dynamically specify query properties at run time and take full advantage of TopLink query features.

TopLink provides the following predefined finders:

- [Predefined CMP Finders](#)
- [Predefined BMP Finders](#)

For more information, see the following:

- [Section 2.14, "Considering EJB Entity Beans with CMP Architecture"](#)
- [Section 2.15, "Considering EJB Entity Beans with BMP Architecture"](#)

### 108.15.1.1 Predefined CMP Finders

Table 108–6 lists the predefined finders you can use with TopLink CMP (using OC4J). The TopLink runtime reserves the method names listed in Table 108–6.

**Table 108–6** Predefined CMP Finders

Method	Arguments	Return
findAll	()	Collection
findManyByEJBQL	(String ejbql) (String ejbql, Vector arguments)	Collection
findManyByQuery	(DatabaseQuery query) (DatabaseQuery query, Vector arguments)	Collection
findManyBySQL	(String sql) (String sql, Vector arguments)	Collection
findByPrimaryKey	(Object primaryKeyObject)	EJBObject
findOneByEJBQL	(String ejbql) (String ejbql, Vector arguments)	EJBObject
findOneByQuery	(DatabaseQuery query) (DatabaseQuery query, Vector arguments)	EJBObject
findOneBySQL	(String sql) (String sql, Vector arguments)	EJBObject

---

**Note:** If the finder is located on a local home, replace EJBObject with EJBLocalObject in finders that contain findOneBy.

---

You can also use each of these finders without a vector of arguments. For example, EJBObject findOneByEJBQL(String ejbql) is a valid dynamic finder, but you must replace the return type of EJBObject with your bean's component interface.

For more information, see Section 111.10, "Using EJB 2.n CMP Finders".

### 108.15.1.2 Predefined BMP Finders

Table 108–7 lists the predefined finders you can use if you extend your BMP EJB from oracle.toplink.ejb.bmp.BMPEntityBase (see Section 2.15, "Considering EJB Entity Beans with BMP Architecture").

The TopLink runtime reserves the method names listed in Table 108–7.

**Table 108–7** Predefined BMP Finders

Method	Arguments	Return
findAll	() (Call) (Expression) (ReadAllQuery)	Enumeration
findAllByNamedQuery	(String queryName, Vector arguments)	Enumeration
findByPrimaryKey	(Object primaryKeyObject)	Object
findOne	(Call) (Expression) (ReadObjectQuery)	Object

**Table 108-7 (Cont.) Predefined BMP Finders**

Method	Arguments	Return
<code>findOneByNamedQuery</code>	<code>(String queryName, Vector arguments)</code>	Object

For more information about using EJB finders, see [Section 111.10, "Using EJB 2.n CMP Finders"](#).

## 108.15.2 Default Finders

For each finder method defined on the home interface of an entity bean, whose name matches `findBy<CMP-FIELD-NAME>` where `<CMP-FIELD-NAME>` is the name of a persistent field on the bean, TopLink generates a finder implementation including a TopLink query that uses the TopLink expressions framework. If the return type is a single bean type, TopLink creates a `ReadObjectQuery`; if the return type is a `Collection`, TopLink creates a `ReadAllQuery`.

Although you must still define the finder in the entity home, you do not need to declare the finder in the `ejb-jar.xml` file.

For more information, see [Section 111.10.1, "How to Create a Finder"](#).

## 108.15.3 Call Finders

Finders that use a `Call` let you create dynamic queries that you generate at run time rather than at deployment time.

For more information, see the following:

- [Section 108.9, "Call Queries"](#).
- [Section 108.15.1, "Predefined Finders"](#)

## 108.15.4 DatabaseQuery Finders

Finders that use a `DatabaseQuery` lets you create dynamic queries that you generate at run time rather than at deployment time.

In addition to finders that take a `DatabaseQuery`, TopLink also provides a default `findAll` finder that returns all the EJB of a given type. As with other dynamic finders, the TopLink runtime reserves the name `findAll`.

For more information, see [Section 108.7, "Database Queries"](#).

For more information on TopLink predefined finders that take a `DatabaseQuery`, see [Section 108.15.1, "Predefined Finders"](#).

## 108.15.5 Named Query Finders

Finders that use a named `DatabaseQuery` stored in a `DescriptorQueryManager` or `Session` let you efficiently reuse frequently executed queries.

For more information, see the following:

- [Section 108.8, "Named Queries"](#)
- [Section 108.15.1, "Predefined Finders"](#)

## 108.15.6 Primary Key Finders

TopLink provides predefined finder implementations that take a primary key class as a Java Object.

Because the EJB 2.0 and 2.1 specifications requires the container to implement the `findByPrimaryKey` call on each bean Home interface, do not delete this finder from a bean.

For more information, see [Section 108.15.1, "Predefined Finders"](#).

## 108.15.7 Expression Finders

Using a finder based on a TopLink Expression offers the following advantages:

- Version-controlled standardized queries in Java code.
- Ability to simplify most complex operations.
- A more complete set of querying features than is available through EJB QL.

Because expressions lets you specify finder search criteria based on the object model, they are frequently the best choice for constructing your finders.

For more information, see [Section 108.2.6, "TopLink Expressions"](#).

For more information on TopLink predefined finders that take an Expression, see [Section 108.15.1, "Predefined Finders"](#).

You can also use an Expression in a finder that takes a DatabaseQuery by using DatabaseQuery method `setSelectionCriteria`. For more information on TopLink predefined finders that take a DatabaseQuery, see [Section 108.15.4, "DatabaseQuery Finders"](#).

## 108.15.8 EJB QL Finders

TopLink supports EJB QL. EJB QL finders let you specify an EJB QL string as the implementation of the query.

EJB QL offers the following advantages:

- It is the EJB 2.0 and 2.1 standard for queries.
- You can use it to construct most queries.
- You can implement dependent-object queries with EJB QL.

The disadvantage of EJB QL is that it is difficult to use when you construct complex queries.

For more information about EJB QL support in TopLink, see [Section 108.2.8, "Query Languages"](#).

For more information on TopLink predefined finders that take EJB QL, see [Section 108.15.1, "Predefined Finders"](#).

## 108.15.9 SQL Finders

Using SQL to define a finder offers the following advantages:

- You can implement logic that cannot be expressed when you use EJB QL or a TopLink Expression.
- It allows for the use of a stored procedure instead of TopLink generated SQL.



- There may be cases in which custom SQL will improve performance.

SQL finders also have the following disadvantages:

- Writing complex custom SQL statements requires a significant maintenance effort if the database tables change.
- Hard-coded SQL limits portability to other databases.
- No validation is performed on the SQL string. Errors in SQL statements will not be detected until run time.
- The use of SQL for a function other than `SELECT` may result in unpredictable errors.

For more information on TopLink predefined finders that take SQL, see [Section 108.15.1, "Predefined Finders"](#).

### 108.15.10 Redirect Finders

Redirect finders enable you to implement a finder that is defined on an arbitrary helper class as a static method. When you invoke the finder, TopLink redirects the call to the specified static method.

Redirect queries are complex and require an extra helper method to define the query. However, because they support complex logic, they are often the best choice when you need to implement logic unrelated to the bean on which the redirect method is called.

For more information, see the following:

- [Section 108.10, "Redirect Queries"](#)
- [Section 111.10, "Using EJB 2.n CMP Finders"](#)

### 108.15.11 The `ejbSelect` Method

The `ejbSelect` method is a query method intended for internal use within an entity bean instance. Specified on the abstract bean itself, the `ejbSelect` method is not directly exposed to the client in the home or component interface. Defined as abstract, each bean can include zero or more such methods.

`ejbSelect` methods have the following characteristics:

- The method name must have `ejbSelect` as its prefix.
- It must be declared as public.
- It must be declared as abstract.
- The `throws` clause must specify the `javax.ejb.FinderException`, although it may also specify application-specific exceptions as well.
- The `result-type-mapping` tag in the `ejb-jar.xml` file determines the return type for `ejbSelect` methods. Set the flag to `Remote` to return `EJBObjects`; set it to `Local` to return `EJBLocalObjects`.

The format for an `ejbSelect` method definition looks as follows:

```
public abstract type.ejbSelect<METHOD>(...);
```

The `ejbSelect` query return type is not restricted to the entity bean type on which the `ejbSelect` is invoked. Instead, it can return any type corresponding to a container-managed relationship or container-managed field, with the following exception: In the case that the `ejbSelect` method return type is a

`java.util.Collection`, the result must be the entity type on which the selector was defined.

Although the `select` method is not based on the identity of the entity bean instance on which it is invoked, it can use the primary key of an entity bean as an argument. This creates a query that is logically scoped to a particular entity bean instance.

For more information and examples on using TopLink queries in the `ejbSelect` method, see [Section 111.10, "Using EJB 2.n CMP Finders"](#).

## 108.16 Queries and the Cache

When you execute a query, TopLink retrieves the information from either the database or the TopLink session cache. You can configure the way queries use the TopLink cache to optimize performance.

TopLink maintains a client-side cache to reduce the number of read operations required from the database. TopLink caches objects written to and read from the database to maintain object identity. The sequence in which a query checks the cache and database affects query performance. By default, primary key queries check the cache before accessing the database, and all queries check the cache before rebuilding an object from its row.

---

---

**Note:** You can override the default behavior in the caching policy configuration information in the TopLink descriptor. For more information, see [Section 102.2.4, "Explicit Query Refreshes"](#).

---

---

This section illustrates ways to manipulate the relationship between query and cache, and explains the following:

- [How to Configure the Cache](#)
- [How to Use In-Memory Queries](#)
- [Primary Key Queries and the Cache](#)
- [How to Disable the Identity Map Cache Update During a Read Query](#)
- [How to Refresh the Cache](#)
- [How to Cache Query Results in the Session Cache](#)
- [How to Cache Query Results in the Query Cache](#)
- [How to Use Caching and EJB 2.n CMP Finders](#)

### 108.16.1 How to Configure the Cache

The cache in a TopLink application holds objects that have already been read from or written to the database. Use of the cache in a TopLink application reduces the number of accesses to the database. Because accessing the database consumes time and resources, an effective caching strategy is important to the efficiency of your application.

For more information about configuring and using the cache, see [Chapter 102, "Introduction to Cache"](#).

## 108.16.2 How to Use In-Memory Queries

An in-memory query is a query that is run against the shared session cache. Careful configuration of in-memory querying improves performance, but not all queries benefit from in-memory querying. For example, queries for individual objects based on primary keys generally see performance gains from in-memory querying; queries not based on primary keys are less likely to benefit.

By default, queries that look for a single object based on primary keys attempt to retrieve the required object from the cache first, and then to search the database if the object is not in the cache. All other query types search the database first, by default. You can specify whether a given query runs against the in-memory cache, the database, or both.

In-memory querying lets you perform queries on the cache rather than the database.

---

---

**Note:** You cannot expect an ordered result from an in-memory query as ordering is not supported for these queries.

---

---

In-memory querying supports the following relationships:

- One-to-one
- One-to-many
- Many-to-many
- Aggregate collection
- Direct collection

---

---

**Note:** By default, the relationships themselves must be in memory for in-memory traversal to work. Ensure that you trigger all value holders to enable in-memory querying to work across relationships.

---

---

This section describes the following:

- [Configuring Cache Usage for In-Memory Queries](#)
- [Expression Options for In-Memory Queries](#)
- [Handling Exceptions Resulting from In-Memory Queries](#)

### 108.16.2.1 Configuring Cache Usage for In-Memory Queries

You can configure in-memory query cache usage at the query level using `ReadObjectQuery` and `ReadAllQuery` methods:

- `checkCacheByPrimaryKey`: The default setting; if a read-object query contains an expression that compares at least the primary key, you can obtain a cache hit if you process the expression against the objects in memory.
- `checkCacheByExactPrimaryKey`: If a read-object query contains an expression where the primary key is the only comparison, you can obtain a cache hit if you process the expression against the object in memory.
- `checkCacheThenDatabase`: You can configure any read-object query to check the cache completely before you resort to accessing the database.

- `checkCacheOnly`: You can configure any read-all query to check only the parent session cache (not the unit of work cache) and return the result from the parent session cache without accessing the database.
- `conformResultsInUnitOfWork`: You can configure any read-object or read-all query within the context of a unit of work to conform the results with the changes to the object made within that unit of work. This includes new objects, deleted objects and changed objects. For more information and limitations on conforming, see [Section 115.4, "Using Conforming Queries and Descriptors"](#).

Alternatively, you can configure cache usage using the `ObjectLevelReadQuery` method `setCacheUsage`, passing in the appropriate `ObjectLevelReadQuery` field: `CheckCacheByPrimarykey`, `CheckCacheByExactPrimarykey`, `CheckCacheThenDatabase`, `CheckCacheOnly`, `ConformResultsInUnitOfWork`, or `DoNotCheckCache`.

### 108.16.2.2 Expression Options for In-Memory Queries

You can use a subset of `Expression` (see [Table 108–8](#)) and `ExpressionMath` (see [Table 108–9](#)) methods with in-memory queries. For more information about these options, see [Chapter 110, "Introduction to TopLink Expressions"](#).

**Table 108–8 Expressions Operator Support for In-Memory Queries**

Expressions Operator	In-Memory Query Support
<code>addMonths</code>	
<code>and</code>	✓
<code>anyof<sup>1</sup></code>	✓
<code>anyofAllowingNone<sup>1</sup></code>	✓
<code>asciiValue</code>	
<code>between</code>	✓
<code>concat</code>	✓
<code>currentDate</code>	
<code>dateToString</code>	
<code>decode</code>	
<code>equal</code>	✓
<code>get<sup>1</sup></code>	✓
<code>getAllowingNull<sup>1</sup></code>	✓
<code>getFunction</code>	
<code>greaterThan</code>	✓
<code>greaterThanEqual</code>	✓
<code>hexToRaw</code>	
<code>ifNull</code>	
<code>in</code>	✓
<code>isNull</code>	✓
<code>lastDay</code>	

**Table 108–8 (Cont.) Expressions Operator Support for In-Memory Queries**

<b>Expressions Operator</b>	<b>In-Memory Query Support</b>
leftPad	
leftTrim	
length	✓
lessThan	✓
lessThanEqual	✓
like	✓
monthsBetween	
newTime	
nextDay	
notBetween	✓
notIn	✓
notNull	✓
or	✓
ref	
replace	
rightPad	
rightTrim	
subQuery	
substring	✓
toCharacter	
toDate	
toLowerCase	✓
toNumber	✓
toUpperCase	✓
toUpperCasedWords	
translate	
trim	✓
truncateDate	

<sup>1</sup> For more information, see [Section 108.7.1.5, "Join Reading and Object-Level Read Queries"](#).

**Table 108–9 ExpressionMath Operator Support for In-Memory Queries**

<b>ExpressionMath Operator</b>	<b>In-Memory Query Support</b>
abs	✓
acos	✓
add	✓

**Table 108–9 (Cont.) ExpressionMath Operator Support for In-Memory Queries**

ExpressionMath Operator	In-Memory Query Support
asin	✓
atan	✓
atan2	
ceil	✓
chr	
cos	✓
cosh	
exp	✓
floor	✓
ln	
log	✓
max	✓
min	✓
mod	
none	
power	✓
round	✓
sign	
sin	✓
sinh	
sqrt	✓
subtract	✓
tan	✓
tanh	
trunc	

**108.16.2.3 Handling Exceptions Resulting from In-Memory Queries**

In-memory queries may fail for several reasons, the most common of which are the following:

- The query expression is too complex to execute in memory.
- There are untriggered value holders in which indirection (lazy loading) is used. All object models that use indirection must first trigger value holders before they conform on the relevant objects.

TopLink provides a mechanism to handle indirection exceptions. To specify how the application must handle these exceptions, use the following InMemoryQueryIndirectionPolicy methods:

- `throwIndirectionException`: The default setting; it is the only setting that throws indirection exceptions.

- `triggerIndirection`: Triggers all valueholders to eliminate the problem.
- `ignoreIndirectionExceptionReturnConformed`: Returns conforming if an untriggered value holder is encountered. That is, results from the database are expected to conform, and an untriggered value holder is taken to mean that the underlying attribute has not changed.
- `ignoreIndirectionExceptionReturnNotConformed`: Returns not conforming if an untriggered value holder is encountered.

---

**Note:** When you build new applications, consider throwing all conform exceptions. This provides more detailed feedback for unsuccessful in-memory queries. For more information, see [Section 115.16.4.2, "Handling Exceptions During Conforming"](#).

---

### 108.16.3 Primary Key Queries and the Cache

When a query searches for a single object by a primary key, TopLink extracts the primary key from the query and attempts to return the object from the cache without accessing the database. If the object is not in the cache, the query executes against the database, builds the resulting object(s), and places it in the identity map.

If the query is based on a nonprimary key selection criteria or is a read-all query, the query executes against the database (unless you are using `ReadObjectQuery` or `ReadAllQuery` method `checkCacheOnly`). The query matches primary keys from the result set to objects in the cache, and returns the cached objects, if any, in the result set.

If an object is not in the cache, TopLink builds the object. If the query is a refreshing query, TopLink updates the contents of any objects with the results from the query. Use "equals" on the object identity to properly configure and use an identity map.

Clients can refresh objects when they want to ensure that they have the latest data at a particular time.

#### Traversing Relationships with Compound Primary Keys

When getting objects by using compound primary keys to traverse relationships, you must create use query keys (see [Section 110.4, "Query Keys and Expressions"](#)). By adding a query key for each mapped attribute in a class with a complex primary key, TopLink can use the primary key on the cache.

Consider the class `MyClass` with two attributes: A and B. Both A and B are mapped as 1:1 mappings to the database and designated primary keys.

You should create a query key for each attribute (such as `MyQueryKeyA` and `MyQueryKeyB`) that will map the attributes of the primary key of `MyClass` *without* going through the other classes. You can then use the query key to find the object in the cache and query the object's primary key:

```
builder.get("MyQueryKeyA").equal(new Long("123456"));
```

### 108.16.4 How to Disable the Identity Map Cache Update During a Read Query

To disable the identity map cache update, which is normally performed by a read query, call the `dontMaintainCache` method. This improves the query performance when you read objects that are not needed later by the application and can avoid exceptions during partial object queries (see [Section 109.2.1.2, "Reading Objects Using Partial Object Queries"](#)).

[Example 108–1](#) demonstrates how code reads `Employee` objects from the database and writes the information to a file.

**Example 108–1 Disabling the Identity Map Cache Update**

```
// Reads objects from the employee table and writes them to an employee file
void writeEmployeeTableToFile(String filename, Session session) {
    Vector employeeObjects;
    // Create ReadAllQuery and set Employee as its reference class
    ReadAllQuery query = new ReadAllQuery(Employee.class);
    ExpressionBuilder builder = query.getExpressionBuilder();
    query.setSelectionCriteria(builder.get("id").greaterThan(100));
    query.dontMaintainCache();
    Vector employees = (Vector) session.executeQuery(query);
    // Write all the employee data to a file
    Employee.writeToFile(filename, employees);
}
```

## 108.16.5 How to Refresh the Cache

You can refresh objects in the cache to ensure that they are current with the database, while preserving object identity. This section describes how to use query API to perform the following:

- Configure query refreshing at the descriptor level (see [Section 119.9, "Configuring Cache Refreshing"](#)) to apply cache refreshing to all queries of a particular object type. Before configuring cache refresh options, consider their effect on performance (see [Section 12.10, "Optimizing Cache"](#)).

### 108.16.5.1 Object Refresh

To refresh objects in the cache with the data in the database, call the `Session` method `refreshObject` or the `ReadObjectQuery` method `setShouldRefreshIdentityMapResult(true)`.

### 108.16.5.2 Cascading Object Refresh

You can control the depth at which a refreshing updates objects and their related objects. There are the following three options:

1. `CascadePrivateParts`: Default refresh behavior. Refreshes the local level object and objects that are referenced in privately owned relationships.
2. `CascadeNone`: Refreshes only the first level of the object, but does not refresh related objects.
3. `CascadeAll`: Refreshes the entire object tree, stopping when it reaches leaf objects.
4. `CascadeMapping`: Refreshes each mapping that is configured to cascade refresh.

### 108.16.5.3 Refreshing the Identity Map Cache During a Read Query

Include the `refreshIdentityMapResult` method in a query to force refreshing of an identity map with the results of the query, as the following example shows:

**Example 108–2 Refreshing the Result of a Query in the Identity Map Cache During a Read Query**

```
// Create ReadObjectQuery and set Employee as its reference class
ReadObjectQuery query = new ReadObjectQuery(Employee.class);
```



```

ExpressionBuilder builder = query.getExpressionBuilder();
query.setSelectionCriteria(builder.get("lastName").equal("Smith"));
query.refreshIdentityMapResult();
Employee employee = (Employee) session.executeQuery(query);

```

The `refreshIdentityMapResult` method refreshes the object's attributes, but not the attributes of its privately owned parts. However, under most circumstances, you should refresh an object's privately owned parts and other related objects to ensure consistency with the database.

To refresh privately owned or related parts, use the following methods:

- `cascadePrivateParts`: Refreshes all privately owned objects
- `cascadeAllParts`: Refreshes all related objects

#### **Example 108–3 Using the `cascadePrivateParts` Method**

```

ReadAllQuery query = new ReadAllQuery(Employee.class);
query.refreshIdentityMapResult();
query.cascadePrivateParts();
Vector employees = (Vector) session.executeQuery(query);

```

---



---

**Note:** If the object is in the session cache, you can also use the `refreshObject` method to refresh an object and its privately owned parts.

---



---

## 108.16.6 How to Cache Query Results in the Session Cache

By default, TopLink stores query results in the session cache enabling TopLink to execute the query repeatedly, without accessing the database. This is useful when you execute queries that run against static data.

By default, a read-all query always goes to the database, as it does not know how many objects it is seeking. However if the object already exists in the cache, time can be saved by not having to build a new object from the row.

For more information, see [Chapter 102, "Introduction to Cache"](#).

## 108.16.7 How to Cache Query Results in the Query Cache

In addition to TopLink's object cache, TopLink also supports a query cache. There is the following distinction between the two:

- The **object cache** indexes objects by their primary key, allowing primary key queries to obtain cache hits. By using the object cache, queries that access the data source can avoid the cost of building the objects and their relationships if the object is already present.
- The **query cache** is distinct from the object cache. The query cache is indexed by the query and the query parameters—not the object's primary key. This allows for any query executed with the same parameters to obtain a query cache hit and return the same result set.

By default, a `ReadQuery` does not cache its query result set. You can, however, configure the query to cache its result set. This is useful for frequently executed queries whose result set infrequently changes. The query cache always maintains hard references to the result set; the number of results sets for distinct parameters stored in the query cache is configurable. The query cache maintains its size number of the last executed queries with distinct parameters.

For more information, see [Section 111.13.1, "How to Cache Results in a ReadQuery"](#).

You can apply a cache invalidation policy to the query's internal cache (see [Section 111.13.2, "How to Configure Cache Expiration at the Query Level"](#)). For more information, see [Section 102.2.5, "Cache Invalidation"](#).

### 108.16.7.1 Internal Query Cache Restrictions

TopLink does not support the use of the query cache with cursors: if you use query caching with cursors, TopLink will throw an exception. For information on cursor query results, see [Section 108.5.3, "Stream and Cursor Query Results"](#) and [Section 111.11, "Handling Cursor and Stream Query Results"](#).

## 108.16.8 How to Use Caching and EJB 2.n CMP Finders

TopLink caches enterprise beans that EJB finders retrieve. For your application, you can configure the caching of the EJB finders' results in a variety of ways, force the cache to be refreshed, or disable the caching.

This section describes the following:

- [Caching Options](#)
- [Disabling Cache for Returned Finder Results](#)
- [Refreshing Finder Results](#)

### 108.16.8.1 Caching Options

You can apply various configurations to the underlying query to achieve the correct caching behavior for the application. There are several ways to control the caching options for queries. For most queries, you can set caching options using Oracle JDeveloper or TopLink Workbench.

You can set the caching options on a per-finder basis. [Table 108–10](#) lists the valid values.

**Table 108–10 Finder Caching Options**

This Setting . . .	Causes Finders to . . .	When the Search Involves a Finder That . . .
ConformResultsInUnitOfWork <sup>1</sup>	Check the unit of work cache before querying the session cache or the database. The finder's results always conform to uncommitted new, deleted, and changed objects.	Returns either a single bean or a collection.
DoNotCheckCache	Query the database, bypassing the TopLink internal caches.	Returns either a single bean or a collection.
CheckCacheByExactPrimaryKey	Check the session cache for the object.	Contains only a primary key, and returns a single bean.
CheckCacheByPrimaryKey	Check the session cache for the object.	Contains a primary key (and may contain other search parameters), and returns a single bean.
CheckCacheThenDatabase	Search the session cache before accessing the database.	Returns a single bean.
CheckCacheOnly	Search the parent session cache only (not the unit of work cache), but not the database.	Returns either a single bean or a collection.

<sup>1</sup> Default.

For more information about the TopLink queries, as well as the TopLink unit of work and how it integrates with JTS, see [Chapter 113, "Introduction to TopLink Transactions"](#).

---

**Note:** To apply caching options to finders with manually created queries (`findOneByQuery`, `findManyByQuery`), use the TopLink API.

---

### 108.16.8.2 Disabling Cache for Returned Finder Results

By default, TopLink adds all returned objects to the session cache. However, if you know the set of returned objects is very large, and you want to avoid the expense of storing these objects, you can disable this behavior. To override the default configuration, implement the `dontMaintainCache` method on the query, or disable returned object caching for the query in Oracle JDeveloper or TopLink Workbench.

### 108.16.8.3 Refreshing Finder Results

A finder may return information from the database for an object whose primary key is already in the cache. When set to `true`, the Refresh Cache option (in Oracle JDeveloper and TopLink Workbench) causes the query to refresh the object's nonprimary key attributes with the returned information. This occurs on `findByPrimaryKey` finders as well as all expression and SQL finders for the bean.

If you build a query in Java code, you can set this option by including the `refreshIdentityMapResult` method. This method automatically cascades changes to privately owned parts of the beans. If you require different behavior, configure the query using a dynamic finder instead.

---

**Note:** When you invoke this option from within a transaction, the refresh action overwrites object attributes, including any that have not yet been written to the database.

---

If your application includes an `OptimisticLock` field, use the refresh cache option in conjunction with the `onlyRefreshCacheIfNewerVersion` option. This ensures that the application refreshes objects in the cache only if the version of the object in the database is newer than the version in the cache.

For finders that have no refresh cache setting, the `onlyRefreshCacheIfNewerVersion` method has no effect.

## 108.17 Query API

[Table 108–11](#) summarizes the query support provided by each type of session. For each session type, it shows the type of query operation (create, read, update, delete) that you can perform and whether or not you can execute a `DatabaseQuery` or `Call`. For example, using a unit of work, you can use session queries to read and delete; using a server session, you can use session queries to create, read, update, and delete.

**Table 108–11** Session Query API Summary

Session	Create	Read	Update	Delete	Execute Database Query	Execute Call
Unit of work		✓		✓	✓	✓

**Table 108–11 (Cont.) Session Query API Summary**

Session	Create	Read	Update	Delete	Execute Database Query	Execute Call
Database	✓	✓	✓	✓	✓	✓
Server						
Client		✓			✓	✓

[Example 108–4](#) summarizes the important TopLink packages that provide query and expression support:

**Example 108–4 Query and Expression Packages**

```
oracle.toplink.queryframework  
oracle.toplink.expressions  
oracle.toplink.querykeys  
oracle.toplink.descriptors.DescriptorQueryManager
```

---

---

## Using Basic Query API

This chapter explains essential TopLink query API calls that you are most likely to use throughout the development cycle:

This chapter includes the following sections:

- [Using Session Queries](#)
- [Using DatabaseQuery Queries](#)
- [Using Named Queries](#)
- [Using a SQLCall](#)
- [Using a StoredProcedureCall](#)
- [Using a StoredFunctionCall](#)
- [Using Java Persistence Query Language \(JPQL\) Calls](#)
- [Using EIS Interactions](#)
- [Handling Exceptions](#)
- [Handling Collection Query Results](#)
- [Handling Report Query Results](#)

For more information, see [Chapter 111, "Using Advanced Query API"](#).

### 109.1 Using Session Queries

This section provides examples of how to use the session query methods for the following:

- [How to Read Objects with a Session Query](#)
- [How to Create, Update, and Delete Objects with a Session Query](#)

---

---

**Note:** Oracle recommends that you perform all data source operations using a unit of work: doing so is the most efficient way to manage transactions, concurrency, and referential constraints. For more information, see [Chapter 113, "Introduction to TopLink Transactions"](#).

---

---

For more information, see [Section 108.6, "Session Queries"](#).

## 109.1.1 How to Read Objects with a Session Query

Using the session query API, you can perform the following read operations:

- [Reading an Object with a Session Query](#)
- [Reading All Objects with a Session Query](#)
- [Refreshing an Object with a Session Query](#)

### 109.1.1.1 Reading an Object with a Session Query

The `readObject` method retrieves a single object from the database. The application must specify the class of object to read. If no object matches the criteria, a null value is returned.

For example, the basic read operation is:

```
session.readObject(MyDomainObject.class);
```

This example returns the first instance of `MyDomainObject` found in the table used for `MyDomainObject`. `TopLink` provides the `Expression` class to specify querying parameters for a specific object.

When you search for a single, specific object using a primary key, the `readObject` method is more efficient than the `readAllObjects` method, because `readObject` can find an instance in the cache without accessing database. Because a `readAllObjects` method does not know how many objects match the criteria, it always searches the database to find matching objects, even if it finds matching objects in the cache.

#### **Example 109–1** *readObject Using an Expression*

```
import oracle.toplink.sessions.*;
import oracle.toplink.expressions.*;

// Use an expression to read in the employee whose last name is Smith. Create an
// expression using the Expression Builder and use it as the selection criterion
// of the search
Employee employee = (Employee) session.readObject(Employee.class,
    new ExpressionBuilder().get("lastName").equal("Smith"));
```

### 109.1.1.2 Reading All Objects with a Session Query

The `readAllObjects` method retrieves a `List` of objects from the database and does not put the returned objects in order. If the query does not find any matching objects, it returns an empty `List`.

Specify the class for the query. You can also include an expression to define more complex search criteria, as illustrated in [Example 109–2](#).

#### **Example 109–2** *readAllObjects Using an Expression*

```
// Returns a List of employees whose employee salary is greater than 10000
List employees = session.readAllObjects(Employee.class,
    new ExpressionBuilder().get("salary").greaterThan(10000));
```

### 109.1.1.3 Refreshing an Object with a Session Query

The `refreshObject` method causes `TopLink` to update the object in memory using data from the database. This operation refreshes any privately owned objects as well.

---



---

**Note:** A privately owned object is one that cannot exist without its parent, or source object.

---



---

## 109.1.2 How to Create, Update, and Delete Objects with a Session Query

Using the session query API, you can perform the following create, update, and delete operations:

- [Writing a Single Object to the Database with a Session Query](#)
- [Writing All Objects to the Database with a Session Query](#)
- [Adding New Objects to the Database with a Session Query](#)
- [Modifying Existing Objects in the Database with a Session Query](#)
- [Deleting Objects in the Database with a Session Query](#)

### 109.1.2.1 Writing a Single Object to the Database with a Session Query

When you invoke the `writeObject` method, the method performs a *does-exist* check to determine whether or not an object exists. If the object exists, `writeObject` updates the object; if it does not exist, `writeObject` inserts a new object.

The `writeObject` method writes privately owned objects in the correct order to maintain referential integrity.

Call the `writeObject` method when you cannot verify that an object exists in the database.

#### **Example 109–3 Writing a Single Object Using `writeObject`**

```
// Create an instance of the employee and write it to the database
Employee susan = new Employee();
susan.setName("Susan");
...
// Initialize the susan object with all other instance variables
session.writeObject(susan);
```

### 109.1.2.2 Writing All Objects to the Database with a Session Query

You can call the `writeAllObjects` method to write multiple objects to the database. The `writeAllObjects` method performs the same *does-exist* check as the `writeObject` method and then performs the appropriate insert or update operations.

#### **Example 109–4 Writing Several Objects Using `writeAllObjects`**

```
// Read a List of all the current employees in the database.
List employees = session.readAllObjects(Employee.class);

// Modify any employee data as necessary
...

// Create a new employee and add it to the list of employees
Employee susan = new Employee();
...
// Initialize the new instance of employee
employees.add(susan);
// Write all employees to the database. The new instance of susan not currently in
// the database will be inserted. All the other employees currently stored in the
// database will be updated
```

```
session.writeAllObjects(employees);
```

### 109.1.2.3 Adding New Objects to the Database with a Session Query

The `insertObject` method creates a new object in the database, but does not perform the *does-exist* check before it attempts the insert operation. The `insertObject` method is more efficient than the `writeObject` method if you are certain that the object does not yet exist in the database. If the object does exist, the database throws an exception when you execute the `insertObject` method.

### 109.1.2.4 Modifying Existing Objects in the Database with a Session Query

The `updateObject` method updates existing objects in the database, but does not perform the *does-exist* check before it attempts the update operation. The `updateObject` is more efficient than the `writeObject` method if you are certain that the object does exist in the database. If the object does not exist, the database throws an exception when you execute the `updateObject` method.

### 109.1.2.5 Deleting Objects in the Database with a Session Query

To delete a `TopLink` object from the database, read the object from the database and then call the `deleteObject` method. This method deletes both the specified object and any privately owned data.

## 109.2 Using DatabaseQuery Queries

This section describes creating and executing `DatabaseQuery` queries to perform a variety of basic persistence operations, including the following:

- [How to Read Objects Using a DatabaseQuery](#)
- [How to Create, Update, and Delete Objects with a DatabaseQuery](#)
- [How to Update and Delete Multiple Objects with a DatabaseQuery](#)
- [How to Read Data with a DatabaseQuery](#)
- [How to Update Data with a DatabaseQuery](#)
- [How to Specify a Custom SQL String in a DatabaseQuery](#)
- [How to Specify a Custom JPQL String in a DatabaseQuery](#)
- [How to Specify a Custom EJB QL String in a DatabaseQuery](#)
- [How to Use Parameterized SQL and Statement Caching in a DatabaseQuery](#)

### 109.2.1 How to Read Objects Using a DatabaseQuery

This section provides examples that illustrate how to read objects using a `DatabaseQuery`, including the following:

- [Performing Basic DatabaseQuery Read Operations](#)
- [Reading Objects Using Partial Object Queries](#)
- [Reading Objects Using Report Queries](#)
- [Reading Objects Using Query-By-Example](#)
- [Specifying Read Ordering](#)
- [Specifying a Collection Class](#)



- [Specifying the Maximum Rows Returned](#)
- [Configuring Query Timeout at the Query Level](#)
- [Using Batch Reading](#)
- [Using Join Reading with ObjectLevelReadQuery](#)

### 109.2.1.1 Performing Basic DatabaseQuery Read Operations

[Example 109–5](#) illustrates a simple read query. It uses a TopLink expression, but does not use its own arguments for the query. Instead, it relies on the search parameters the expression provides. This example builds the expression within its code, but does not register the query with the session.

#### **Example 109–5 Simple ReadAllQuery**

```
// This example returns a List of employees whose employee ID is > 100

// Initialize the DatabaseQuery by specifying the query type
// and set the reference class for the query
ReadAllQuery query = new ReadAllQuery(Employee.class);

// Retrieve ExpressionBuilder from the query
ExpressionBuilder builder = query.getExpressionBuilder();

// Configure the query execution. Because this example uses an expression,
// it uses the setSelectionCriteria method
query.setSelectionCriteria(builder.get("id").greaterThan(100));

// Execute the query
List employees = (List) session.executeQuery(query);
```

[Example 109–6](#) illustrates a complex readObject query that uses all available configuration options.

#### **Example 109–6 Named Read Query with Two Arguments**

```
// Initialize the DatabaseQuery by specifying the query type
// and set the reference class for the query
ReadObjectQuery query = new ReadObjectQuery(Employee.class);
// Retrieve ExpressionBuilder from the query
ExpressionBuilder builder = query.getExpressionBuilder();
// Define two expressions that map to the first and last names of the employee
Expression firstNameExpression =
emp.get("firstName").equal(emp.getParameter("firstName"));
Expression lastNameExpression =
emp.get("lastName").equal(emp.getParameter("lastName"));

// Configure the query execution. Because this example uses an expression,
// it uses the setSelectionCriteria method
query.setSelectionCriteria(firstNameExpression.and(lastNameExpression));
// Specify the required arguments for the query
query.addArgument("firstName");
query.addArgument("lastName");

// Add the query to the session
session.addQuery("getEmployeeWithName", query);

// Execute the query by referencing its name and providing values
// for the specified arguments
Employee employee =
```

```
(Employee)session.executeQuery("getEmployeeWithName", "Bob", "Smith");
```

### 109.2.1.2 Reading Objects Using Partial Object Queries

[Example 109–7](#) demonstrates the use of partial object reading. It reads only the last name and primary key for the employees. This reduces the amount of data read from the database.

#### **Example 109–7 Using Partial Object Reading**

```
/* Read all the employees from the database, ask the user to choose one and return
it. This uses partial object reading to read just the last name of the employees.
Since TopLink automatically includes the primary key of the object, the full
object can easily be read for editing */
List list;
// Fetch data from database and add to list box
ReadAllQuery query = new ReadAllQuery(Employee.class);
query.addPartialAttribute("lastName");
// The next line avoids a query exception
query.dontMaintainCache();
List employees = (List) session.executeQuery(query);
list.addAll(employees);

// Display list box
....
// Get selected employee from list
Employee selectedEmployee = (Employee)session.readObject(list.getSelectedItem());
return selectedEmployee;
```

### 109.2.1.3 Reading Objects Using Report Queries

[Example 109–8](#) reports the total and average salaries for Canadian employees grouped by their city.

#### **Example 109–8 Querying Reporting Information on Employees**

```
ExpressionBuilder emp = new ExpressionBuilder();
ReportQuery query = new ReportQuery(Employee.class, emp);
query.addMaximum("max-salary", emp.get("salary"));
query.addAverage("average-salary", emp.get("salary"));
query.addAttribute("city", emp.get("address").get("city"));

query.setSelectionCriteria(emp.get("address").get("country").equal("Canada"));
query.addOrdering(emp.get("address").get("city"));
query.addGrouping(emp.get("address").get("city"));
List reports = (List) session.executeQuery(query);
```

The `ReportQuery` class provides an extensive reporting API, including methods for computing average, maximum, minimum, sum, standard deviation, variance, and count of attributes. For more information about the available methods for the `ReportQuery`, see the *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.

---

---

**Note:** Because `ReportQuery` inherits from `ReadAllQuery`, it also supports most `ReadAllQuery` properties.

---

---

### 109.2.1.4 Reading Objects Using Query-By-Example

Query-by-example enables you to specify query selection criteria in the form of a sample object instance that you populate with only the attributes you want to use for the query.

To define a query-by-example, provide a `ReadObjectQuery` or a `ReadAllQuery` with a sample persistent object instance and an optional query-by-example policy. The sample instance contains the data to query, and, optionally, a `QueryByExamplePolicy` (see [Defining a QueryByExamplePolicy](#)) that specifies configuration settings, such as the operators to use and the attribute values to ignore. You can also combine a query-by-example with an expression (see [Combining Query-by-Example and Expressions](#)).

For more information, see [Section 108.2.8.6, "Query-by-Example"](#).

#### **Example 109–9 Using Query-by-Example to Query an Employee**

[Example 109–9](#) queries the employee Bob Smith.

```
Employee employee = new Employee();
employee.setFirstName("Bob");
employee.setLastName("Smith");

// Create a query and set Employee as its reference class
ReadObjectQuery query = new ReadObjectQuery(Employee.class);
query.setExampleObject(employee);

Employee result = (Employee) session.executeQuery(query);
```

#### **Example 109–10 Using Query-by-Example to Query an Employee's Address**

[Example 109–10](#) queries across the employee's address.

```
Employee employee = new Employee();
Address address = new Address();
address.setCity("Ottawa");
employee.setAddress(address);

// Create a query and set Employee as its reference class
ReadAllQuery query = new ReadAllQuery(Employee.class);
query.setExampleObject(employee);

List results = (List) session.executeQuery(query);
```

### **Defining a QueryByExamplePolicy**

TopLink support for query-by-example includes a query-by-example policy. You can edit the policy to modify query-by-example default behavior. You can modify the policy to do the following:

- Use `LIKE` or other operations to compare attributes. By default, query-by-example allows only `EQUALS`.
- Modify the set of values query-by-example ignores (the `IGNORE` set). The default ignored values are zero (0), empty strings, and `FALSE`.
- Force query-by-example to consider attribute values, even if the value is in the `IGNORE` set.
- Use `isNull` or `notNull` for attribute values.

To specify a query-by-example policy, include an instance of `QueryByExamplePolicy` with the query.

#### **Example 109–11 Query-by-Example Policy Using like Operator**

[Example 109–11](#) uses `like` operator for strings and includes only objects whose salary is greater than zero.

```
Employee employee = new Employee();
employee.setFirstName("B%");
employee.setLastName("S%");
employee.setSalary(0);

// Create a query and set Employee as its reference class
ReadAllQuery query = new ReadAllQuery(Employee.class);
query.setExampleObject(employee);
// Query by example policy section adds like and greaterThan
QueryByExamplePolicy policy = new QueryByExamplePolicy();
policy.addSpecialOperation(String.class, "like");
policy.addSpecialOperation(Integer.class, "greaterThan");
policy.alwaysIncludeAttribute(Employee.class, "salary");
query.setQueryByExamplePolicy(policy);
List results = (List) session.executeQuery(query);
```

### **Example 109–12 Query-by-Example Policy Using Keywords**

[Example 109–12](#) uses keywords for strings and ignores the value -1.

```
Employee employee = new Employee();
employee.setFirstName("bob joe fred");
employee.setLastName("smith mc mac");
employee.setSalary(-1);

// Create a query and set Employee as its reference class
ReadAllQuery query = new ReadAllQuery(Employee.class);
query.setExampleObject(employee);
// Query by example policy section
QueryByExamplePolicy policy = new QueryByExamplePolicy();
policy.addSpecialOperation(String.class, "containsAnyKeyWords");
policy.excludeValue(-1);
query.setQueryByExamplePolicy(policy);
List results = (List) session.executeQuery(query);
```

### **Combining Query-by-Example and Expressions**

To create more complex query-by-example queries, combine query-by-example with `TopLink` expressions, as shown in [Example 109–13](#).

### **Example 109–13 Combining Query-by-Example with Expressions**

```
Employee employee = new Employee();
employee.setFirstName("Bob");
employee.setLastName("Smith");

// Create a query and set Employee as its reference class
ReadAllQuery query = new ReadAllQuery(Employee.class);

query.setExampleObject(employee);

// Specify expression
ExpressionBuilder builder = query.getExpressionBuilder();
query.setSelectionCriteria(builder.get("salary").between(100000,200000));
List results = (List) session.executeQuery(query);
```

#### **109.2.1.5 Specifying Read Ordering**

Ordering is a common DatabaseQuery option. You can order a collection of objects returned from a `ReadAllQuery` using the `addOrdering`, `addAscendingOrdering`,

or `addDescendingOrdering` methods. You can apply order based on attribute names, or on query keys and expressions.

**Example 109–14 A Query with Simple Ordering**

```
// Retrieves objects ordered by last name then first name in ascending order
ReadAllQuery query = new ReadAllQuery(Employee.class);
query.addAscendingOrdering ("lastName");
query.addAscendingOrdering ("firstName");
List employees = (List) session.executeQuery(query);
```

**Example 109–15 A Query with Complex Ordering**

```
// Retrieves objects ordered by street address, descending
// case-insensitive order of cities, and manager's last name
ReadAllQuery query = new ReadAllQuery(Employee.class);
ExpressionBuilder emp = query.getExpressionBuilder();
query.addOrdering (emp.getAllowingNull("address").get("street"));
query.addOrdering(
    emp.getAllowingNull("address").get("city").toUpperCase().descending());
query.addOrdering(emp.getAllowingNull("manager").get("lastName"));
List employees = (List) session.executeQuery(query);
```

Note the use of `getAllowingNull`, which creates an outer join for the address and manager relationships. This ensures that employees without an address or manager still appear in the list.

For more information about configuring read ordering, see [Section 119.7.1.4, "Configuring Read All Query Order"](#).

### 109.2.1.6 Specifying a Collection Class

By default, a `ReadAllQuery` returns its result objects in a list. You can configure the query to return the results in any collection class that implements the `Collection` or `Map` interface, as shown in [Example 109–16](#).

**Example 109–16 Specifying the Collection Class for a Collection**

```
ReadAllQuery query = new ReadAllQuery(Employee.class);
query.useCollectionClass(LinkedList.class);
LinkedList employees = (LinkedList) getSession().executeQuery(query);
```

**Example 109–17 Specifying the Collection Class for a Map**

```
ReadAllQuery query = new ReadAllQuery(Employee.class);
query.useMapClass(HashMap.class, "getFirstName");
HashMap employees = (HashMap) getSession().executeQuery(query);
```

### 109.2.1.7 Specifying the Maximum Rows Returned

You can limit a query to a specified maximum number of rows. Use this feature to avoid queries that can return an excessive number of objects.

To specify a maximum number of rows, use the `setMaxRows` method, and pass an integer that represents the maximum number of rows for the query, as shown in [Example 109–18](#).

**Example 109–18 Setting the Maximum Returned Object Size**

```
ReadAllQuery query = new ReadAllQuery(Employee.class);
query.setMaxRows(5);
List employees = (List) session.executeQuery(query);
```

The `setMaxRows` method limits the number of rows the query returns, but does not let you acquire more records after the initial result set.

If you want to browse the result set in fixed increments, use either cursors or censored streams. For more information, see [Section 111.11, "Handling Cursor and Stream Query Results"](#).

### 109.2.1.8 Configuring Query Timeout at the Query Level

You can set the maximum amount of time that TopLink waits for results from a query. This forces a hung or lengthy query to abort after the specified time has elapsed. TopLink throws a `DatabaseException` after the timeout interval.

To specify a timeout interval on a per-query basis, use `DatabaseQuery` method `setQueryTimeout` and pass the timeout interval as an integer representing the number of seconds before the timeout interval should occur, as [Example 109–19](#) shows.

#### **Example 109–19 DatabaseQuery Timeout**

```
// Create the appropriate query and set timeout limits
ReadAllQuery query = new ReadAllQuery(Employee.class);
query.setQueryTimeout(2);
try {
    List employees = (List) session.executeQuery(query);
}
catch (DatabaseException ex) {
    // timeout occurs
}
```

To specify a timeout interval for all queries on a particular object type, configure a query timeout interval at the descriptor level (see [Section 119.8, "Configuring Query Timeout at the Descriptor Level"](#)).

### 109.2.1.9 Using Batch Reading

Batch reading propagates query selection criteria through an object's relationship attribute mappings. You can also nest batch read operations down through complex object graphs. This significantly reduces the number of required SQL select statements and improves database access efficiency.

Consider the following guidelines when you implement batch reading:

- Use batch reading for processes that read in objects and all their related objects.
- Do not enable batch reading for both sides of a bidirectional relationship.
- Avoid nested batch read operations, because they result in multiple joins on the database, slowing query execution.

For more information, see [Section 12.12.9.2, "Reading Case 2: Batch Reading Objects"](#).

For example, in reading  $n$  employees and their related projects, TopLink may require  $n + 1$  select operations. All employees are read at once, but the projects of each are read individually. With batch reading, all related projects can also be read with one select operation by using the original selection criteria, for a total of only two select operations.

To implement batch reading, add the batch read attribute to a query, use the `query.addBatchReadAttribute(Expression anExpression)` API, as the following example shows:

...

```

ReadAllQuery raq = new ReadAllQuery(Trade.class);
ExpressionBuilder tradeBuilder = raq.getBuilder();
...
Expression batchReadProduct = tradeBuilder.get("product");
readAllQuery.addBatchReadAttribute(batchReadProduct);
Expression batchReadPricingDetails = batchReadProduct.get("pricingDetails");
readAllQuery.addBatchReadAttribute(batchReadPricingDetails);
...

```

Alternatively, you can add batch reading at the mapping level for a descriptor. For more information, see [Section 28.5, "Configuring Batch Reading"](#).

You can combine batch reading and indirection (lazy loading) to provide controlled reading of object attributes. For example, if you have one-to-one back pointer relationship attributes, you can defer back pointer instantiation until the end of the query, when all parent and owning objects are instantiated. This prevents unnecessary database access and optimizes TopLink cache use.

#### 109.2.1.10 Using Join Reading with ObjectLevelReadQuery

Use join reading with `ObjectLevelReadQuery` to configure a query for a class to return the data to build an instance of that class and its related objects. For more information, see [Section 108.7.1.5, "Join Reading and Object-Level Read Queries"](#).

To use join reading with an `ObjectLevelReadQuery`, you can use any of Oracle JDeveloper, TopLink Workbench ((see [Section 119.7.1.5, "Configuring Named Query Optimization"](#)), or Java.

---



---

**Note:** You cannot use TopLink Workbench to create an `ObjectLevelReadQuery` with a join expression on a one-to-many mapped attribute: you must use Java.

---



---

**109.2.1.10.1 Using Java** You can use `ObjectLevelReadQuery` API to add joined attributes for mappings.

You can use any of the following API:

- Use the `ObjectLevelReadQuery` method `addJoinedAttribute` with a join expression or attribute name for one-to-one or one-to-many mapped attributes.
 

Using this method, you can add multiple joined attributes, including nested joins. The source and target can be the same class type.

On a one-to-one mapped attribute, use this method to get the class of the `ObjectLevelReadQuery` and the target of the one-to-one mapped attribute of that class with a single database hit.

On a one-to-many mapped attribute, use this method to get the class of the `ObjectLevelReadQuery` and the target collection of the one-to-many mapped attribute of that class with a single database hit.
- Use the `ObjectLevelReadQuery` method `setShouldFilterDuplicates` with a join expression on a one-to-many mapped attribute to filter duplicate rows. This method defaults to `true`.

Use a join expression to configure nested batch reads and inner or outer joins (see [Section 110.2.7, "Expressions for Joining and Complex Relationships"](#)). You can also specify inner or outer joins using the mapping methods `useInnerJoinFetch` or `useOuterJoinFetch`.

[Example 109–20](#) is based on the TopLink ThreeTierEmployee example project. It shows a ReadAllQuery configured to join-read multiple attributes.

**Example 109–20 Join Reading Multiple Attributes**

```
ReadAllQuery query = new ReadAllQuery(Employee.class);

Expression managedEmployees =
    query.getExpressionBuilder().anyOfAllowingNone("managedEmployees");
query.addJoinedAttribute(managedEmployees);
query.addJoinedAttribute(managedEmployees.get("address"));
query.addJoinedAttribute(managedEmployees.anyOf("phoneNumbers"));

List employees = (List) getSession().executeQuery(query);
Use the ObjectLevelReadQuery method
addJoinedAttribute(java.lang.String attributeName) to configure the
query to join-read a single attribute, as Example 109–21 shows.
```

**Example 109–21 Join Reading a Single Attribute**

```
ReadAllQuery query = new ReadAllQuery(Employee.class);
query.addJoinedAttribute("address");
List employees = (List) getSession().executeQuery(query);
```

## 109.2.2 How to Create, Update, and Delete Objects with a DatabaseQuery

You can create, update or delete object with a DatabaseQuery using a DatabaseSession. For more information, see [Section 108.6, "Session Queries"](#).

This section describes the following:

- [Using Write Query](#)
- [Performing Noncascading Write Queries](#)
- [Disabling the Identity Map Cache During a Write Query](#)

### 109.2.2.1 Using Write Query

To execute a write query, use a WriteObjectQuery instance instead of using the writeObject method of the session. Likewise, substitute DeleteObjectQuery, UpdateObjectQuery, and InsertObjectQuery objects for their respective Session methods.

**Example 109–22 Using a WriteObjectQuery**

```
WriteObjectQuery writeQuery = new WriteObjectQuery();
writeQuery.setObject(domainObject);
session.executeQuery(writeQuery);
```

**Example 109–23 Using InsertObjectQuery, UpdateObjectQuery, and DeleteObjectQuery**

```
InsertObjectQuery insertQuery= new InsertObjectQuery();
insertQuery.setObject(domainObject);
session.executeQuery(insertQuery);

// When you use UpdateObjectQuery without a unit of work,
// UpdateObjectQuery writes all direct attributes to the database
UpdateObjectQuery updateQuery= new UpdateObjectQuery();
updateQuery.setObject(domainObject2);
session.executeQuery(updateQuery);
```



```
DeleteObjectQuery deleteQuery = new DeleteObjectQuery();
deleteQuery.setObject(domainObject2);
session.executeQuery(deleteQuery);
```

### 109.2.2.2 Performing Noncascading Write Queries

When you execute a write query, it writes both the object and its privately owned parts to the database by default. To build write queries that do not update privately owned parts, include the `dontCascadeParts` method in your query definition.

Use this method to do the following:

- Increase performance when you know that only the object's direct attributes have changed.
- Resolve referential integrity dependencies when you write large groups of new, independent objects.

---



---

**Note:** Because the unit of work resolves referential integrity internally, this method is not required if you use the unit of work to write to the database.

---



---

#### *Example 109–24 Performing a Noncascading Write Query*

```
// the Employee is an existing employee read from the database
Employee.setFirstName("Bob");
UpdateObjectQuery query = new UpdateObjectQuery();
query.setObject(Employee);
query.dontCascadeParts();
session.executeQuery(query);
```

### 109.2.2.3 Disabling the Identity Map Cache During a Write Query

When you write objects to the database, `TopLink` copies them to the session cache by default. To disable this within a query, call the `dontMaintainCache` method within the query. This improves query performance when you insert objects into the database, but must be used only on objects that will not be required later by the application.

#### *Example 109–25 Disabling the Identity Map Cache During a Write Query*

[Example 109–25](#) reads all the objects from a flat file and writes new copies of the objects into a table.

```
// Reads objects from an employee file and writes them to the employee table
void createEmployeeTable(String filename, Session session) {
    Iterator iterator;
    Employee employee;
    // Read the employee data file
    List employees = Employee.parseFromFile(filename);
    Iterator iterator = employees.iterator();
    while (iterator.hasNext()) {
        Employee employee = (Employee) iterator.next();
        InsertObjectQuery query = new InsertObjectQuery();
        query.setObject(employee);
        query.dontMaintainCache();
        session.executeQuery(query);
    }
}
```

---

---

**Note:** Disable the identity map only when object identity is unimportant in subsequent operations.

---

---

## 109.2.3 How to Update and Delete Multiple Objects with a DatabaseQuery

Using the unit of work, you can perform update and delete operations on multiple objects.

This section describes the following:

- [Using UpdateAll Queries](#)
- [Using DeleteAll Queries](#)

### 109.2.3.1 Using UpdateAll Queries

Use an `UpdateAllQuery` to update a large number of objects at once. With this query, you can update a large number of objects with a single SQL statement, instead of reading the objects into memory and updating them individually. [Example 109–26](#) shows an `UpdateAllQuery` to give all full-time employees a raise.

#### *Example 109–26 Using UpdateAllQuery*

```
// Give all full time employees a 10% raise
UpdateAllQuery updateQuery = new UpdateAllQuery(Employee.class);
ExpressionBuilder employee = updateQuery.getExpressionBuilder();
updateQuery.setSelectionCriteria(employee.get("status").equal("FULL_TIME"));
updateQuery.addUpdateExpression(employee.get("salary"),
    ExpressionMath.multiply(employee.get("salary"), new Float(1.10)));
UpdateAllQuery takes the cache into consideration and ensures that the cache is
kept up to date. You can configure the UpdateAllQuery to invalidate cache (see
Section 102.2.5, "Cache Invalidation") by setting the cache usage to INVALIDATE_
CACHE (default), or to not use the cache by specifying NO_CACHE option. You can
manipulate these settings through the setCacheUsage method. You can only update
the cache for expressions that can conform. For more information on cache, see
Chapter 102, "Introduction to Cache".
```

---

---

**Note:** You can set an attribute within an aggregate only, but not an entire aggregate.

---

---

You can use an `UpdateAll` query with optimistic locking (see [Section 16.4, "Descriptors and Locking"](#)) at the level of updating a row in a database—there should be no updates in the cache. You will update the locking field on the database. There is also support for version and timestamp locking, as well as indirect support for field locking.

### 109.2.3.2 Using DeleteAll Queries

[Example 109–27](#) shows a `DeleteAllQuery` to eliminate all part-time employee positions.

#### *Example 109–27 Using DeleteAllQuery*

```
// Delete all part-time employees
DeleteAllQuery deleteQuery = new DeleteAllQuery(Employee.class);
ExpressionBuilder employee = deleteQuery.getExpressionBuilder();
deleteQuery.setSelectionCriteria(employee.get("status").equal("PART_TIME"));
```

```
deleteQuery.setObjects(domainObjects);
session.executeQuery(deleteQuery);
```

For more information, see [Section 108.7.3.6, "DeleteAllQuery"](#).

## 109.2.4 How to Read Data with a DatabaseQuery

This section describes the following:

- [Using a DataReadQuery](#)
- [Using a DirectReadQuery](#)
- [Using a ValueReadQuery](#)

### 109.2.4.1 Using a DataReadQuery

You can use a `DataReadQuery` to execute a selecting SQL string that returns a `Collection` of the `Record` objects representing the result set, as [Example 109–28](#) shows.

#### *Example 109–28 Using a DataReadQuery*

```
DataReadQuery dataReadQuery = new DataReadQuery();
dataReadQuery.setSQLString("Select * from EMPLOYEE");

// queryResults is a list of Record objects
List queryResults = (List)session.executeQuery(dataReadQuery);
```

### 109.2.4.2 Using a DirectReadQuery

You can use a `DirectReadQuery` to read a single column of data (that is, one field) that returns a `Collection` of the `Record` objects representing the result set, as [Example 109–29](#) shows.

#### *Example 109–29 Using a DirectReadQuery*

```
DirectReadQuery directReadQuery = new DirectReadQuery();
directReadQuery.setSQLString("Select * from EMPLOYEE");

// queryResults is a list of Record objects
List queryResults = (List)session.executeQuery(directReadQuery);
```

### 109.2.4.3 Using a ValueReadQuery

You can use a `ValueReadQuery` to read a single data value (that is, one field). A single data value is returned, or null if no rows are returned, as [Example 109–30](#) shows.

#### *Example 109–30 Using a ValueReadQuery*

```
ValueReadQuery valueReadQuery = new ValueReadQuery();
valueReadQuery.setSQLString("SELECT DISTINCT CURRENT_TIMESTAMP FROM SYSTABLES");

// result is a single Object value
Object result = session.executeQuery(valueReadQuery);
```

---



---

**WARNING:** Allowing an unverified SQL string to be passed into methods (for example: `setSQLString` method) makes your application vulnerable to SQL injection attacks.

---



---

## 109.2.5 How to Update Data with a DatabaseQuery

You can use a `DataModifyQuery` to execute a nonselecting SQL statement (directly or as an `SQLCall`), as [Example 109–31](#) shows. This is equivalent to `Session` method `executeNonSelectingCall` (see [Section 109.4, "Using a SQLCall"](#)).

### **Example 109–31 Using a DataModifyQuery**

```
DataModifyQuery query = new DataModifyQuery(new SQLCall("Delete from Employee"));
session.executeQuery(query);
```

## 109.2.6 How to Specify a Custom SQL String in a DatabaseQuery

All `DatabaseQuery` objects provide a `setSQLString` method that you can use to define a custom SQL string.

For more information about using custom SQL in queries, see [Section 109.4, "Using a SQLCall"](#).

[Example 109–32](#) uses SQL to read all employee IDs.

### **Example 109–32 A Direct Read Query with SQL**

```
DirectReadQuery query = new DirectReadQuery();
query.setSQLString("SELECT EMP_ID FROM EMPLOYEE");
List ids = (List) session.executeQuery(query);
Example 109–33 uses SQL to switch to a different database.
```

### **Example 109–33 A Data Modify Query with SQL**

```
DataModifyQuery query = new DataModifyQuery();
query.setSQLString("USE SALESDATABASE");
session.executeQuery(query);
```

---

---

**WARNING:** Allowing an unverified SQL string to be passed into methods (for example: `setSQLString` method) makes your application vulnerable to SQL injection attacks.

---

---

## 109.2.7 How to Specify a Custom JPQL String in a DatabaseQuery

For information, see "How to Specify a Custom JPQL String in a DatabaseQuery" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_Basic\\_Query\\_API\\_%28ELUG%29#How\\_to\\_Specify\\_a\\_Custom\\_JPQL\\_String\\_in\\_a\\_DatabaseQuery](http://wiki.eclipse.org/Using_Basic_Query_API_%28ELUG%29#How_to_Specify_a_Custom_JPQL_String_in_a_DatabaseQuery)

## 109.2.8 How to Specify a Custom EJB QL String in a DatabaseQuery

All `DatabaseQuery` objects provide a `setEJBQLString` method that you can use to specify a custom EJB QL string.

For JPA queries, see [Section 109.2.7, "How to Specify a Custom JPQL String in a DatabaseQuery"](#).

Provide both a reference class and a `SELECT` clause, and execute the query in the usual manner.

### **Example 109–34 EJB QL**

```
ReadAllQuery query = new ReadAllQuery(EmployeeBean.class);
query.setEJBQLString("SELECT OBJECT(emp) FROM EmployeeBean emp");
```

```
...
List returnedObjects = (List)session.executeQuery(query);

```

[Example 109–35](#) defines the query similarly to [Example 109–34](#), but creates, fills, and passes a vector of arguments to the `executeQuery` method.

**Example 109–35 A Simple ReadAllQuery Using EJB QL and Passing Arguments**

```
// First define the query
ReadAllQuery query = new ReadAllQuery(EmployeeBean.class);
query.setEJBQLString("SELECT OBJECT(emp) FROM EmployeeBean emp WHERE emp.firstName = ?1");
query.addArgument("1", String.class);
...
// Next define the arguments
Vector arguments = new Vector();
arguments.add("Bob");
...
// Finally, execute the query passing in the arguments
List returnedObjects = (List)session.executeQuery(query, arguments);
```

## 109.2.9 How to Use Parameterized SQL and Statement Caching in a DatabaseQuery

By default, TopLink enables parameterized SQL (parameter binding) and statement caching. This causes TopLink to use a prepared statement, binding all SQL parameters and caching the prepared statement. When you reexecute this query, you avoid the SQL preparation, which improves performance.

To disable parameterized SQL and statement caching on individual queries, use DatabaseQuery methods `setShouldBindAllParameters` and `setShouldCacheStatement`, passing in an argument of `false`. To re-enable this feature, pass in an argument of `true`.

**Example 109–36 A Simple ReadObjectQuery with Parameterized SQL**

```
ReadObjectQuery query = new ReadObjectQuery(Employee.class);
query.setShouldBindAllParameters(true);
query.setShouldCacheStatement(true);
```

Alternatively, you can configure parameterized SQL and binding at any of the following levels:

- project level—applies to all named queries (see [Section 20.7, "Configuring Named Query Parameterized SQL and Statement Caching at the Project Level"](#));
- descriptor level—applies on a per-named-query basis (see [Section 119.7.1.9, "Configuring Named Query Options"](#));
- session database login level—applies to all queries (see [Section 98.6, "Configuring JDBC Options"](#)) and provides additional parameter binding API to alleviate the limit imposed by some drivers on SQL statement size;

For more information about using parameterized SQL and binding for data access optimization, see [Section 12.11.5, "How to Use Parameterized SQL \(Parameter Binding\) and Prepared Statement Caching for Optimization"](#).

---

**Note:** For applications using a Java EE data source or external connection pool, you must configure statement caching in the Java EE server's data source—not in TopLink.

---

## 109.3 Using Named Queries

Named queries improve application performance because they are prepared once and they (and all their associated supporting objects) can be efficiently reused thereafter making them well suited for frequently executed operations.

You can configure named queries at the session (see [Section 89.13, "Configuring Named Queries at the Session Level"](#)) or descriptor (see [Section 119.7, "Configuring Named Queries at the Descriptor Level"](#)) level.

For a session-level named query, you can execute the query using any of the following Session API methods:

- `executeQuery(String queryName)`
- `executeQuery(String queryName, arg1)`
- `executeQuery(String queryName, arg1, arg2)`
- `executeQuery(String queryName, arg1, arg2, arg3)`
- `executeQuery(String queryName, Vector args)`

### **Example 109–37 Executing a Session-Level Named Query**

```
Vector args = new Vector();
args.add("Sarah");
Employee sarah = (Employee)session.executeQuery(
    "employeeReadByFirstName",
    args
);
```

For a descriptor-level named query, you can execute the query using any of the following Session API calls, as [Example 109–38](#) shows:

- `executeQuery(String queryName, Class domainClass)`
- `executeQuery(String queryName, Class domainClass, arg1)`
- `executeQuery(String queryName, Class domainClass, arg1, arg2)`
- `executeQuery(String queryName, Class domainClass, arg1, arg2, arg3)`
- `executeQuery(String queryName, Class domainClass, Vector args)`

### **Example 109–38 Executing a Descriptor Level Named Query**

```
Vector args = new Vector();
args.add("Sarah");
Employee sarah = (Employee)session.executeQuery(
    "ReadByFirstName",
    Employee.class,
    args
);
```

For more information, see [Section 108.8, "Named Queries"](#)

## 109.4 Using a SQLCall

The TopLink expression framework enables you to define complex queries at the object level. If your application requires a more complex query or one that accesses data directly, you can specify a custom SQL string in an `SQLCall` object and execute

the SQL string in the context of a `DatabaseQuery` or using `Session API` for executing `Call` objects.

You can provide an `SQLCall` object to any query instead of an expression, but the SQL string contained in the `SQLCall` must return all data required to build an instance of the queried class.

The SQL string can be a complex SQL query that includes input, output, and input/output arguments using JDBC data types.

---



---

**WARNING:** Allowing an unverified SQL string to be passed into methods makes your application vulnerable to SQL injection attacks.

---



---

### 109.4.1 How to Configure a SQLCall Without Arguments

You can configure a `SQLCall` without arguments and execute it directly using `Session API`. Use this approach when you want to execute a SQL string without arguments (or with hard-coded argument values).

#### To configure a SQLCall input without arguments:

1. Instantiate a `SQLCall` object.
2. Pass the SQL string into the constructor as [Example 109–39](#) shows.  
Alternatively, you can use `SQLCall` method `setSQLString`.
3. Execute the `SQLCall` using the appropriate `Session API` as [Example 109–39](#) shows.

You can use any of the following `Session` methods, depending on the type of SQL string you define:

- `executeSelectingCall`: returns a `List` of `Record` objects, each representing a database row.
- `executeNonSelectingCall`: returns `void`.

#### **Example 109–39 Executing a SQLCall Without Arguments**

```
List result = session.executeSelectingCall(
    new SQLCall("SELECT * FROM EMPLOYEE WHERE DEPT_ID = 44")
);
```

### 109.4.2 How to Configure a SQLCall with Arguments Using JDBC Data Types

You can configure a `SQLCall` that takes any combination of input, output, or input/output arguments. Use this approach when you want to bind argument values to the `SQLCall` at runtime, receive output values from the `SQLCall` at execution time, or both.

#### To configure a SQLCall with arguments using JDBC data types:

1. Instantiate a `SQLCall` object.
2. Create the SQL string and designate arguments as input, output, or input/output.

`TopLink` assumes that a token in the custom SQL string of an `SQLCall` is an argument if it is prefixed with one or more number signs ( `#` ), as follows:

- Input parameter prefix: `#` (see [Example 109–40](#)).

- Output parameter prefix: ### (see [Example 109–41](#)).
  - Input/output parameter prefix: ##### (see [Example 109–42](#)).
3. Pass the SQL string into the constructor as [Example 109–40](#), [Example 109–41](#), and [Example 109–42](#) show.

Alternatively, you can use `SQLCall` method `setSQLString`.

4. For each output argument, use the appropriate `SQLCall` method `setCustomSQLArgumentType` to specify the Java data type `TopLink` uses to return the output value, as [Example 109–41](#) shows.

For an input argument, `TopLink` automatically converts the Java data type to the appropriate JDBC data type.

For an input/output argument, the type of the input value determines the type of the output value. As [Example 109–42](#) shows, the data type of the argument value passed into `in_out` is `String` ("MacDonald") so `TopLink` returns the output value (for `EMP_ID`) as a `String`.

5. Instantiate a `DatabaseQuery` appropriate for your SQL string.
6. Configure the `DatabaseQuery` with your `SQLCall` using `DatabaseQuery` method `setCall`, as [Example 109–40](#), [Example 109–41](#), and [Example 109–42](#) show.
7. Specify the names for all input and input/output arguments using `DatabaseQuery` method `addArgument`, as [Example 109–40](#), [Example 109–41](#), and [Example 109–42](#) show.
8. Create a `Vector` of argument values in the same order as you specified argument names in step 7 as [Example 109–40](#), [Example 109–41](#), and [Example 109–42](#) show.
9. Bind values to the arguments and execute the `DatabaseQuery` using `Session` method `executeQuery(DatabaseQuery, java.util.Vector)`, passing in your `DatabaseQuery` and `Vector` of argument values, as [Example 109–40](#), [Example 109–41](#), and [Example 109–42](#) show.

**Example 109–40 Specifying an SQLCall with an Input Argument Using the # Prefix: JDBC Data Types**

```
SQLCall sqlCall = new SQLCall("INSERT INTO EMPLOYEE (L_NAME) VALUES (#last_name)");

DataModifyQuery query = new DataModifyQuery();
query.setCall(sqlCall);
query.addArgument("last_name"); // input

Vector arguments = new Vector();
arguments.add("MacDonald");
session.executeQuery(query, arguments);
```

**Example 109–41 Specifying a SQLCall with an Output Argument Using the ### Prefix: JDBC Data Types**

```
SQLCall sqlCall = new SQLCall(
    "BEGIN INSERT INTO EMPLOYEE (L_NAME) VALUES (#last_name)
    RETURNING EMP_ID INTO ###employee_id; END;");

sqlCall.setCustomSQLArgumentType("employee_id", Integer.class); // specify output value type

ValueReadQuery query = new ValueReadQuery();
query.setCall(sqlCall);
query.addArgument("last_name"); // input
```



```

Vector args = new Vector();
args.add("MacDonald");

Integer employeeID = (Integer) getSession().executeQuery(query, args);

```

**Example 109–42 Specifying a SQLCall with an Input/Output Argument Using the #### Prefix: JDBC Data Types**

```

SQLCall sqlCall = new SQLCall(
    "BEGIN INSERT INTO EMPLOYEE (L_NAME) VALUES (####in_out)
    RETURNING EMP_ID INTO ####in_out; END;");

ValueReadQuery query = new ValueReadQuery();
query.setCall(sqlCall);
query.addArgument("in_out"); // input and output

Vector args = new Vector();
args.add("MacDonald"); // type of input argument determines type of output value

String lastName = (String) getSession().executeQuery(query, args);

```

### 109.4.3 What You May Need to Know About Using a SQLCall

When using SQL calls, you can use a `ReturningPolicy` to control whether or not `TopLink` writes a parameter out or retrieves a value generated by the database.

If you want to invoke a stored procedure or stored function, use a `StoredProcedureCall` or `StoredFunctionCall`.

Alternatively, you can specify a simple SQL string directly on `DatabaseQuery`. You can use this approach to avoid the overhead of creating a `SQLCall` object when your SQL string is simple, uses hard-coded arguments (or no arguments), and you do not require the additional API that `SQLCall` provides.

For more information, see the following:

- [Section 109.2.6, "How to Specify a Custom SQL String in a DatabaseQuery"](#)
- [Section 119.27, "Configuring Returning Policy"](#)
- [Section 109.5, "Using a StoredProcedureCall"](#)
- [Section 109.6, "Using a StoredFunctionCall"](#)

## 109.5 Using a StoredProcedureCall

The `TopLink` expression framework enables you to define complex queries at the object level. If your application requires a more complex query or one that invokes an existing stored procedure that your database provides, you can define a `StoredProcedureCall` object using both JDBC and PL/SQL data types and invoke the stored procedure in the context of a `DatabaseQuery`.

If you are using Oracle Database, you can pass in both JDBC and PL/SQL (non-JDBC) data types.

If you are using a non-Oracle database, you may pass in only JDBC data types.

This section describes the following:

- [How to Configure a StoredProcedureCall Without Arguments](#)
- [How to Configure a StoredProcedureCall with Arguments Using JDBC Data Types](#)
- [How to Configure a PLSQLStoredProcedureCall with PL/SQL Data Type Arguments](#)

- [How to Specify a Simple Optimistic Version Locking Value with a StoredProcedureCall Using JDBC Data Types](#)
- [How to Configure a StoredProcedureCall Output Parameter Event Using JDBC or PL/SQL Data Types](#)
- [What You May Need to Know About Using a StoredProcedureCall](#)

### 109.5.1 How to Configure a StoredProcedureCall Without Arguments

You can configure a `StoredProcedureCall` without arguments and execute it directly using `Session` API. Use this approach when you want to execute a stored procedure that does not take arguments or return values.

#### To configure a `StoredProcedureCall` without arguments using JDBC data types:

1. Instantiate a `StoredProcedureCall` object.
2. Set the name of the stored procedure to execute using `StoredProcedureCall` method `setProcedureName`, as [Example 109-43](#) shows.
3. Execute the `StoredProcedureCall` using the appropriate `Session` API, as [Example 109-43](#) shows.

You can use any of the following `Session` methods, depending on the type of stored procedure you are executing:

- `executeSelectingCall`: returns a `List` of `Record` objects, each representing a database row.
- `executeNonSelectingCall`: returns `void`.

#### **Example 109-43 Executing a SQLCall Without Arguments**

```
StoredProcedureCall spcall = new StoredProcedureCall();
spcall.setProcedureName("Read_All_Employees");
spcall.useNamedCursorOutputAsResultSet("RESULT_SET");

List employees = (List) getSession().executeSelectingCall(spcall);
```

### 109.5.2 How to Configure a StoredProcedureCall with Arguments Using JDBC Data Types

You can configure a `StoredProcedureCall` that takes any combination of input, output, or input/output arguments. Use this approach when you want to bind argument values to the `StoredProcedureCall` at runtime, receive output values from the `StoredProcedureCall` at execution time, or both.

---

---

**Note:** Use this procedure when all input, output, and input/output arguments are JDBC data types. If one or more arguments are PL/SQL (non-JDBC) data types, see [Section 109.5.3, "How to Configure a PLSQLStoredProcedureCall with PL/SQL Data Type Arguments"](#).

---

---

#### To configure a `StoredProcedureCall` with arguments using JDBC data types:

1. Instantiate a `StoredProcedureCall` object.
2. Specify the name of the stored procedure to call using `StoredProcedureCall` method `setProcedureName`, as [Example 109-44](#), [Example 109-45](#), and [Example 109-46](#) show.

3. For each argument, use the appropriate `StoredProcedureCall` methods to specify whether arguments are input, output, or input/output arguments:
  - Input argument: `addNamedArgument` (see [Example 109–44](#)).
  - Output argument: `addNamedOutputArgument` (see [Example 109–45](#)).
  - Input/output argument: `addNamedInOutArgument` (see [Example 109–46](#)).

In general, you should always specify the return Java data type for all output and input/output arguments, as [Example 109–45](#) and [Example 109–46](#) show. If you do not specify a return Java data type, the default is `java.lang.String`.

Typically, you specify arguments using the stored procedure argument name as is. However, you may associate a stored procedure argument name with an alternate name that you use in the `DatabaseQuery`, as [Example 109–44](#) shows. Use this approach to specify a more meaningful argument name if the stored procedure argument name is cryptic.

4. Instantiate a `DatabaseQuery` appropriate for your stored procedure.
5. Configure the `DatabaseQuery` with your `StoredProcedureCall` using `DatabaseQuery` method `setCall`, as [Example 109–44](#), [Example 109–45](#), and [Example 109–46](#) show.
6. Specify the names for all input and input/output arguments using `DatabaseQuery` method `addArgument`, as [Example 109–44](#), [Example 109–45](#), and [Example 109–46](#) show.

If you associated stored procedure argument names with more meaningful alternate names in step 3, use the alternate names in the `DatabaseQuery` method `addArgument`, as [Example 109–44](#) shows.

7. Create a `Vector` of argument values in the same order as you specified argument names in step 6, as [Example 109–44](#), [Example 109–45](#), and [Example 109–46](#) show.
8. Bind values to the arguments and execute the `DatabaseQuery` using `Session` method `executeQuery(DatabaseQuery, java.util.Vector)`, passing in your `DatabaseQuery` and `Vector` of argument values, as [Example 109–44](#), [Example 109–45](#), and [Example 109–46](#) show.

#### **Example 109–44 Specifying a StoredProcedureCall with an Input Argument: JDBC Data Types**

```
// CREATE PROCEDURE INSERT_EMPLOYEE(L_NAME IN VARCHAR) AS
// BEGIN
//     Insert an EMP record initialized with last name.
// END;
```

```
StoredProcedureCall spcall = new StoredProcedureCall();
spcall.setProcedureName("INSERT_EMPLOYEE");
spcall.addNamedArgument("L_NAME", "last_name");
```

```
DataModifyQuery query = new DataModifyQuery();
query.setCall(spcall);
query.addArgument("last_name"); // input
```

```
Vector arguments = new Vector();
arguments.add("MacDonald");
session.executeQuery(query, arguments);
```

**Example 109–45 Specifying a StoredProcedureCall with an Output Argument: JDBC Data Types**

```
// CREATE PROCEDURE GET_EMP_ID(L_NAME IN VARCHAR, EMP_ID OUT INTEGER) AS
// BEGIN
//     Insert an EMP record initialized with last name and return the EMP_ID for this record.
// END;
```

```
StoredProcedureCall spcall = new StoredProcedureCall();
spcall.setProcedureName("GET_EMP_ID");
spcall.addNamedArgument("L_NAME");
spcall.addNamedOutputArgument(
    "EMP_ID",        // procedure parameter name
    "EMP_ID",        // out argument field name
    Integer.class   // Java type corresponding to type returned by procedure
);
```

```
ValueReadQuery query = new ValueReadQuery();
query.setCall(spcall);
query.addArgument("L_NAME"); // input
```

```
Vector args = new Vector();
args.add("MacDonald");
```

```
Integer employeeID = (Integer) getSession().executeQuery(query, args);
```

**Example 109–46 Specifying a StoredProcedureCall with an Input/Output Argument: JDBC Data Types**

```
// CREATE PROCEDURE INSERT_EMPLOYEE(IN_OUT INOUT VARCHAR) AS
// BEGIN
//     Insert an EMP record initialized with last name
//     and return the EMP_CODE_NAME for this record.
// END;
```

```
StoredProcedureCall spcall = new StoredProcedureCall();
spcall.setProcedureName("INSERT_EMP"); // returns EMP_CODE_NAME after insert
spcall.addNamedInOutArgument(
    "IN_OUT",        // procedure parameter name
    "IN_OUT",        // out argument field name
    String.class     // Java type corresponding to type returned by procedure
);
```

```
ValueReadQuery query = new ValueReadQuery();
query.setCall(sqlCall);
query.addArgument("INOUT"); // input and output
```

```
Vector args = new Vector();
args.add("MacDonald"); // type of input argument determines type of output value
```

```
String employeeCode = (String) getSession().executeQuery(query, args);
```

**109.5.3 How to Configure a PLSQLStoredProcedureCall with PL/SQL Data Type Arguments**

You must use the `oracle.toplink.platform.database.oracle.PLSQLStoredProcedureCall` class if any combination of input, output, or input/output arguments are PL/SQL (non-JDBC) data types. Use this approach when you want to bind argument values to the `PLSQLStoredProcedureCall` at run time, receive output values from the `PLSQLStoredProcedureCall` at execution time, or both.

---



---

**Note:** If all arguments are JDBC (not PL/SQL data types), see Section 109.5.2, "How to Configure a StoredProcedureCall with Arguments Using JDBC Data Types".

---



---

**To configure a PLSQLStoredProcedureCall with arguments using JDBC and PL/SQL data types:**

1. Instantiate a `PLSQLStoredProcedureCall` object.
2. Specify the name of the stored procedure to call using `PLSQLStoredProcedureCall` method `setProcedureName`, as [Example 109-47](#), [Example 109-48](#), and [Example 109-49](#) show.
3. For each argument, use the appropriate `PLSQLStoredProcedureCall` methods to specify whether arguments are input, output, or input/output arguments:
  - Input argument: `addNamedArgument` (see [Example 109-47](#)).
  - Output argument: `addNamedOutputArgument` (see [Example 109-48](#)).
  - Input/output argument: `addNamedInOutArgument` (see [Example 109-49](#)).

You must specify the data type for all arguments: input, output, and input/output. You use `oracle.toplink.platform.database.jdbc.JDBCTypes` to specify JDBC types and

`oracle.toplink.platform.database.oracle.OraclePLSQLTypes` to specify PL/SQL (non-JDBC) types.

For JDBC and PL/SQL input arguments (and the in value of input/output arguments), you may use any Java type with sufficient size and precision for the argument.

For JDBC output arguments (and the out value of input/output arguments) TopLink converts the JDBC data types to Java types as before. For PL/SQL output arguments (and the out value of input/output arguments), TopLink converts PL/SQL data types to the Java data types that [Table 109-1](#) lists.

Typically, you specify arguments using the stored procedure argument name as is. However, you may associate a stored procedure argument name with an alternate name that you use in the `DatabaseQuery`, as [Example 109-47](#) shows. Use this approach to specify a more meaningful argument name if the stored procedure argument name is cryptic.

4. Instantiate a `DatabaseQuery` appropriate for your stored procedure.
5. Configure the `DatabaseQuery` with your `PLSQLStoredProcedureCall` using `DatabaseQuery` method `setCall`, as [Example 109-47](#), [Example 109-48](#), and [Example 109-49](#) show.
6. Specify the names for all input and input/output arguments using `DatabaseQuery` method `addArgument`, as [Example 109-47](#), [Example 109-48](#), and [Example 109-49](#) show.

If you associated stored procedure argument names with more meaningful alternate names in step 3, use the alternate names in the `DatabaseQuery` method `addArgument`, as [Example 109-47](#) shows.

7. Create a `Vector` of argument values in the same order as you specified argument names in step 6, as [Example 109-47](#), [Example 109-48](#), and [Example 109-49](#) show.

8. Bind values to the arguments and execute the DatabaseQuery using Session method executeQuery(DatabaseQuery, java.util.Vector), passing in your DatabaseQuery and Vector of argument values, as [Example 109–47](#), [Example 109–48](#), and [Example 109–49](#) show.

**Example 109–47 Specifying a PLSQLStoredProcedureCall with an Input Argument: JDBC and PL/SQL Data Types**

```
// CREATE PROCEDURE INSERT_EMPLOYEE(L_NAME IN VARCHAR, MANAGER IN BOOLEAN) AS
// BEGIN
//     Insert an EMP record initialized with last name and whether or not the employee
//     is a manager.
// END;

PLSQLStoredProcedureCall plsqlcall = new PLSQLStoredProcedureCall();
plsqlcall.setProcedureName("INSERT_EMPLOYEE");
plsqlcall.addNamedArgument("L_NAME", JDBCTypes.VARCHAR_TYPE, 40); // must define length
plsqlcall.addNamedArgument("MANAGER", OraclePLSQLTypes.PLSQLBoolean);

DataModifyQuery query = new DataModifyQuery();
query.setCall(plsqlcall);
query.addArgument("L_NAME"); // input
query.addArgument("MANAGER"); // input

Vector arguments = new Vector();
arguments.add("MacDonald");
arguments.add(Integer.valueOf(1));
session.executeQuery(query, arguments);
```

**Example 109–48 Specifying a PLSQLStoredProcedureCall with an Output Argument: JDBC and PL/SQL Data Types**

```
// CREATE PROCEDURE GET_EMP_ID(L_NAME IN VARCHAR, EMP_ID OUT PLS_INTEGER) AS
// BEGIN
//     Insert an EMP record initialized with last name and return EMP_ID for this row.
// END;

PLSQLStoredProcedureCall plsqlcall = new PLSQLStoredProcedureCall();
plsqlcall.setProcedureName("GET_EMP_ID");
plsqlcall.addNamedArgument("L_NAME", JDBCTypes.VARCHAR_TYPE, 25); // must define length
plsqlcall.addNamedOutputArgument("EMP_ID", OraclePLSQLTypes.PLSQLInteger);

ValueReadQuery query = new ValueReadQuery();
query.setCall(plsqlcall);
query.addArgument("L_NAME"); // input

Vector args = new Vector();
args.add("MacDonald");

Number employeeID = (Number) getSession().executeQuery(query, args);
```

**Example 109–49 Specifying a PLSQLStoredProcedureCall with an Input/Output Argument: JDBC and PL/SQL Data Types**

```
// CREATE PROCEDURE INSERT_EMP(IN_OUT INOUT PLS_INTEGER) AS
// BEGIN
//     Insert an EMP record initialized with department id and return
//     the EMP_ID for this record.
// END;

PLSQLStoredProcedureCall plsqlcall = new PLSQLStoredProcedureCall();
plsqlcall.setProcedureName("INSERT_EMP");
plsqlcall.addNamedInOutArgument("IN_OUT", OraclePLSQLTypes.PLSQLInteger);

ValueReadQuery query = new ValueReadQuery();
```

```

query.setCall(plsqlcall);
query.addArgument("IN_OUT");           // input and output

Vector args = new Vector();
args.add(Integer.valueOf(1234));       // department id

Integer employeeID = new Integer(BigDecimal.intValue(
    getSession().executeQuery(query, args)));

```

## 109.5.4 How to Specify a Simple Optimistic Version Locking Value with a StoredProcedureCall Using JDBC Data Types

When using optimistic version locking, you typically delegate the responsibility for updating the version field to TopLink.

Alternatively, you may choose to use stored procedures to manually update the version field for all of create, read, update, and delete operations.

When using optimistic locking and stored procedure calls, you may only use a simple, sequential numeric value that the stored procedure can generate independently of TopLink. To use a complex value, such as a timestamp, you must delegate the responsibility for updating the version field to TopLink.

For more information, see [Section 16.4.1, "Optimistic Version Locking Policies"](#).

### To specify a simple optimistic version locking value with a StoredProcedureCall using JDBC data types:

1. Create stored procedures for create, read, update, and delete operations.

Each stored procedure is responsible for checking and updating the optimistic lock field: a simple sequential numeric value in your database.

[Example 109–50](#) shows a typical stored procedure for the update operation.

#### **Example 109–50** Stored Procedure for Update Operation Using Simple Optimistic Version Locking

```

PROCEDURE Update_Employee (
    P_EMP_ID NUMBER,
    P_SALARY NUMBER,
    P_END_DATE DATE,
    P_MANAGER_ID NUMBER,
    P_START_DATE DATE,
    P_F_NAME VARCHAR2,
    P_L_NAME VARCHAR2,
    P_GENDER VARCHAR2,
    P_ADDR_ID NUMBER,
    P_VERSION NUMBER,
    P_START_TIME DATE,
    P_END_TIME DATE,
    O_ERROR_CODE OUT NUMBER) AS
BEGIN
Update SALARY set SALARY = P_SALARY WHERE (EMP_ID = P_EMP_ID);
Update EMPLOYEE set END_DATE = P_END_DATE, MANAGER_ID = P_MANAGER_ID, VERSION = P_VERSION +
1, START_DATE = P_START_DATE, F_NAME = P_F_NAME, L_NAME = P_L_NAME, GENDER = P_GENDER, ADDR_
ID = P_ADDR_ID where ((EMP_ID = P_EMP_ID) and (VERSION = P_VERSION));
O_ERROR_CODE := SQL%ROWCOUNT;
END;

```

2. Create a StoredProcedureCall for each of your custom create, read, update, and delete stored procedures.

[Example 109–51](#) shows the StoredProcedureCall for the update stored procedure in [Example 109–50](#).

**Example 109–51 StoredProcedureCall for Update Stored Procedure**

```

UpdateObjectQuery updateQuery = new UpdateObjectQuery();
call = new StoredProcedureCall();
call.setUsesBinding(true);
call.setProcedureName("Update_Employee");
call.addNamedArgument("P_EMP_ID", "EMP_ID");
call.addNamedArgument("P_SALARY", "SALARY");
call.addNamedArgument("P_END_DATE", "END_DATE");
call.addNamedArgument("P_MANAGER_ID", "MANAGER_ID");
call.addNamedArgument("P_START_DATE", "START_DATE");
call.addNamedArgument("P_F_NAME", "F_NAME");
call.addNamedArgument("P_L_NAME", "L_NAME");
call.addNamedArgument("P_GENDER", "GENDER");
call.addNamedArgument("P_ADDR_ID", "ADDR_ID");
call.addNamedArgument("P_VERSION", "VERSION");
call.addNamedArgument("P_START_TIME", "START_TIME");
call.addNamedArgument("P_END_TIME", "END_TIME");
call.addNamedOutputArgument("O_ERROR_CODE", "O_ERROR_CODE", Long.class);
updateQuery.setCall(call);

```

For more information, see the following:

- [Section 109.5.1, "How to Configure a StoredProcedureCall Without Arguments"](#)
  - [Section 109.5.2, "How to Configure a StoredProcedureCall with Arguments Using JDBC Data Types"](#)
  - [Section 109.5.3, "How to Configure a PLSQLStoredProcedureCall with PL/SQL Data Type Arguments"](#)
3. Configure the TopLink descriptor query manager to use your StoredProcedureCall objects for create, read, update, and delete operations.

[Example 109–52](#) shows how to use a descriptor customizer class to update the TopLink descriptor query manager with the update StoredProcedureCall from [Example 109–51](#).

**Example 109–52 Configuring the TopLink Descriptor Query Manager with a StoredProcedureCall**

```

import oracle.toplink.tools.sessionconfiguration.DescriptorCustomizer;
import oracle.toplink.descriptors.ClassDescriptor;

public class EmployeeDescriptorCustomizer implements DescriptorCustomizer {

    public void customize(ClassDescriptor descriptor) {
        descriptor.getQueryManager().setUpdateQuery(updateQuery);
    }
}

```

For more information, see the following:

- [Section 23.4, "Configuring Custom SQL Queries for Basic Persistence Operations"](#)
  - [Section 119.34, "Configuring a Descriptor Customizer Class"](#)
4. Define a StoredProcedureCall output parameter event to handle any errors.



---



---

**Note:** In the Oracle database, the rowcount is not maintained when calling a stored procedure. You must ensure that the rowcount is returned using an output parameter. Use the `Session` event `outputParametersDetected` to check the rowcount and raise an error. Alternatively, the stored procedure could check the rowcount and throw an exception.

---



---

For more information, see [Section 109.5.5, "How to Configure a StoredProcedureCall Output Parameter Event Using JDBC or PL/SQL Data Types"](#)

## 109.5.5 How to Configure a StoredProcedureCall Output Parameter Event Using JDBC or PL/SQL Data Types

TopLink manages output parameter events for databases that support them. For example, if a stored procedure returns an error code that indicates that the application wants to check for an error condition, TopLink raises the session event `outputParametersDetected` to allow the application to process the output parameters.

### To configure a StoredProcedureCall output parameter event using JDBC or PL/SQL data types:

1. Create a `StoredProcedureCall` using JDBC arguments, PL/SQL arguments, or both. [Example 109–53](#) shows a `StoredProcedureCall` using JDBC arguments.

For more information, see the following:

- [Section 109.5.2, "How to Configure a StoredProcedureCall with Arguments Using JDBC Data Types"](#)
- [Section 109.5.3, "How to Configure a PLSQLStoredProcedureCall with PL/SQL Data Type Arguments"](#)

#### **Example 109–53** *Stored Procedure*

```
PROCEDURE Update_Employee (
    P_EMP_ID NUMBER,
    P_SALARY NUMBER,
    P_END_DATE DATE,
    P_MANAGER_ID NUMBER,
    P_START_DATE DATE,
    P_F_NAME VARCHAR2,
    P_L_NAME VARCHAR2,
    P_GENDER VARCHAR2,
    P_ADDR_ID NUMBER,
    P_VERSION NUMBER,
    P_START_TIME DATE,
    P_END_TIME DATE,
    O_ERROR_CODE OUT NUMBER) AS
BEGIN
Update SALARY set SALARY = P_SALARY WHERE (EMP_ID = P_EMP_ID);
Update EMPLOYEE set END_DATE = P_END_DATE, MANAGER_ID = P_MANAGER_ID, VERSION = P_VERSION +
1, START_DATE = P_START_DATE, F_NAME = P_F_NAME, L_NAME = P_L_NAME, GENDER = P_GENDER, ADDR_
ID = P_ADDR_ID where ((EMP_ID = P_EMP_ID) and (VERSION = P_VERSION));
O_ERROR_CODE := SQL%ROWCOUNT;
END;
```

2. Create a `SessionEventListener` that handles the `outputParametersDetected` event, as [Example 109–54](#) shows.

Subclassing the `oracle.toplink.sessions.SessionEventAdapter` is an easy way to create a `SessionEventListener`: you only need to override the specific `SessionEventListener` methods you are interested in.

In [Example 109–54](#), `SessionEvent` method `getProperty` uses an argument value of `ERROR_CODE`. This property name and its data type is defined in the `StoredProcedureCall` method `addNamedOutputArgument`.

**Example 109–54 *SessionEventListener for outputParametersDetected Event***

```
import oracle.toplink.sessions.SessionEventAdapter;
import oracle.toplink.sessions.SessionEvent;

public class OptimisticLockListener extends SessionEventAdapter {
    public OptimisticLockListener() {
    }

    public void outputParametersDetected(SessionEvent event) {
        DatabaseQuery query = event.getQuery();
        if ((query != null) && query.isObjectLevelModifyQuery()) {
            Number rowcount = new Integer(1);
            if (event.getResult() instanceof Map) {
                rowcount = (Number)((Map)event.getResult()).get("O_ERROR_CODE");
            }
            if (rowcount.longValue() <= 0) {
                if (query.isDeleteObjectQuery()) {
                    DeleteObjectQuery deleteQuery = (DeleteObjectQuery)query;
                    throw OptimisticLockException.objectChangedSinceLastReadWhenDeleting(
                        deleteQuery.getObject(), deleteQuery);
                }
                else if (query.isWriteObjectQuery()) {
                    WriteObjectQuery updateQuery = (WriteObjectQuery)query;
                    throw OptimisticLockException.objectChangedSinceLastReadWhenUpdating(
                        updateQuery.getObject(), updateQuery);
                }
            }
        }
    }
}
```

3. Add your `SessionEventListener` instance to the session event manager as [Example 109–55](#) shows.

You must do this step before executing your stored procedure.

For more information, see [Section 87.2.5, "Managing Session Events with the Session Event Manager"](#)

**Example 109–55 *Adding SessionEventListener to the Session Event Manager***

```
getSession().getEventManager().addListener(new OptimisticLockListener());
```

4. Execute the query.

If there is an error and a `SessionEvent` of type `outputParametersDetected` is raised, `TopLink` will notify your `SessionEventListener`.

## 109.5.6 What You May Need to Know About Using a StoredProcedureCall

`TopLink` automatically converts PL/SQL data types into the Java data types that [Table 109–1](#) lists for out arguments (and the out value of input/output arguments).

**Table 109–1 TopLink PL/SQL to Java Data Type Conversion: Out Arguments and Out Value of Input/Output Arguments**

PL/SQL Data Type	OraclePLSQLTypes Enum	Java Type
BINARY_INTEGER	BinaryInteger	java.math.BigDecimal
BOOLEAN	PLSQLBoolean	java.lang.Integer
DEC	Dec	java.math.BigDecimal
INT	Int	java.math.BigDecimal
NATURAL	Natural	java.math.BigDecimal
NATURALN	NaturalN	java.math.BigDecimal
PLS_INTEGER	PLSQLInteger	java.math.BigDecimal
POSITIVE	Positive	java.math.BigDecimal
POSITIVEN	PositiveN	java.math.BigDecimal
SIGNTYPE	SignType	java.lang.Integer

You may use the value from any Java type for a PL/SQL in argument (or in value of an input/output argument) as long as the size and precision of the Java type is appropriate for the PL/SQL type.

---

**Note:** You no longer need to use DatabaseQuery method `bindAllParameters` when using a `StoredProcedureCall` with OUT or INOUT parameters. However, you should always specify the Java type for all OUT and INOUT parameters. If you do not, be aware of the fact that they default to type `String`.

---

## 109.6 Using a StoredFunctionCall

The TopLink expression framework enables you to define complex queries at the object level. If your application requires a more complex query or one that invokes an existing stored function that your database provides, you can define a `StoredFunctionCall` object using both JDBC and PL/SQL data types and invoke the stored function in the context of a `DatabaseQuery`.

Note that not all databases provide stored functions.

In [Example 109–56](#), note that the name of the stored function is set using `StoredFunctionCall` method `setProcedureName`.

### Example 109–56 Creating a StoredFunctionCall

```
StoredFunctionCall functionCall = new StoredFunctionCall();
functionCall.setProcedureName("CHECK_VALID_EMPLOYEE");
functionCall.addNamedArgument("EMP_ID");
functionCall.setResult("FUNCTION_RESULT", String.class);
ValueReadQuery query = new ValueReadQuery();
query.setCall(functionCall);
query.addArgument("EMP_ID");
Vector args = new Vector();
args.addElement(new Integer(44));
String valid = (String) session.executeQuery(query, args);
```

## 109.6.1 What You May Need to Know About Using a StoredFunctionCall

In general, both stored procedures and stored functions let you specify input parameters, output parameters, and input and output parameters. For more information, see [Section 109.5, "Using a StoredProcedureCall"](#). However, stored procedures need not return values, while stored functions always return a single value.

The `StoredFunctionCall` class extends `StoredProcedureCall` to add one new method: `setResult`. Use this method to specify the name (and alternatively both the name and type) under which `TopLink` stores the return value of the stored function.

When `TopLink` prepares a `StoredFunctionCall`, it validates its SQL and throws a `ValidationException` under the following circumstances:

- If your current platform does not support stored functions. Stored functions are supported only for Oracle.
- If you fail to specify the return type.

## 109.7 Using Java Persistence Query Language (JPQL) Calls

The `TopLink` expression framework lets you define complex queries at the object level.

Alternatively, you can specify a custom JPQL string in an JPQL call object and provide that object to any query. See "Using Java Persistence Query Language (JPQL) Calls" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_Basic\\_Query\\_API\\_%28ELUG%29#Using\\_Java\\_Persistence\\_Query\\_Language\\_.28JPQL.29\\_Calls](http://wiki.eclipse.org/Using_Basic_Query_API_%28ELUG%29#Using_Java_Persistence_Query_Language_.28JPQL.29_Calls) for information.

## 109.8 Using EIS Interactions

For an EIS root descriptor, you can define EIS interactions to invoke methods on an EIS.

`TopLink` represents EIS interactions using instances of `oracle.toplink.eis.interactions.EISInteraction`. These classes implement the `Call` interface and can be used wherever a `Call` can be used.

[Table 109–2](#) lists the type of EIS interactions that `TopLink` supports.

**Table 109–2 EIS Interactions**

EIS Interaction Type	Description
<code>IndexedInteraction</code>	Defines the specification for a call to a JCA interaction that uses indexed records. Builds the input and output records from the arguments by position.
<code>MappedInteraction</code>	Defines the specification for a call to a JCA interaction that uses mapped records. Builds the input and output records from the arguments by name.
<code>XMLInteraction</code>	Specifies an instance of <code>MappedInteraction</code> that defines the specification for a call to a JCA interaction that uses XML records defined by the XML schema document (XSD) associated with the EIS project (for more information, see <a href="#">Section 5.6.3, "How to Import an XML Schema"</a> ).
<code>QueryStringInteraction</code>	Specifies an instance of <code>MappedInteraction</code> that defines the specification for a call to a JCA interaction that uses a query string. Prefix arguments in the query string with a number sign ( # ) character.

**Table 109–2 (Cont.) EIS Interactions**

EIS Interaction Type	Description
XQueryInteraction	Specifies an instance of <code>XMLInteraction</code> that defines the specification for a call to a JCA interaction that uses XQuery. Translates the XQuery from the query arguments.

You can use `TopLink` to define an interaction for each basic persistence operation (`insert`, `update`, `delete`, `read object`, `read all`, or `does exist`) so that when you query and modify your EIS-mapped objects, the `TopLink` runtime will use the appropriate EIS interaction. For more information, see [Section 76.5, "Configuring Custom EIS Interactions for Basic Persistence Operations"](#).

You can also use `TopLink` to define an interaction as a named query for `read object` and `read-all object` queries. These queries are not called for basic persistence operations; you can call these additional queries by name in your application for special purposes. For more information, see [Section 119.7.1.8, "Creating an EIS Interaction for a Named Query"](#).

## 109.9 Handling Exceptions

Most exceptions in queries are database exceptions, resulting from a failure in the database operation. Write operations can also throw an `OptimisticLockException` on a write, update, or delete operation in applications that use optimistic locking. To catch these exceptions, execute all database operations within a `try-catch` block:

```
try {
    List employees = session.readAllObjects(Employee.class);
}
catch (DatabaseException exception) {
    // handle exception
}
```

See [Appendix A, "Troubleshooting a TopLink Application"](#) for more information about exceptions in a `TopLink` application.

## 109.10 Handling Collection Query Results

`TopLink` provides a `useCollectionClass` method to all subclasses of `DataReadQuery` and `ReadAllQuery`. Use this method to configure a query to return results as any concrete instance of `Collection` or `Map`. You can set various collection class types for queries, such as `ArrayList`, `HashSet`, `HashMap`, and `TreeSet`.

## 109.11 Handling Report Query Results

[Table 109–3](#) lists the `ReportQuery` methods you can use to configure how a `ReportQuery` returns its results.

By default, the `ReportQuery` returns a `Collection` of `ReportQueryResult` objects.

**Table 109–3 Report Query Result Options**

Method	Query Returns	Description
<code>setShouldReturnSingleAttribute</code>	<code>Collection</code>	Returns a single attribute (not wrapped in a <code>ReportQueryResult</code> ). Use this option if you know that the <code>ReportQuery</code> returns only one attribute.

**Table 109–3 (Cont.) Report Query Result Options**

Method	Query Returns	Description
<code>setShouldReturnSingleResult</code>	<code>ReportQueryResult</code>	Returns only the first <code>ReportQueryResult</code> object (not wrapped in a <code>Collection</code> or <code>Map</code> ). Use this option if you know that the <code>ReportQuery</code> returns only one row.
<code>setShouldReturnSingleValue</code>	<code>Object</code>	Returns only a single value. Use this option if you know that the <code>ReportQuery</code> returns only one row that contains only one attribute.

For more information, see the following:

- [Section 108.7.5, "Report Query"](#)
- [Section 108.5.2, "Report Query Results"](#)
- [Section 109.2.1.3, "Reading Objects Using Report Queries"](#)
- [Section 119.7.1.6, "Configuring Named Query Attributes"](#)

---

## Introduction to TopLink Expressions

Using the TopLink expressions framework, you can specify query search criteria based on your domain object model.

This chapter includes the following sections:

- [Expression Framework](#)
- [Expression Components](#)
- [Parameterized Expressions](#)
- [Query Keys and Expressions](#)
- [Multiple Expressions](#)
- [Data Queries and Expressions](#)
- [Creating an Expression](#)
- [Creating and Using a User-Defined Function](#)

### 110.1 Expression Framework

The TopLink expression framework provides methods through the following classes:

- The `Expression` class provides most general functions, such as `toUpperCase`.
- The `ExpressionMath` class supplies mathematical methods.

[Example 110–1](#) illustrates how to use the `Expression` class.

**Example 110–1 Using the Expression Class**

```
expressionBuilder.get("lastName").equal("Smith");
```

[Example 110–2](#) illustrates how to use the `ExpressionMath` class.

**Example 110–2 Using the ExpressionMath Class**

```
ExpressionMath.abs(ExpressionMath.subtract(emp.get("salary"),  
emp.get("spouse").get("salary")).greaterThan(10000)
```

This division of functionality enables TopLink expressions to provide similar mathematical functionality to the `java.lang.Math` class, but keeps both the `Expression` and `ExpressionMath` classes from becoming unnecessarily complex.

#### 110.1.1 Expressions Compared to SQL

Expressions offer the following advantages over SQL when you access a database:

- Expressions are easier to maintain because the database is abstracted.

- Changes to descriptors or database tables do not affect the querying structures in the application.
- Expressions enhance readability by standardizing the `Query` interface so that it looks similar to traditional Java calling conventions. For example, the Java code required to get the street name from the `Address` object of the `Employee` class looks like this:

```
emp.getAddress().getStreet().equals("Meadowlands");
```

The expression to get the same information is similar:

```
emp.get("address").get("street").equal("Meadowlands");
```

- Expressions allow read queries to transparently query between two classes that share a relationship. If these classes are stored in multiple tables in the database, `TopLink` automatically generates the appropriate join statements to return information from both tables.
- Expressions simplify complex operations. For example, the following Java code retrieves all employees that live on "Meadowlands" whose salary is greater than 10,000:

```
ExpressionBuilder emp = new ExpressionBuilder();
Expression exp = emp.get("address").get("street").equal("Meadowlands");
Vector employees = session.readAllObjects(Employee.class,
    exp.and(emp.get("salary").greaterThan(10000)));
```

`TopLink` automatically generates the appropriate SQL from that code:

```
SELECT t0.VERSION, t0.ADDR_ID, t0.F_NAME, t0.EMP_ID, t0.L_NAME, t0.MANAGER_ID,
t0.END_DATE, t0.START_DATE, t0.GENDER, t0.START_TIME, t0.END_TIME, t0.SALARY
FROM EMPLOYEE t0, ADDRESS t1 WHERE ((t1.STREET = 'Meadowlands') AND (t0.SALARY
> 10000)) AND (t1.ADDRESS_ID = t0.ADDR_ID)
```

## 110.2 Expression Components

A simple expression usually consists of the following three parts:

1. The *attribute*, which represents a mapped attribute or query key of the persistent class
2. The *operator*, which is an expression method that implements boolean logic, such as `GreaterThan`, `Equal`, or `Like`
3. The *constant* or *comparison*, which refers to the value used to select the object

In the following code fragment:

```
expressionBuilder.get("lastName").equal("Smith");
```

- The attribute is `lastName`.
- The operator is `equal`.
- The constant is the string "Smith".

The `expressionBuilder` substitutes for the object or objects to be read from the database. In this example, `expressionBuilder` represents `employees`.

You can use the following components when constructing an `Expression`:

- [Boolean Logic](#)
- [Database Functions and Operators](#)
- [Mathematical Functions](#)
- [XMLType Functions](#)



- Platform and User-Defined Functions
- Expressions for One-to-One and Aggregate Object Relationships
- Expressions for Joining and Complex Relationships

## 110.2.1 Boolean Logic

Expressions use standard boolean operators, such as AND, OR, and NOT, and you can combine multiple expressions to form more complex expressions. [Example 110-3](#) illustrates a code fragment that queries for projects managed by a selected person, and that have a budget greater than or equal to 1,000,000.

### **Example 110-3 Using Boolean Logic in an Expression**

```
ExpressionBuilder project = new ExpressionBuilder();
Expression hasRightLeader, bigBudget, complex;
Employee selectedEmp = someWindow.getSelectedEmployee();
hasRightLeader = project.get("teamLeader").equal(selectedEmp);
bigBudget = project.get("budget").greaterThanEqual(1000000);
complex = hasRightLeader.and(bigBudget);
Vector projects = session.readAllObjects(Project.class, complex);
```

## 110.2.2 Database Functions and Operators

### Functions

TopLink expressions support a variety of database functions, including, but not limited to, the following:

- toUpperCase
- toLowerCase
- toDate
- decode
- locate
- monthsBetween
- nextDay
- replace
- reverse
- substring
- translate

---

**Note:** Some functions may be database platform specific.

---

Database functions let you define more flexible queries. You can use these functions in either a report query items using a SELECT clause, or with comparisons in a query's selection criteria using a WHERE clause. [Example 110-4](#) illustrates a code fragment that matches several last names, including "SMART", "Smith", and "Smothers":

### **Example 110-4 Using a Database Function Supported by the Expression API**

```
emp.get("lastName").toUpperCase().like("SM%")
```

You access most functions using `Expression` methods such as `toUpperCase`.

Some functions have very specific purpose: you can use `ascending` and `descending` functions only within an ordering expression to place the result in ascending or descending order, as [Example 110-5](#) shows:

**Example 110-5 Using an Ordering Database Function**

```
readAllQuery.addOrderBy(expBuilder.get("address").get("city").ascending())
```

---

**Note:** Ordering is not supported for in-memory queries (see [Section 108.16.2, "How to Use In-Memory Queries"](#)).

---

You can use aggregate functions, such as `average`, `minimum`, `maximum`, `sum` and so forth, with the `ReportQuery` (see [Section 108.7.5, "Report Query"](#)).

### Operators

Operators are relation operations that compare two values. `TopLink` expressions support the following operators:

- `like`
- `notLike`
- `equal`
- `notEqual`
- `lessThan`
- `lessThanEqual`
- `equalsIgnoreCase`
- `greaterThan`
- `greaterThanEqual`
- `in`
- `notIn`
- `between`
- `notBetween`

[Example 110-4](#) demonstrates the use of the `like` operator.

## 110.2.3 Mathematical Functions

Mathematical functions are available through the `ExpressionMath` class. Mathematical function support in expressions is similar to the support provided by the Java class `java.lang.Math`. [Example 110-6](#) illustrates using the `abs` and `subtract` methods.

**Example 110-6 Using Mathematical Functions in an Expression**

```
ExpressionMath.abs(ExpressionMath.subtract(emp.get("salary"), emp.get("spouse")  
.get("salary"))).greaterThan(10000)
```

## 110.2.4 XMLType Functions

You can use the following operators when constructing queries against data mapped to Oracle Database XMLType column:

- `extract`: Takes an XPath string and returns an XMLType which corresponds to the part of the original document that matches the XPath.
- `extractValue`: Takes an XPath string and returns either a numerical or string value based on the contents of the node pointed to by the XPath.
- `existsNode`: Takes an XPath expression and returns the number of nodes that match the Xpath.
- `getStringVal`: Gets the string representation of an XMLType object.
- `getNumberVal`: Gets the numerical representation of an XMLType object.
- `isFragment`: Evaluates to 0 if the XML is a well formed document. Evaluates to 1 if the document is a fragment.

[Example 110–7](#) illustrates how to use the `extract` operator in a query:

### **Example 110–7 Using the XMLType Extract Operator**

```
Expression criteria =
builder.get("resume").extract("//education/degree/text()").getStringVal().equal("BCS");
Vector employees = session.readAllObject(Employee.class, criteria);
```

## 110.2.5 Platform and User-Defined Functions

You can use the Expression method `getFunction` to access database functions that TopLink does not support directly. [Example 110–8](#) illustrates how to access a database function named `VacationCredit` from within an expression, even though there is no support for such a function in the Expression API.

### **Example 110–8 Using a Database Function Not Supported by the Expression API**

```
emp.get("lastName").getFunction("VacationCredit").greaterThan(42)
```

This expression produces the following SQL:

```
SELECT . . . WHERE VacationCredit(EMP.LASTNAME) > 42
```

The Expression API includes additional forms of the `getFunction` method that allow you to specify arguments. For more information, see *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.

You can also access a custom function that you create. For more information on creating a custom function in TopLink, see [Section 110.8, "Creating and Using a User-Defined Function"](#).

## 110.2.6 Expressions for One-to-One and Aggregate Object Relationships

Expressions can include an attribute that has a one-to-one relationship with another persistent class. A one-to-one relationship translates naturally into a SQL join that returns a single row.

[Example 110–9](#) illustrates a code fragment that accesses fields from an employee's address.

### **Example 110–9 Using an Expression with a One-to-One Relationship**

```
emp.get("address").get("country").like("%S%")
```

[Example 110–9](#) corresponds to joining the EMPLOYEE table to the ADDRESS table, based on the address foreign key, and checking for the country name.

You can nest these relationships infinitely, so it is possible to ask for complex information as follows:

```
project.get("teamLeader").get("manager").get("manager").get("address").get("street")
```

## 110.2.7 Expressions for Joining and Complex Relationships

You can query against complex relationships, such as one-to-many, many-to-many, direct collection, and aggregate collection relationships. Expressions for these types of relationships are more complex to build, because the relationships do not map directly to joins that yield a single row per object.

This section describes the following:

- [What You May Need to Know About Joins](#)
- [Using TopLink Expression API for Joins](#)

### 110.2.7.1 What You May Need to Know About Joins

A **join** is a relational database query that combines rows from two or more tables. Relational databases perform a join whenever multiple tables appear in the query's FROM clause. The query's select list can select any columns from any of these tables.

An inner join (sometimes called a "simple join") is a join of two or more tables that returns only those rows that satisfy the join condition.

An outer join extends the result of an inner join. An outer join returns all rows that satisfy the join condition and also returns some or all of those rows from one table for which no rows from the other satisfy the join condition. Outer joins can be categorized as left or right:

- A query that performs a left outer join of tables A and B returns all rows from A. For all rows in A that have no matching rows in B, the database returns null for any select list expressions containing columns of B.
- A query that performs a right outer join of tables A and B returns all rows from B. For all rows in B that have no matching rows in A, the database returns null for any select list expressions containing columns of A.

When you query with a join expression, TopLink can use joins to check values from other objects or other tables that represent parts of the same object. Although this works well under most circumstances, it can cause problems when you query against a one-to-one relationship, in which one side of the relationship is not present.

For example, Employee objects may have an Address object, but if the Address is unknown, it is null at the object level and has a null foreign key at the database level. When you attempt a read that traverses the relationship, missing objects cause the query to return unexpected results. Consider the following expression:

```
(emp.get("firstName").equal("Steve")).or(emp.get("address").get("city").equal("Ottawa"))
```

In this case, employees with no address do not appear in the result set, regardless of their first name. Although not obvious at the object level, this behavior is fundamental to the nature of relational databases.

Outer joins rectify this problem in the databases that support them. In this example, the use of an outer join provides the expected result: all employees named Steve appear in the result set, even if their address is unknown.

To implement an outer join, use Expression method `getAllowingNull`, rather than `get`, and Expression method `anyOfAllowingNone`, rather than `anyOf`.

For example:

```
(emp.get("firstName").equal("Steve")).or(
emp.getAllowingNull("address").get("city").equal("Ottawa"))
```

Support and syntax for outer joins vary widely between databases and database drivers. TopLink supports outer joins for most databases.

### 110.2.7.2 Using TopLink Expression API for Joins

You can use joins anywhere expressions are used, including: selection-criteria, ordering (see [Section 109.2.1.5, "Specifying Read Ordering"](#)), report queries (see [Section 108.7.5, "Report Query"](#)), partial objects (see [Section 108.7.1.3, "Partial Object Queries"](#)), one-to-one relational mappings (see [Section 28.8, "Configuring Joining at the Mapping Level"](#)), and join reading (see [Section 108.7.1.5, "Join Reading and Object-Level Read Queries"](#)).

Use the expression API shown in [Table 110–1](#) to configure inner and outer join expressions.

**Table 110–1 Expression API for Joins**

Expression API	Type of Join	Type of Mapping
<code>get</code>	inner	one-to-one
<code>getAllowingNull</code>	outer	one-to-one
<code>anyOf</code>	inner	one-to-many, many-to-many
<code>anyOfAllowingNone</code>	outer	one-to-many, many-to-many

To query across a one-to-many or many-to-many relationship, use the `anyOf` operation. As its name suggests, this operation supports queries that return all items on the "many" side of the relationship that satisfy the query criteria.

[Example 110–10](#) illustrates an expression that returns employees who manage at least one employee (through a one-to-many relationship) with a salary less than \$10,000.

#### **Example 110–10 Using an Expression with a One-to-Many Relationship**

```
emp.anyOf("managedEmployees").get("salary").lessThan(10000);
```

[Example 110–11](#) illustrates how to query across a many-to-many relationship using a similar strategy:

#### **Example 110–11 Using an Expression with a Many-to-Many Relationship**

```
emp.anyOf("projects").equal(someProject)
```

TopLink translates these queries into SQL that joins the relevant tables using a `DISTINCT` clause to remove duplicates. TopLink translates [Example 110–10](#) into the following SQL:

```
SELECT DISTINCT . . . FROM EMP t1, EMP t2 WHERE
t2.MANAGER_ID = t1.EMP_ID AND t2.SALARY < 10000
```

You can use one-to-one and one-to-many join expressions in an `ObjectLevelReadyQuery` to configure joins on a per-query basis (see [Section 108.7.1.5, "Join Reading and Object-Level Read Queries"](#)).

You can also configure joins at the mapping level (see [Section 28.8, "Configuring Joining at the Mapping Level"](#)).

---

**Note:** Calling `anyOf` once would result in a different outcome than if you call it twice. For example, if you query for an employee with a telephone area code of 613 and a number of 123-4599, you would use a single `anyOf` and a temporary variable. If you query for an employee, who has a telephone with an area code of 613, and who has a telephone with a number of 123-4599, you would call `anyOf` twice.

---

## 110.3 Parameterized Expressions

A relationship mapping differs from a regular query because it retrieves data for many different objects. To be able to specify these queries, supply arguments when you execute the query. Use the `getParameter` and `getField` methods to acquire values for the arguments.

A parameterized expression executes searches and comparisons based on variables instead of constants. This approach lets you build expressions that retrieve context-sensitive information. This technique is useful when you define EJB finders (see [Section 108.15](#), "EJB 2.n CMP Finders").

Parameterized expressions require that the relationship mapping know how to retrieve an object or collection of objects based on its current context. For example, a one-to-one mapping from `Employee` to `Address` must query the database for an address based on foreign key information from the `Employee` table. Each mapping contains a query that `TopLink` constructs automatically based on the information provided in the mapping. To specify expressions yourself, use the mapping customization mechanisms.

You can use parameterized expressions to create reusable queries (see [Section 108.8](#), "Named Queries").

### 110.3.1 Expression Method `getParameter`

The `Expression` method `getParameter` returns an expression that becomes a parameter in the query. This lets you create a query that includes user input in the search criteria. The parameter must be either the fully qualified name of the field from a descriptor's row, or a generic name for the argument.

Parameters you construct this way are global to the current query, so you can send this message to any expression object.

[Example 110–12](#) illustrates how to use a custom query to find an employee by first name.

#### **Example 110–12 Using a Parameterized Expression in a Custom Query**

```
Expression firstNameExpression;

ReadObjectQuery query = new ReadObjectQuery(Employee.class);
ExpressionBuilder emp = query.getExpressionBuilder();
firstNameExpression = emp.get("firstName").equal(emp.getParameter("firstName"));
query.setSelectionCriteria(firstNameExpression);
query.addArgument("firstName");
Vector v = new Vector();
v.addElement("Sarah");
Employee e = (Employee) session.executeQuery(query, v);
```

[Example 110–13](#) illustrates how to use a custom query to find all employees that live in the same city as a given employee.

**Example 110–13 Using Nested Parameterized Expressions**

```

Expression addressExpression;
ReadObjectQuery query = new ReadObjectQuery(Employee.class);
ExpressionBuilder emp = query.getExpressionBuilder();
addressExpression =
    emp.get("address").get("city").equal(
        emp.getParameter("employee").get("address").get("city"));
query.setName("findByCity");
query.setSelectionCriteria(addressExpression);
query.addArgument("employee");
Vector v = new Vector();
v.addElement(employee);
Employee e = (Employee) session.executeQuery(query, v);

```

[Example 110–14](#) illustrates how to obtain a simple one-to-many mapping from class `PolicyHolder` to `Policy` using a nondefault selection criteria. The `SSN` field of the `POLICY` table is a foreign key to the `SSN` field of the `HOLDER` table.

**Example 110–14 Using a Parameterized Expression in a Mapping**

```

OneToManyMapping mapping = new OneToManyMapping();
mapping.setAttributeName("policies");
mapping.setGetMethodName("getPolicies");
mapping.setSetMethodName("setPolicies");
mapping.setReferenceClass(Policy.class);

// Build a custom expression here rather than using the defaults
ExpressionBuilder policy = new ExpressionBuilder();
mapping.setSelectionCriteria(policy.getField("POLICY.SSN").equal(policy.
    getParameter("HOLDER.SSN")));

```

### 110.3.2 Expression Method `getField`

The `Expression` method `getField` returns an expression that represents a database field with the given name. Use this method to construct the selection criteria for a mapping. The argument is the fully qualified name of the required field. Because fields are not global to the current query, you must send this method to an expression that represents the table from which this field is derived. See also [Section 110.6, "Data Queries and Expressions"](#).

[Example 110–15](#) illustrates how to use the `Expression` method `getField`.

**Example 110–15 Using Expression Method `getParameter`**

```

ExpressionBuilder address = new ExpressionBuilder();
Expression exp = address.getField("ADDRESS.EMP_ID").equal(
    address.getParameter("EMPLOYEE.EMP_ID")
);
exp = exp.and(address.getField("ADDRESS.TYPE").equal(null));

```

## 110.4 Query Keys and Expressions

A query key is a schema-independent alias for a database field name.

Query keys are supported in relational database projects only.

Query keys are generated automatically for all direct and relationship mappings. The name of the query key is the class attribute name.

For more information on how query keys are created and modified, see [Section 119.10, "Configuring Query Keys"](#).

[Example 110–16](#) illustrates how to use the query key `firstName` for the corresponding directly mapped `Employee` attribute.

**Example 110–16 Using an Automatically Generated Query Key in an Expression**

```
Vector employees = session.readAllObjects(Employee.class,  
    new ExpressionBuilder().get("firstName").equal("Bob"));
```

[Example 110–17](#) illustrates how to use a one-to-one query key within the TopLink expression framework.

**Example 110–17 Using a One-to-One Query Key in an Expression**

```
ExpressionBuilder employee = new ExpressionBuilder();  
Vector employees = session.readAllObjects(Employee.class,  
    employee.get("address").get("city").equal("Ottawa"));
```

To access one-to-many and many-to-many query keys that define a distinct join across a collection relationship, use `Expression` method `anyOf`.

## 110.5 Multiple Expressions

Expressions support subqueries (SQL subselects) and parallel selects. To create a subquery, use a single expression builder. With parallel selects, use multiple expression builders when you define a single query. This lets you specify joins for unrelated objects at the object level.

### 110.5.1 How to Use Subselects and Subqueries

Some queries compare the results of other, contained queries (or subqueries). SQL supports this comparison through subselects. TopLink expressions provide subqueries to support subselects.

Subqueries lets you define complex expressions that query on aggregated values (counts, min, max) and unrelated objects (`exists`, `in`, comparisons). To obtain a subquery, pass an instance of a report query to any expression comparison operation, or use the `subQuery` operation on an expression builder. The subquery is not required to have the same reference class as the parent query, and it must use its own expression builder.

You can nest subqueries, or use them in parallel. Subqueries can also make use of custom SQL.

For expression comparison operations that accept a single value (`equal`, `greaterThan`, `lessThan`), the subquery result must return a single value. For expression comparison operations that accept a set of values (`in`, `exists`), the subquery result must return a set of values.

[Example 110–18](#) illustrates how to create an expression that matches all employees with more than five managed employees.

**Example 110–18 A Subquery Expression Using a Comparison and Count Operation**

```
ExpressionBuilder emp = new ExpressionBuilder();  
ExpressionBuilder managedEmp = new ExpressionBuilder();  
ReportQuery subQuery = new ReportQuery(Employee.class, managedEmp);  
subQuery.addCount();  
subQuery.setSelectionCriteria(managedEmp.get("manager").equal(emp));  
Expression exp = emp.subQuery(subQuery).greaterThan(5);
```

[Example 110–19](#) illustrates how to create an expression that matches the employee with the highest salary in the city of Ottawa.



**Example 110–19 A Subquery Expression Using a Comparison and Max Operation**

```

ExpressionBuilder emp = new ExpressionBuilder();
ExpressionBuilder ottawaEmp = new ExpressionBuilder();
ReportQuery subQuery = new ReportQuery(Employee.class, ottawaEmp);
subQuery.addMax("salary");
subQuery.setSelectionCriteria(ottawaEmp.get("address").get("city").equal("Ottawa"));
Expression exp =
    emp.get("salary").equal(subQuery).and(emp.get("address").get("city").equal("Ottawa"));

```

[Example 110–20](#) illustrates how to create an expression that matches all employees that have no projects.

**Example 110–20 A Subquery Expression Using a Not Exists Operation**

```

ExpressionBuilder emp = new ExpressionBuilder();
ExpressionBuilder proj = new ExpressionBuilder();
ReportQuery subQuery = new ReportQuery(Project.class, proj);
subQuery.addAttribute("id");
subQuery.setSelectionCriteria(proj.equal(emp.anyOf("projects")));
Expression exp = emp.notExists(subQuery);

```

## 110.5.2 How to Use Parallel Expressions

Parallel expressions enable you to compare unrelated objects. Parallel expressions require multiple expression builders, but do not require the use of report queries. Each expression must have its own expression builder, and you must use the constructor for expression builder that takes a `class` as an argument. The class does not have to be the same for the parallel expressions, and you can create multiple parallel expressions in a single query.

Only one of the expression builders is considered the primary expression builder for the query. This primary builder makes use of the zero argument expression constructor, and `TopLink` obtains its class from the query.

[Example 110–21](#) illustrates how to create an expression that matches all employees with the same last name as another employee of different gender, and accounts for the possibility that returned results could be a spouse.

**Example 110–21 A Parallel Expression on Two Independent Employees**

```

ExpressionBuilder emp = new ExpressionBuilder();
ExpressionBuilder spouse = new ExpressionBuilder(Employee.class);
Expression exp = emp.get("lastName").equal(spouse.get("lastName"))
    .and(emp.get("gender").notEqual(spouse.get("gender")));

```

## 110.6 Data Queries and Expressions

You can use expressions to retrieve data rather than objects. This is a common approach when you work with unmapped information in the database, such as foreign keys and version fields.

Expressions that query for objects generally refer to object attributes, which may in turn refer to other objects. Data expressions refer to tables and their fields. You can combine data expressions and object expressions within a single query. `TopLink` provides two main methods for expressions that query for data: `getField` and `getTable`.

## 110.6.1 How to Use the getField Method

The `getField` method lets you retrieve data from either an unmapped table or an unmapped field from an object. In either case, the field must be part of a table represented by that object's class; otherwise, TopLink raises an exception when you execute the query.

You can also use the `getField` method to retrieve the foreign key information for an object.

[Example 110–22](#) illustrates how to use the data expression method (operator) `getField` with an object.

### **Example 110–22 Using getField with an Object**

```
builder.getField("[FIELD_NAME]").greaterThan("[ARGUMENT]");
```

## 110.6.2 How to Use the getTable Method

The `getTable` method returns an expression that represents an unmapped table in the database. This expression provides a context from which to retrieve an unmapped field when you use the `getField` method.

[Example 110–23](#) illustrates how to combine both `getField` and `getTable` in the same expression.

### **Example 110–23 Using getTable and getField Together**

```
builder.getTable("[TABLE_NAME]").getField("[FIELD_NAME]").equal("[ARGUMENT]");
```

A common use for the `getTable` and `getField` methods is to retrieve information from a link table (or reference table) that supports a many-to-many relationship.

[Example 110–24](#) reads a many-to-many relationship that uses a link table and also checks an additional field in the link table. This code combines an object query with a data query, using the employee's manager as the basis for the data query. It also features parameterization for the project ID.

### **Example 110–24 Using a Data Query Against a Link Table**

```
ExpressionBuilder emp = new ExpressionBuilder();
Expression manager = emp.get("manager");
Expression linkTable = manager.getTable("PROJ_EMP");
Expression empToLink = emp.getField("EMPLOYEE
    .EMP_ID").equal(linkTable.getField("PROJ_EMP.EMP_ID"));
Expression projToLink = linkTable.getField("PROJ_EMP
    .PROJ_ID").equal(emp.getParameter("PROJECT.PROJ_ID"));
Expression extra = linkTable.getField("PROJ_EMP.TYPE").equal("W");
query.setSelectionCriteria((empToLink.and(projToLink)).and(extra));
```

## 110.7 Creating an Expression

You can create an expression using Oracle JDeveloper, TopLink Workbench, or Java.

Use either Oracle JDeveloper or TopLink Workbench for creating basic expressions for use in named queries (see [Section 110.7.1, "How to Create an Expression Using TopLink Workbench"](#)).

Use Java code to create more complex expressions and to take full advantage of the features in the expressions API (see [Section 110.7.2, "How to Create an Expression Using Java"](#)).

## 110.7.1 How to Create an Expression Using TopLink Workbench

To create TopLink expressions for named queries, use this procedure:

1. From the **Named Queries Format** tab, click **Edit** (or double-click a query string). The Expression Builder dialog box appears.

See [Section 108.8, "Named Queries"](#) for more information.

**Figure 110–1 Expression Builder Dialog Box**

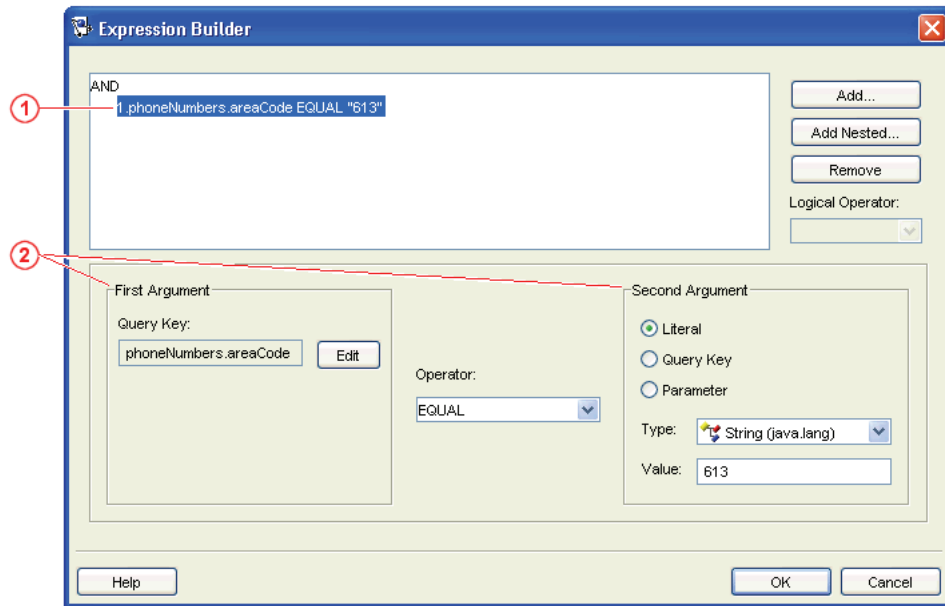


Figure 110–1 numbered callouts identify the following user-interface components:

1. Expression tree
  2. Arguments
2. Click **Add** or **Add Nested** to create a new expression. TopLink assigns a sequence number to each node and nested node.  
Click **Remove** to remove an existing expression.
  3. Select the node and use the **Logical Operator** list to specify the operator for the node (**AND**, **OR**, **Not AND**, or **Not OR**).

Use this table to complete the argument fields for each expression:

Field	Description
<b>First Argument</b>	Click <b>Edit</b> and select the query key for the first argument. The Choose Query Key dialog box appears. Continue with <a href="#">Section 110.7.1.1, "Adding Arguments"</a> .
<b>Operator</b>	Specify how TopLink should evaluate the expression. Valid operators include: Equal, Not Equal, Equal Ignore Case, Greater Than, Greater Than Equal, Less Than, Less Than Equal, Like, Not Like, Like Ignore Case, Is Null, and Not Null.

Field	Description
<b>Second Argument</b>	<p>Specify the second argument:</p> <ul style="list-style-type: none"> <li>▪ <b>Literal</b>—Select the <b>Type</b> and enter a literal value for <b>Value</b>.</li> <li>▪ <b>Query Key</b>—Click <b>Edit</b> and select the query key.</li> <li>▪ <b>Parameter</b>—Click <b>Add</b> to add a new parameter and then select from the list.</li> </ul> <p>Continue with <a href="#">Section 110.7.1.1, "Adding Arguments"</a></p>

Click **OK**. TopLink Workbench adds the expression to the **Named Queries** tab.

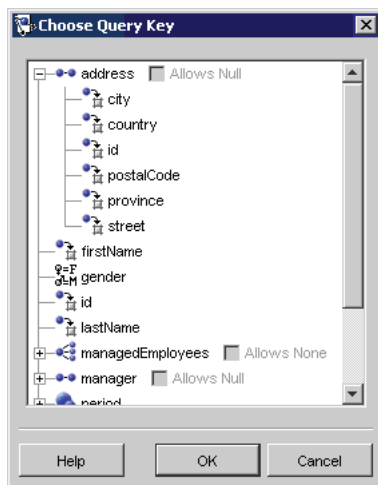
### 110.7.1.1 Adding Arguments

Each expression contains elements (*arguments*) to evaluate. Expressions using the Is Null or Not Null operators require only a single argument.

To add new arguments, use this procedure:

1. Select an existing expression or click **Add** (or **Add Nested**) to add a new expression to the named query.
2. For the **First Argument**, click **Edit**. The Choose Query Key dialog box appears.

**Figure 110–2 Choose Query Key**



3. Select the attribute, specify if the query allows a null value, and click **OK**.  
 Use the **Allows Null** and **Allows None** options to define an expression with an outer join.  
 Check the **Allows Null** option to use the ExpressionBuilder method `getAllowingNull`.  
 Check the **Allows None** option to use the ExpressionBuilder method `anyOfAllowingNone`.  
 For more information, see [Section 110.2.7.2, "Using TopLink Expression API for Joins"](#).
4. Use the **Operator** list to specify how TopLink should evaluate the expression.
5. For the **Second Argument**, select **Literal**, **Query Key**, or **Parameter**.

- For **Literal** arguments, choose the literal type (such as **String** or **Integer**) and enter the literal value.
- For **Query Key** arguments, click **Edit**. The Choose Query Key dialog box appears (see step 3 and [Figure 110–2](#)).
- For **Parameter** arguments, click **Add** to add a parameter and then use the list to select it.

Repeat this procedure for each expression or subexpression.

#### **Example 110–25 Sample Expression**

The following expression will find employees who:

- have a manager with the last name Jones or have no manager, and
- work on projects with the name Beta or project ID 4, and
- live in Canada and have a salary of more than 25,000, or live in the United States and have a salary of more than 37,500

```
AND
  1.manager(Allows Null).lastName EQUAL "Jones"
  2.OR
    2.1.projects.name LIKE "BETA"
    2.2.projects.id EQUAL "4"
  3.OR
    3.1.AND
      3.1.1.address.country EQUAL "Canada"
      3.1.2.salary GREATER THAN "25000"
    3.2.AND
      3.1.1.address.country EQUAL "United States"
      3.1.2.salary GREATER THAN "37500"
```

## 110.7.2 How to Create an Expression Using Java

To create an expression in Java code, use the `Expression` class or `ExpressionBuilder` method `get`.

The `ExpressionBuilder` acts as a substitute for the objects that you query. To construct a query, call methods on the `ExpressionBuilder` that correspond to the attributes of the objects. We recommend that you name `ExpressionBuilder` objects according to the type of objects against which you do a query.

---

**Note:** An instance of `ExpressionBuilder` is specific to a particular query. Do not attempt to build another query using an existing builder, because it still contains information related to the first query.

---

[Example 110–26](#) illustrates how to use the query key `lastName` to reference the field name `L_NAME`.

#### **Example 110–26 Using ExpressionBuilder to Build a Simple Expression**

```
Expression expression = new ExpressionBuilder().get("lastName").equal("Young");
```

[Example 110–27](#) illustrates how to create a complex expression by combining two smaller expressions with a logical and operator.

**Example 110–27 Combining Two Expressions with a Logical AND Operator**

```

ExpressionBuilder emp = new ExpressionBuilder();
Expression exp1, exp2;
exp1 = emp.get("firstName").equal("Ken");
exp2 = emp.get("lastName").equal("Young");
return exp1.and(exp2);

```

[Example 110–28](#) illustrates how to create an expression using the `notLike` operator.

**Example 110–28 Using Database Function `notLike` in an Expression**

```

Expression expression = new ExpressionBuilder().get("lastName").notLike("%ung");

```

## 110.8 Creating and Using a User-Defined Function

Different databases sometimes implement the same functions in different ways. For example, an argument that specifies that data returns in ascending order might be `ASC` or `ASCENDING`. To manage differences, TopLink recognizes functions and other operators that vary according to the relational database.

Although most platform-specific operators exist in TopLink, if necessary, you can create your own operators.

To create a user-defined function, use the `ExpressionOperator` class.

An `ExpressionOperator` has a selector and a `Vector` of strings:

- The selector is the identifier (*id*) by which users refer to the function.
- The strings are the constant strings used in printing the function. When printed, the strings alternate with the function arguments.

You can also specify whether the operator is prefix or postfix. In a prefix operator, the first constant string prints before the first argument; in a postfix, it prints afterwards.

Where you create a user-defined function and how you add it to the TopLink expression framework depends on whether you want the new function available to all database platforms or to only a specific database platform.

This section describes the following:

- [How to Make a User-Defined Function Available to a Specific Platform](#)
- [How to Make a User-Defined Function Available to All Platforms](#)

### 110.8.1 How to Make a User-Defined Function Available to a Specific Platform

To make the function that overrides a specific operation on your own platform, use the following procedure:

1. Create a subclass of the desired `DatabasePlatform` (from `oracle.toplink.platform.database` or `oracle.toplink.platform.database.oracle` package) that provides a public method that calls the protected superclass method `addOperator`:

```

...
public class MyDatabasePlatform extends DatabasePlatform {
    protected void initializePlatformOperators() {
        super.initializePlatformOperators();
        // Create user-defined function
        ExpressionOperator toUpper = new ExpressionOperator();
        toUpper.setSelector(ExpressionOperator.ToUpperCase);
        Vector v = new Vector();
    }
}

```

```

v.addElement("UPPERCASE(");
v.addElement(")");
toUpper.printAs(v);
toUpper.bePrefix();
toUpper.setNodeClass(FunctionExpression.class);

// Make it available to this platform only
addOperator(toUpper);
}
}

```

2. Configure your session to use your platform subclass (see [Section 20.2](#), "[Configuring Relational Database Platform at the Project Level](#)" or [Section 98.2](#), "[Configuring a Relational Database Platform at the Session Level](#)").

## 110.8.2 How to Make a User-Defined Function Available to All Platforms

To make the function available to all platforms, use `ExpressionOperator` method `addOperator`, as [Example 110–29](#) shows.

### **Example 110–29 Adding a toUpper Function for All Platforms**

```

ExpressionOperator toUpper = new ExpressionOperator();
toUpper.setSelector(600);
Vector v = new Vector();
v.addElement("NUPPER(");
v.addElement(")");
toUpper.printAs(v);
toUpper.bePrefix();
toUpper.setNodeClass(FunctionExpression.class);

ExpressionOperator.addOperator(toUpper);

```

---



---

**Note:** Represent the number in the `setSelector` method by a constant value. Ensure that this number is greater than 500 (numbers below 500 are reserved in `TopLink`).

---



---

### 110.8.2.1 Using a User-Defined Function

Regardless of whether you added the function for all platforms or for a specific platform, [Example 110–30](#) illustrates how to use the `Expression` method `getFunction` to access the user-defined expression operator represented by a constant with the value 600.

### **Example 110–30 Accessing a User-Defined Function**

```

ReadObjectQuery query = new ReadObjectQuery(Employee.class);
ExpressionBuilder builder = query.getExpressionBuilder();
Expression functionExpression = builder.get("firstName").
    getFunction(600).equal("BOB");
query.setSelectionCriteria(functionExpression);
session.executeQuery(query);

```





---

---

## Using Advanced Query API

This section explains more advanced TopLink query API calls and techniques that you are most likely to use later in the development cycle.

This chapter includes the following sections:

- [Using Redirect Queries](#)
- [Using Historical Queries](#)
- [Using Queries with Fetch Groups](#)
- [Using Read-Only Queries](#)
- [Querying on Interfaces](#)
- [Querying on an Inheritance Hierarchy](#)
- [Appending Additional Join Expressions](#)
- [Using Queries on Variable One-to-One Mappings](#)
- [Using Oracle Database Features](#)
- [Using EJB 2.n CMP Finders](#)
- [Handling Cursor and Stream Query Results](#)
- [Handling Query Results Using Pagination](#)
- [Using Queries and the Cache](#)

For more information about the available query API, see *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.

### 111.1 Using Redirect Queries

A redirect query is a named query that delegates query execution control to your application. redirect queried allow you to define the query implementation in code as a static method.

To perform complex operations, you can combine query redirectors with the TopLink query framework.

This section describes [How to Create a Redirect Query](#).

#### 111.1.1 How to Create a Redirect Query

To perform complex operations, you can combine query redirectors with the TopLink query framework. To create a redirector, implement the `oracle.toplink.queryframework.QueryRedirector` interface. The query

mechanism executes the `Object invokeQuery(DatabaseQuery query, Record arguments, Session session)` method and waits for the results.

`TopLink` provides one preimplemented redirector, the `MethodBasedQueryRedirector` method. To use this redirector, create a static `invoke` method on a class, and use the `setMethodName(String)` call to specify the method to invoke.

### **Example 111–1 Redirect Query**

```
ReadObjectQuery query = new ReadObjectQuery(Employee.class);
query.setName("findEmployeeByAnEmployee");
query.addArgument("employee");

MethodBaseQueryRedirector redirector = new
    MethodBaseQueryRedirector(QueryRedirectorTest.class, "findEmployeeByAnEmployee");
query.setRedirector(redirector);
Descriptor descriptor = getSession().getDescriptor(query.getReferenceClass());
descriptor.getQueryManager().addQuery(query.getName(), query);

Vector arguments = new Vector();
arguments.addElement(employee);
objectFromDatabase =
    getSession().executeQuery("findEmployeeByAnEmployee", Employee.class, arguments);

public class QueryRedirectorTest {

    public static Object findEmployeeByAnEmployee(
        DatabaseQuery query,
        oracle.toplink.sessions.Record arguments,
        oracle.toplink.sessions.Session
        session) {
        ((ReadObjectQuery) query).setSelectionObject(arguments.get("employee"));
        return session.executeQuery(query);
    }
}
```

## 111.2 Using Historical Queries

To make a query time-aware, you specify an `AsOfClause` that `TopLink` appends to the query. Use the `AsOfClause` class if your historical schema is based on time stamps or the `AsOfSCNClause` class if your historical schema is based on database system change numbers. You can specify an `AsOfClause` at the time you acquire a historical session so that `TopLink` appends the same clause to all queries, or you can specify an `AsOfClause` on a query-by-query basis.

[Example 111–2](#) shows how to create a query that uses a particular `AsOfClause`. This query will read all `Employee` objects as of the time specified by `timestamp` using the appropriate history tables described by the `HistoryPolicy` set on the `Employee` descriptor.

### **Example 111–2 Using a Historical Session**

```
ReadAllQuery historicalQuery = new ReadAllQuery(Employee.class);
AsOfClause asOfClause = new AsOfClause(timestamp);
historicalQuery.setAsOfClause(asOfClause);
historicalQuery.dontMaintainCache();
List pastEmployees = (List)historicalSession.executeQuery(historicalQuery);
```

## 111.3 Using Queries with Fetch Groups

You can use a fetch group with a `ReadObjectQuery` or `ReadAllQuery`. When you execute the query, `TopLink` retrieves only the attributes in the fetch group. `TopLink` automatically executes a query to fetch all the attributes excluded from this subset when and if you call a getter method on any one of the excluded attributes.

---

**Note:** When you use fetch groups outside of CMP, use weaving (see [Section 2.10, "Using Weaving"](#)).

---

This section describes the following:

- [How to Configure Default Fetch Group Behavior](#)
- [How to Query with a Static Fetch Group](#)
- [How to Query with a Dynamic Fetch Group](#)

For more information about fetch groups, see [Section 108.7.1.6, "Fetch Groups and Object-Level Read Queries"](#).

### 111.3.1 How to Configure Default Fetch Group Behavior

You can optionally designate at most one fetch group as the default fetch group for a descriptor's reference class.

If you execute a `ReadObjectQuery` or `ReadAllQuery` without specifying a fetch group, `TopLink` will use the default fetch group unless you configure the query otherwise, as [Example 111-3](#) shows.

#### **Example 111-3 Configuring Default Fetch Group Behavior**

```
// at the descriptor level
FetchGroup group = new FetchGroup("nameOnly");
group.addAttribute("firstName");
group.addAttribute("lastName");
employeeDescriptor.getFetchGroupManager().addFetchGroup(group);
// set the default fetch group
employeeDescriptor.getFetchGroupManager().setDefaultFetchGroup(group);

// when query1 is executed, the default fetch group applies
ReadAllQuery query1 = new ReadAllQuery(Employee.class);

// when query2 is executed, the default fetch group does not apply
ReadAllQuery query2 = new ReadAllQuery(Employee.class);
query2.setShouldUsedefaultFetchGroup(false);
```

### 111.3.2 How to Query with a Static Fetch Group

[Example 111-4](#) shows how to configure a `ReadObjectQuery` for the `Employee` class with a `FetchGroup` named `nameOnly` previously stored in the `FetchGroupManager` owned by the `Employee` class's descriptor.

#### **Example 111-4 Configuring a Query with a FetchGroup Using the FetchGroupManager**

In this example, only the `Employee` attributes `firstName` and `lastName` are fetched. If you call the `Employee` method `get` for any other attribute, `TopLink` executes another query to retrieve all unfetched attribute values. Thereafter, calling that `get` method will return the value directly from the object.

```
// create static fetch group at the descriptor level
FetchGroup group = new FetchGroup("nameOnly");
group.addAttribute("firstName");
group.addAttribute("lastName");
descriptor.getFetchGroupManager().addFetchGroup(group);

// use static fetch group at query level
ReadAllQuery query = new ReadAllQuery(Employee.class);
query.setFetchGroupName("nameOnly");
```

### 111.3.3 How to Query with a Dynamic Fetch Group

[Example 111–5](#) shows how to create a `FetchGroup` instance dynamically, at the time you create and execute a query, and configure the query with that `FetchGroup` directly.

In this example, only the `firstName`, `lastName`, and `salary` attributes are fetched. If you call the `Employee` method `get` for any other attribute, `TopLink` executes another query to retrieve all unfetched attribute values. Thereafter, calling that `get` method will return the value directly from the object.

#### *Example 111–5 Configuring a Query with a FetchGroup Dynamically*

```
// dynamic fetch group query
ReadAllQuery query = new ReadAllQuery(Employee.class);
FetchGroup group = new FetchGroup("nameAndSalary");
group.addAttribute("firstName");
group.addAttribute("lastName");
group.addAttribute("salary");
query.setFetchGroup(group);
```

## 111.4 Using Read-Only Queries

[Example 111–6](#) shows how to create an object-level read query to return data that you know is read-only. Using such a query for read-only data can improve performance.

#### *Example 111–6 Configuring an ObjectLevelReadQuery as Read-Only*

```
ReadAllQuery query = new ReadAllQuery(Employee.class);
query.setIsReadOnly(true);
```

For more information, see the following:

- [Section 108.7.1.4, "Read-Only Query"](#)
- [Section 12.12.5, "How to Use Read-Only Queries for Optimization"](#)

## 111.5 Querying on Interfaces

When you define descriptors for an interface to enable querying, `TopLink` supports querying on an interface, as follows:

- If there is only a single implementor of the interface, the query returns an instance of the concrete class.
- If there are multiple implementors of the interfaces, the query returns instances of all implementing classes.

## 111.6 Querying on an Inheritance Hierarchy

When you query on a class that is part of an inheritance hierarchy, the session checks the descriptor to determine the type of the class, as follows:

- If you configure the descriptor to read subclasses (the default configuration), the query returns instances of the class and its subclasses.
- If you configure the descriptor not to read subclasses, the query returns only instances of the queried class, but no instances of the subclasses.
- If you configure the descriptor to outer-join subclasses, the query returns instances of the class and its subclasses.
- If neither of these conditions applies, the class is a leaf class and does not have any subclasses. The query returns instances of the queried class.

## 111.7 Appending Additional Join Expressions

You can set the query manager to automatically append an expression to every query it performs on a class. For example, you can add an expression that filters the database for the valid instances of a given class.

Use this to do the following:

- Filter logically deleted objects
- Enable two independent classes to share a single table without inheritance
- Filter historical versions of objects

### 111.7.1 How to Append Additional Join Expressions Using Java

Using Java, configure a descriptor with additional join expressions by creating an amendment method (see [Section 119.35, "Configuring Amendment Methods"](#)), and then using the `DescriptorQueryManager` methods `setAdditionalJoinExpression` or `setMultipleTableJoinExpression`, as [Example 111-7](#) shows.

#### **Example 111-7 Registering a Query That Includes a Join Expression**

In [Example 111-7](#), the `join` expression filters invalid instances of `employee` from the query.

```
public static void addToDescriptor(Descriptor descriptor) {
    ExpressionBuilder builder = new ExpressionBuilder();
    descriptor.getQueryManager().setAdditionalJoinExpression(
        (builder.getField("EMP.STATUS").notEqual("DELETED")).and(
            builder.getField("EMP.STATUS").notEqual("HISTORICAL"))
    );
}
```

## 111.8 Using Queries on Variable One-to-One Mappings

`TopLink` does not provide a method to directly query against variable one-to-one mappings. To query against this type of mapping, combine `TopLink DirectQueryKeys` and `TopLink ReportQueries` to create query selection criteria for classes that implement the interface, as follows:

1. Create two `DirectQueryKeys` to query for the possible implementors of the interface:

- The first `DirectQueryKey` is for the class indicator field for the variable one-to-one mapping.
  - The second `DirectQueryKey` is for the foreign key to the class or table that implements the interface.
2. Create a `subSelect` statement for each concrete class that implements the interface included in the query selection criteria.
  3. Implement a `ReportQuery`.

#### **Example 111–8 Creating DirectQueryKeys**

```
// The DirectQueryKeys as generated in the TopLink project Java
// source code from TopLink Workbench
...
descriptor.addDirectQueryKey("locationTypeCode", "DEALLOCATION.DEALLOCATIONOBJECTTYPE");
descriptor.addDirectQueryKey("locationTypeId", "DEALLOCATION.DEALLOCATIONOBJECTID");
;
```

## 111.9 Using Oracle Database Features

If you are using Oracle Database, you can take advantage of TopLink support for the following Oracle Database features:

- Oracle Hints (see [Section 111.9.1, "How to Use Oracle Hints"](#))
- Hierarchical Queries (see [Section 111.9.2, "How to Use Hierarchical Queries"](#))

### 111.9.1 How to Use Oracle Hints

Oracle Hints is Oracle Database feature through which you can make decisions usually reserved for the optimizer. You use hints to specify things such as join order for a join statement, or the optimization approach of an SQL call.

The TopLink query framework supports Oracle Hints with the following API:

```
setHintString("/*[hints or comments]*/");
```

TopLink adds the hint to the SQL string as a comment immediately following a `SELECT`, `UPDATE`, `INSERT`, or `DELETE` statement.

Add hints to a read query as follows:

1. Create a `ReadObjectQuery` or a `ReadAllQuery`
2. Set the selection criteria.
3. Add hints as needed.

For example, the following code uses the `FULL` hint (which explicitly chooses a full table scan for the specified table):

```
// Create the query and set Employee as its reference class
ReadObjectQuery query = new ReadObjectQuery(Employee.class);
// Retrieve ExpressionBuilder from the query
ExpressionBuilder builder = query.getExpressionBuilder();
query.setSelectionCriteria(builder.get("id").equal(new Integer(1)));
// Add the hint
query.setHintString("/*+ FULL */");
```

This code generates the following SQL:

```
SELECT /*+ FULL */ FROM EMPLOYEE WHERE ID=1
```

To add hints to `WRITE`, `INSERT`, `UPDATE`, and `DELETE`, create custom queries for these operations in the TopLink query framework, then specify hints as required. For more information, see the following:

- [Section 23.4, "Configuring Custom SQL Queries for Basic Persistence Operations"](#)
- [Section 76.5, "Configuring Custom EIS Interactions for Basic Persistence Operations"](#)

For more information about the available hints, see Oracle Database documentation.

## 111.9.2 How to Use Hierarchical Queries

Hierarchical Queries is Oracle Database mechanism that lets you select database rows based on hierarchical order. For example, you can design a query that reads the row of a given employee, followed by the rows of people this employee manages, followed by their managed employees, and so on.

To create a hierarchical query, use the `setHierarchicalQueryClause` method. This method takes three parameters, as follows:

```
setHierarchicalQueryClause(startWith, connectBy, orderSibling)
```

This expression requires all three parameters, as described in the subsequent text.

### 111.9.2.1 Using startWith Parameter

The `startWith` parameter in the expression specifies the first object in the hierarchy. This parameter mirrors Oracle Database `START WITH` clause.

To include a `startWith` parameter, build an expression to specify the appropriate object, and pass it as a parameter in the `setHierarchicalQueryClause` method. If you do not specify the root object for the hierarchy, set this value to `null`.

### 111.9.2.2 Using connectBy Parameter

The `connectBy` parameter specifies the relationship that creates the hierarchy. This parameter mirrors Oracle Database `CONNECT BY` clause.

Build an expression to specify the `connectBy` parameter, and pass it as a parameter in the `setHierarchicalQueryClause` method. Because this parameter defines the nature of the hierarchy, it is required for the `setHierarchicalQueryClause` implementation.

### 111.9.2.3 Using orderSibling Parameter

The `orderSibling` parameter in the expression specifies the order in which the query returns sibling objects in the hierarchy. This parameter mirrors Oracle Database `ORDER SIBLINGS` clause.

To include an `orderSibling` parameter, define a vector, and to include the order criteria, use the `addElement` method. Pass the vector as the third parameter in the `setHierarchicalQueryClause` method. If you do not specify an order, set this value to `null`.

#### **Example 111–9 Hierarchical Query**

```
ReadAllQuery raq = new ReadAllQuery(Employee.class);
// Specifies a START WITH expression
Expression startExpr = expressionBuilder.get("id").equal(new Integer(1));
// Specifies a CONNECT BY expression
Expression connectBy = expressionBuilder.get("managedEmployees");
// Specifies an ORDER SIBLINGS BY vector
```

```
Vector order = new Vector();
order.addElement(expressionBuilder.get("lastName"));
order.addElement(expressionBuilder.get("firstName"));
raq.setHierarchicalQueryClause(startExpr, connectBy, order);
Vector employees = uow.executeQuery(raq);
```

This code generates the following SQL:

```
SELECT * FROM EMPLOYEE START WITH ID=1 CONNECT BY PRIOR ID=MANAGER_ID ORDER
SIBLINGS BY LAST_NAME, FIRST_NAME
```

## 111.10 Using EJB 2.n CMP Finders

This section describes how to use EJB finders in TopLink, including the following:

- [How to Create a Finder](#)
- [How to Use DatabaseQuery Finders](#)
- [How to Use Named Query Finders](#)
- [How to Use Primary Key Finders](#)
- [How to Use EJB QL Finders](#)
- [How to Use SQL Finders](#)
- [How to Use Redirect Finders](#)
- [How to Use the ejbSelect Method](#)

### 111.10.1 How to Create a Finder

In general, to create a finder for an entity bean that uses the TopLink query framework, you must define, declare, and configure it.

For predefined finders (see [Section 108.15.1, "Predefined Finders"](#)), you do not need to explicitly create a finder.

For default finders (see [Section 108.15.2, "Default Finders"](#)), you only need to define the finder method.

To create a finder for an entity bean that uses the TopLink query framework, follow these steps:

1. Define the finder method on the entity bean's `remoteHome` or `localHome` interface.

For entity beans with container-managed persistence, define the method on the entity bean's `Home` interface.

For default finders (see [Section 108.15.2, "Default Finders"](#)), you must define the finder as follows:

- `<RETURN-TYPE> findBy<CMP-FIELD-NAME> (<CMP-FIELD-TYPE>)`
- the first letter of `<CMP-FIELD-NAME>` must be capitalized
- `<RETURN-TYPE>` may be a single bean type or `Collection`.

For example:

```
EmployeeBean (Integer id, String name)
EmployeeHome ..{
    Employee findById(Integer id) throws...;
    Collection findByName(String name) throws...;
}
```



---



---

**Note:** If you are using default finders (see [Section 108.15.2, "Default Finders"](#)), you are finished. TopLink will implement the finder for you at run time.

---



---

2. Declare the finder in the `ejb-jar.xml` file (see [Section 111.10.1.1, "ejb-jar.xml Finder Options"](#)).
3. Start TopLink Workbench.
4. Click the project icon in the **Navigator** and select: **Selected > Update Project from ejb-jar.xml** to read in the finders.
5. Go to the **Queries > Named Queries** tab for the bean (see [Section 109.3, "Using Named Queries"](#)).
6. Select and configure the finder.

---



---

**Notes:** For predefined finders `findOneByQuery` and `findManyByQuery`, the client creates a query at run time and passes it as a parameter to the finder. Because of this, do not configure query options on these finders. Instead, configure options on the query passed into the finder. For more information about predefined finders, see [Section 108.15.1, "Predefined Finders"](#).

---



---

7. If required, create an implementation for the query. Some query options require a query definition in code on a helper class, but most common queries do not.

When you use TopLink CMP, define finder methods on the bean's `Home` interface, not in the entity bean itself. TopLink CMP provides this functionality and offers several strategies to create and customize finders. The EJB container and TopLink automatically generate the implementation.

### 111.10.1.1 ejb-jar.xml Finder Options

The `ejb-jar.xml` file contains a project's EJB entity bean information, including definitions for any finders used for the beans. To create and maintain the `ejb-jar.xml` file, use either a text editor or TopLink Workbench.

The `entity` tag encapsulates a definition for an EJB entity bean. Each bean has its own `entity` tag that contains several other tags that define bean functionality, including bean finders.

[Example 111-10](#) illustrates the structure of a typical finder defined within the `ejb-jar.xml` file.

---



---

**Note:** Use a combination of an escape character and a double-quote ( `\` ) when defining your query using EJB QL. For more information on correct query syntax, see a note at the end of [Configuring Named Query Selection Criteria](#).

---



---

#### **Example 111-10 A Simple Finder Within the ejb-jar.xml File**

```
<entity>...
  <query>
    <query-method>
      <method-name>findLargeAccounts</method-name>
      <method-params>
```

```

        <method-param>double</method-param>
    </method-params>
</query-method>
<ejb-ql><![CDATA[SELECT OBJECT(account) FROM AccountBean account WHERE
    account.balance > ?1]]></ejb-ql>
</query>
...
</entity>

```

The `entity` tag contains zero or more query elements. Each query tag corresponds to a finder method defined on the bean's home or local Home interface.

---

**Note:** You can share a single query between both Home interfaces, as follows:

- Define the same finder (same name, return type, and parameters) on both Home interfaces.
  - Include a single query element in the `ejb-jar.xml` file.
- 

The following are the elements defined in the query section of the `ejb-jar.xml` file:

- `description` (optional): Provides a description of the finder.
- `query-method`: Specifies the method for a finder or `ejbSelect` query.
- `method-name`: Specifies the name of a finder or select method in the entity bean implementation class.
- `method-params`: Contains a list of the fully qualified Java type names of the method parameters.
- `method-param`: Contains the fully qualified Java type name of a method parameter.
- `result-type-mapping` (optional): Specifies how to map an abstract schema type returned by a query for an `ejbSelect` method. You can map the type to an `EJBLocalObject` or `EJBObject` type. Valid values are `Local` or `Remote`.
- `ejb-ql`: Used for all EJB QL finders. It contains the EJB QL query string that defines the finder or `ejbSelect` query. Leave this element empty for non-EJB QL finders.

## 111.10.2 How to Use DatabaseQuery Finders

TopLink provides a predefined finder that takes a `DatabaseQuery` such as a `ReadAllQuery`. To use this feature in a bean, add the following finder definition to the Home interface of your bean:

```
public Collection findManyByQuery(ReadAllQuery query) throws RemoteException,
    FinderException;
```

```
public <EJBLocal/Remote> findOneByQuery(ReadObjectQuery query) throws
    RemoteException, FinderException;
```

To execute a `ReadAllQuery` finder, create the query on the client, as [Example 111-11](#) shows.

### **Example 111-11 A ReadAllQuery Finder**

```
... {
    ReadAllQuery query = new ReadAllQuery(Employee.class);
    query.addJoinedAttribute("address");

```

```
Enumeration employees = getEmployeeHome().findManyByQuery(query);
}
```

### 111.10.3 How to Use Named Query Finders

You can implement an EJB finder method (including TopLink predefined finders) as a named query. For more information, see [Section 109.3, "Using Named Queries"](#). You execute such a finder as you would any other.

### 111.10.4 How to Use Primary Key Finders

TopLink provides a predefined finder (`findByPrimaryKey`) that takes a primary key as an `Object`.

**Example 111–12 Executing a Primary Key Finder**

```
... {
    Employee employee = getEmployeeHome().findByPrimaryKey(primaryKey);
}
```

### 111.10.5 How to Use EJB QL Finders

EJB QL is the standard query language first defined in the EJB 2.0 specification. TopLink supports EJB QL. EJB QL finders let you specify an EJB QL string as the implementation of the query.

---

**Note:** Use a combination of an escape character and a double-quote ( `\` ) when defining your query using EJB QL. For more information on correct query syntax, see a note at the end of [Section 119.7.1.3, "Configuring Named Query Selection Criteria"](#).

---

EJB QL offers the following advantages:

- It is the EJB 2.0 and 2.1 standard for queries.
- You can use it to construct most queries.
- You can implement dependent object queries with EJB QL.

The disadvantage of EJB QL is that it is difficult to use when you construct complex queries.

**To create an EJB QL finder, use this procedure:**

1. Declare the finder on either the `LocalHome` or the `RemoteHome` interface.
2. Start TopLink Workbench.
3. Reimport the `ejb-jar.xml` file to synchronize the project to the file.

TopLink Workbench synchronizes changes between the project and the `ejb-jar.xml` file.

The following is an example of a simple EJB QL query that requires one parameter. In this example, the question mark ( `?` ) in `?1` specifies a parameter:

```
SELECT OBJECT(employee) FROM Employee employee WHERE (employee.name =?1)
```

**To create an EJB QL finder for an entity bean with container-managed persistence, use this procedure:**

1. Declare the finder in the `ejb-jar.xml` file, and enter the EJB QL string in the `ejb-ql` tag.
2. Declare the finder on the `Home` interface, the `LocalHome` interface, or both, as required.
3. Start TopLink Workbench.
4. Specify the `ejb-jar.xml` file location and choose **File > Updated Project** from the `ejb-jar.xml` file to read in the finders.
5. Go to the **Queries > Finders > Named Queries** tab for the bean.
6. Add a finder, and give it the same name as the finder you declared on your bean's home. Then add any required parameters.
7. Select and configure the finder.

The following is an example of a simple EJB QL query that requires one parameter. In this example, the question mark ("?") in `?1` specifies a parameter.

```
SELECT OBJECT(employee) FROM Employee employee WHERE (employee.name =?1)
```

### 111.10.6 How to Use SQL Finders

You can use custom SQL code to specify finder logic. SQL lets you implement logic that might not be possible to express with TopLink expressions or EJB QL.

**To create a SQL finder, use this procedure:**

1. Declare the finder in the `ejb-jar.xml` file, and leave the `ejb-ql` tag empty.
2. Start TopLink Workbench.
3. Specify the `ejb-jar.xml` file location and choose **File > Updated Project** from the `ejb-jar.xml` file to read in the finders.
4. Go the **Queries > Named Queries** tab for the bean.
5. Select the finder, select the **SQL** radio button, and enter the SQL string.
6. Configure the finder.

The following is an example of a simple SQL finder that requires one parameter. In this example, the number sign character (`#`) is used to bind the argument `projectName` within the SQL string:

```
SELECT * FROM EJB_PROJECT WHERE (PROJ_NAME = #projectName)
```

### 111.10.7 How to Use Redirect Finders

Redirect finders let you specify a finder in which the implementation is defined as a static method on an arbitrary helper class. When you invoke the finder, it redirects the call to the specified static method.

For more information about redirect queries, see [Section 108.10, "Redirect Queries"](#).

The finder can have any arbitrary parameters. If the finder includes parameters, TopLink packages them into a `Vector` and passes them to the redirect method.

Redirect finders offer several advantages. Because you define the redirect finder implementation independently from the bean that invokes it, you can build the redirect finder to accept any type and number of parameters. This lets you create a

generic redirect finder that accepts several different parameters and return types, depending on input parameters.

A common strategy for using redirect finders is to create a generic finder that does the following:

- Includes logic to perform several tasks
- Reads the first passed parameter to identify the type of finder requested and select the appropriate logic

The redirect method contains the logic required to extract the relevant data from the parameters and uses it to construct a TopLink query.

The main disadvantage of redirect finders is that they are complex and can be difficult to configure. They also require an extra helper method to define the query. However, because they support complex logic, they are often the best choice when you need to implement logic unrelated to the bean on which the redirect method is called.

**To create a redirect finder, use the following procedure:**

1. Declare the finder in the `ejb-jar.xml` file, and leave the `ejb-ql` tag empty.
2. Declare the finder on the `Home` interface, the `localHome` interface, or both, as required.
3. Create an amendment method.  
For more information, see [Section 119.35, "Configuring Amendment Methods"](#).
4. Start TopLink Workbench.
5. Choose **Advanced Properties > After Load** from the menu for the bean.
6. Specify the class and name of the static method to enable the amendment method for the descriptor.

The amendment method then adds a query to the descriptor's query manager, as follows:

```
ReadAllQuery query = new ReadAllQuery();
query.setRedirector(new MethodBaseQueryRedirector (examples.ejb.cmp20.advanced.
    FinderDefinitionHelper.class, "findAllEmployeesByStreetName"));
descriptor.getQueryManager().addQuery ("findAllEmployeesByStreetName", query);
```

The redirect method must return either a single entity bean (object) or a `Vector`. Here are the possible method signatures:

```
public static Object redirectedQuery(oracle.toplink.sessions.Sessions, Vector args)
and
```

```
public static Vector redirectedQuery(oracle.toplink.sessions.Sessions, Vector args)
```

When you implement the query method, ensure that the method returns the correct type. For methods that return more than one bean, set the return type to `java.util.Vector`. TopLink converts this result to `java.util.Enumeration` (or `Collection`) if required.

---

**Note:** The redirect method also interprets a TopLink session as a parameter. For more information about a TopLink session, see [Part XXI, "TopLink Sessions"](#).

---

At run time, the client invokes the finder from the entity bean home and packages the arguments into the `args` vector in order of appearance from the finder method

signature. The client passes the vector to the redirect finder, which uses them to execute a TopLink expression.

**Example 111–13 A Simple Redirect Query Implementation**

```
public class RedirectorTest {

    private Session session;
    private Project project;

    public static void main(String args[]) {
        RedirectorTest test = new RedirectorTest();
        test.login();

        try {
            // Create the arguments to be used in the query
            Vector arguments = new Vector(1);
            arguments.add("Smith");

            // Run the query
            Object o = test.getSession()
                .executeQuery(test.redirectorExample(), arguments);
            o.toString();
        }
        catch (Exception e) {
            System.out.println("Exception caught -> " + e);
            e.printStackTrace();
        }
    }

    public ReadAllQuery redirectorExample() {
        // Create a redirector
        MethodBasedQueryRedirector redirector = new MethodBasedQueryRedirector();

        // Set the class containing the public static method
        redirector.setMethodClass(RedirectorTest.class);

        // Set the name of the method to be run
        redirector.setMethodName("findEmployeeByLastName");

        // Create a query and add the redirector previously created
        ReadAllQuery readAllQuery = new ReadAllQuery(Employee.class);
        readAllQuery.setRedirector(redirector);
        readAllQuery.addArgument("lastName");

        return readAllQuery;
    }

    // Call the static method
    public static Object findEmployeeByLastName(
        oracle.toplink.sessions.Session session,
        Vector arguments) {

        // Create a query and set Employee as its ref. class
        ReadAllQuery raq = new ReadAllQuery(Employee.class);
        raq.addArgument("lastName");

        // Create the selection criteria
        ExpressionBuilder employee = query.getExpressionBuilder();
        Expression whereClause =
```

```

employee.get("lastName").equal(arguments.firstElement());

// Set the selection criteria
raq.setSelectionCriteria(whereClause);

return (Vector)session.executeQuery(raq, arguments);
}
...
}

```

### 111.10.8 How to Use the `ejbSelect` Method

The `ejbSelect` method is a query method intended for internal use within an entity bean instance. Specified on the abstract bean itself, the `ejbSelect` method is not directly exposed to the client in the home or component interface. Defined as abstract, each bean can include zero or more such methods.

Select methods have the following characteristics:

- The method name must have `ejbSelect` as its prefix.
- It must be declared as `public`.
- It must be declared as `abstract`.
- The `throws` clause must specify the `javax.ejb.FinderException`, although it may also specify application-specific exceptions as well.
- The `result-type-mapping` tag in the `ejb-jar.xml` file determines the return type for `ejbSelect` methods. Set the flag to `Remote` to return `EJBObjects`; set it to `Local` to return `EJBLocalObjects`.

The format for an `ejbSelect` method definition should be similar to the following:

```
public abstract type.ejbSelect<METHOD>(...);
```

The `ejbSelect` query return type is not restricted to the entity bean type on which the `ejbSelect` is invoked. Instead, it can return any type corresponding to a container-managed relationship or container-managed field.

Although the `ejbSelect` method is not based on the identity of the entity bean instance on which it is invoked, it can use the primary key of an entity bean as an argument. This creates a query that is logically scoped to a particular entity bean instance.

#### To create an `ejbSelect` method, use this procedure:

1. Update the `ejb-jar.xml` file as follows:
  - Declare the `ejbSelect` method.
  - Enter the EJB QL string in the `ejb-ql` tag.
  - Specify the return type in the `result-type-mapping` tag (if required).
2. Declare the `ejbSelect` on the abstract bean class.
3. Start TopLink Workbench.
4. Click the project icon in the **Navigator**, and select: **Selected > Update Project from ejb-jar.xml** to read in the finders.
5. Go the **Queries > Named Queries** tab for the bean.
6. Select and configure the `ejbSelect` method.

## 111.11 Handling Cursor and Stream Query Results

Cursors and streams are related mechanisms that let you work with large result sets efficiently. See [Section 108.5.3, "Stream and Cursor Query Results"](#) for more information.

[Table 111–1](#) lists the methods that TopLink provides for all subclasses of `DataReadQuery` and `ReadAllQuery` that you can use to make your query return its results as a cursor or stream.

**Table 111–1 Stream and Cursor Query Result Options**

Method	Query Returns	Description
<code>useScrollableCursor</code>	<code>ScrollableCursor</code>	Allows you access a database result set cursor, allowing you to move forward and backward through the result set.
<code>useCursoredStream</code>	<code>CursoredStream</code>	Allows you to access results one at a time in sequence, as results become available to the underlying database result set cursor.

Using a `ScrollableCursor` or `CursoredStream` combines the features of a TopLink with the ability of the database to cursor data, and breaks up the result set into smaller, more manageable pieces.

The behavior of a query that uses a `ScrollableCursor` or `CursoredStream` differs from other queries in that the elements requested by the client are sent to the client.

This section describes the following:

- [How to Handle Cursors and Java Iterators](#)
- [How to Handle Java Streams](#)
- [How to Optimize Streams](#)
- [How to Use Cursors and Streams with EJB 2.n CMP Finders](#)

### 111.11.1 How to Handle Cursors and Java Iterators

The TopLink scrollable cursor lets you scroll through a result set from the database without reading the whole result set in a single database read operation. The `ScrollableCursor` class implements the Java `ListIterator` interface to allow for direct and relative access within the stream. Scrollable cursors also let you scroll forward and backward through the stream.

#### 111.11.1.1 Traversing Data with Scrollable Cursors

The following methods let you navigate data with a scrollable cursor:

- `relative(int i)`: advances the row number in relation to the current row by one row
- `absolute(int i)`: places the cursor at an absolute row position, 1 being the first row

Several strategies are available for traversing data with cursors. For example, to start at the end of the data set and work toward the first record, do the following:

1. Call the `afterLast` method to place the cursor after the last row in the result set.



2. Use the `hasPrevious` method to determine whether there is a record above the current record. This method returns `false` when you reach the final record in the data set.
3. If the `hasPrevious` method returns `true`, call the `previous` method to move the cursor to the row prior to the current row and read that object.

These are common methods for data traversal, but they are not the only available methods. For more information about the available methods, see *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.

To use the `ScrollableCursor` object, the JDBC driver must be compatible with the JDBC 2.0 specifications.

#### **Example 111–14 Traversing with a Scrollable Cursor**

```
ReadAllQuery query = new ReadAllQuery(Employee.class);
query.useScrollableCursor();
ScrollableCursor cursor = (ScrollableCursor) session.executeQuery(query);

while (cursor.hasNext()) {
    System.out.println(cursor.next().toString());
}
cursor.close();
```

## 111.11.2 How to Handle Java Streams

Java streams let you retrieve query results as individual records or groups of records, which can result in a performance increase. You can use streams to build efficient TopLink queries, especially when the queries are likely to generate large result sets.

### 111.11.2.1 Using Cursored Stream Support

Cursored streams provide the ability to read back a query result set from the database in manageable subsets, and to scroll through the result set stream.

The `useCursoredStream` method of the `ReadAllQuery` class provides cursored stream support.

#### **Example 111–15 Cursored Streams**

```
CursoredStream stream;
ReadAllQuery query = new ReadAllQuery(Employee.class);
query.useCursoredStream();
stream = (CursoredStream) session.executeQuery(query);
```

The query returns an instance of `CursoredStream` rather than a `List`, which can be a more efficient approach. For example, consider the following two code examples. [Example 111–16](#) returns a `List` that contains all employee objects. If ACME has 10,000 employees, the `List` contains references to 10,000 `Employee` objects.

#### **Example 111–16 Using a List**

```
ReadAllQuery query = new ReadAllQuery(Employee.class);
Enumeration employeeEnumeration;

List employees = (List) session.executeQuery(query);
employeeEnumeration = employee.elements();

while (employeeEnumeration.hasMoreElements()) {
    Employee employee = (Employee) employeeEnumeration.nextElement();
    employee.doSomeWork();
}
```

```
}
```

The following example returns a `CursoredStream` instance rather than a `List`. The `CursoredStream` collection appears to contain all 10,000 objects, but initially contains a reference to only the first 10 `Employee` objects. It retrieves the remaining objects in the collection as they are needed. In many cases, the application never needs to read all the objects:

```
ReadAllQuery query = new ReadAllQuery(Employee.class);
query.useCursoredStream();

CursoredStream stream = (CursoredStream) session.executeQuery(query);
while (! stream.atEnd()) {
    Employee employee = (Employee) stream.read();
    employee.doSomeWork();
    stream.releasePrevious();
}
stream.close();
```

---

**Note:** The `releasePrevious` message is optional. This releases any previously read objects and frees system memory. Even though released objects are removed from the cursored stream storage, they may remain in the identity map.

---

### 111.11.3 How to Optimize Streams

To optimize `CursoredStream` performance, provide a *threshold* and *page size* to the `useCursoredStream(Threshold, PageSize)` method, as follows:

- The threshold specifies the number of objects to read into the stream initially. The default threshold is 10.
- The page size specifies the number of objects to read into the stream after the initial group of objects. This occurs after the threshold number of objects is read. Although larger page sizes result in faster overall performance, they introduce delays into the application when `TopLink` loads each page. The default page size is 5.

When you execute a batch-type operation, use the `dontMaintainCache` method with a cursored stream. A batch operation performs simple operations on large numbers of objects and then discards the objects. Cursored streams create the required objects only as needed, and the `dontMaintainCache` ensures that these transient objects are not cached.

### 111.11.4 How to Use Cursors and Streams with EJB 2.n CMP Finders

Large result sets can be resource-intensive to collect and process. To give the client more control over the returned results, configure `TopLink` finders to use cursors. This combines `TopLink`'s `CursoredStream` with the ability of the database to cursor data, and breaks up the result set into smaller, more manageable pieces.

---

**Note:** If you use the transactional attribute `REQUIRED` for an entity bean, wrap all read operations in `UserTransaction` methods `begin` and `commit` to ensure that read operations beyond the first page of the cursor have a transaction in which to work.

---

### 111.11.4.1 Building the Query

You can configure any finder that returns a `java.util.Collection` to use a cursor. When you create the query for the finder, add the `useCursoredStream` option to enable cursoring.

#### **Example 111–17 Cursored Stream in a Finder**

```
ReadAllQuery raq = new ReadAllQuery(ProjectBean.class);
ExpressionBuilder bldr = raq.getExpressionBuilder();
raq.useCursoredStream();
raq.addArgument("projectName");
raq.setSelectionCriteria(bldr.get("name").
like(bldr.getParameter("projectName")));
descriptor.getQueryManager().addQuery("findByNameCursored", query);
```

### 111.11.4.2 Executing the Finder from the Client

`TopLink` offers the following additional elements for traversing finder results:

- `isEmpty` method: As with `java.util.Collection`, `isEmpty` method returns a boolean value indicating whether or not the `Collection` is empty.
- `size` method: As with `java.util.Collection`, `size` method returns an integer indicating the number of elements in the `Collection`.
- `iterator` method: As with `java.util.Collection`, `iterator` method returns a `java.util.Iterator` for enumerating the elements in the `Collection`.

`TopLink` also offers an extended protocol for `oracle.toplink.ejb.cmp.wls.CursoredIterator` (based on `java.util.Iterator`):

- `close` method: Closes the cursor on the server. The client must call this method to close the database connection.
- `hasNext` method: Returns a boolean value indicating whether or not any more elements are in the result set.
- `next` method: Returns the next available element.
- `next(int count)` method: Retrieves a `Vector` of at most `count` elements from the available results, depending on how many elements remain in the result set.

[Example 111–18](#) illustrates client code executing a cursored finder.

#### **Example 111–18 Cursored Finder**

```
// import both CursoredCollection and CursoredIterator
import oracle.toplink.ejb.cmp.wls.*;
//... other imports as necessary
getTransaction().begin();
CursoredIterator cursoredIterator = (CursoredIterator)
getProjectHome().findByNameCursored("proj%").iterator();
Vector projects = new Vector();
for (int index = 0; index < 50; i++) {
Project project = (Project)cursoredIterator.next();
projects.addElement(project);
}
// Rest all at once ...
Vector projects2 = cursoredIterator.next(50);
cursoredIterator.close();
getTransaction().commit();
```

## 111.12 Handling Query Results Using Pagination

You can configure a query to retrieve a result set in pages, that is, a partial result as a List of `pageSize` (or less) results. [Example 111–19](#) demonstrates paging through the result set of a query using `ReadQuery` methods `setMaxRows` and `setFirstResult`.

For more information, see the following:

- [Section 12.12.8, "How to Use Result Set Pagination for Optimization"](#)
- [Section 12.12.6, "How to Use JDBC Fetch Size for Optimization"](#)

### **Example 111–19 Using `setMaxRows` and `setFirstResult` to Page Through a Result Set**

```
...
int pageSize = 100;
int firstResult = 0;
int maxRows = pageSize;
boolean hasNext = true;
List page = null;

while (hasNext) {
    query.setFirstResult(firstResult);
    query.setMaxRows(maxRows);
    page = (List) session.executeQuery(query);
    // process this page of results
    if (page.size() == 0) {
        hasNext = false;
    } else {
        firstResult = firstResult + pageSize;
        maxRows = maxRows + pageSize;
    }
}
...
```

## 111.13 Using Queries and the Cache

This section describes how to use caching options in TopLink queries, including the following:

- [How to Cache Results in a `ReadQuery`](#)
- [How to Configure Cache Expiration at the Query Level](#)

### 111.13.1 How to Cache Results in a `ReadQuery`

By default, each time you execute a `ReadQuery`, TopLink applies the current query configuration to the read operation. In doing so, TopLink will access the session cache, the data source, or both.

Some queries are known to return the same result set (for example, the number of units sold last year by the current sales person). After the first query execution, there is no need to actually execute the query if it is invoked again.

For these types of queries, you can use any TopLink `ReadQuery` and configure it to store its query results in an internal query cache.

After its first execution for a set of query parameters, the query will return its cached result set each time it is invoked with the same query parameters. This improves query performance for frequently executed queries. By default a query will cache the results sets for the last 100 queries of specific parameters. You can configure this query cache as part of the `QueryResultsCachePolicy`.

Enable this feature using `ReadQuery` method `cacheQueryResults` or by calling the `ReadQuery` method `setQueryResultsCachePolicy` with an instance of `QueryResultsCachePolicy`, and disable it using `ReadQuery` method `doNotCacheQueryResults`.

Before using this feature, consider the restrictions in [Section 108.16.7.1, "Internal Query Cache Restrictions"](#). For more information, see [Section 108.16.7, "How to Cache Query Results in the Query Cache"](#).

You can apply a cache invalidation policy to the query's internal cache (see [Section 111.13.2, "How to Configure Cache Expiration at the Query Level"](#)). For more information, see [Section 102.2.5, "Cache Invalidation"](#).

[Example 111–20](#) shows how to configure a `ReadQuery` to cache its results.

**Example 111–20 Configuring a ReadQuery to Cache Its Query Results**

```
ReadObjectQuery query = new ReadObjectQuery(Employee.class);

// Instruct the ReadQuery to cache its query results
query.cacheQueryResults();

// The first time you invoke it, the ReadQuery reads from the database, session
// cache, or both and stores the result set in its internal query cache
Employee employeeFirst = (Employee) session.executeQuery(query);
```

[Example 111–21](#) shows how to configure the `ReadQuery` to stop caching its results. The next time the query is executed, `TopLink` does not use the query cache. Instead, the query accesses the data source.

**Example 111–21 Configuring a ReadQuery to Stop Caching Its Query Results**

```
// Disable query caching
query.doNotCacheQueryResults();

// The ReadQuery does not use the query cahce and instead accesses the database
Employee employee = (Employee) session.executeQuery(query);
```

Alternatively, you can clear the query's internal cache using `ReadQuery` method `clearQueryResults` passing in your session. This clears the currently cached results and ensures that the next query execution reads from the database.

## 111.13.2 How to Configure Cache Expiration at the Query Level

You can configure a `ReadQuery` with a `CacheInvalidationPolicy`.

If you configure a query to cache results in its own internal cache (see [Section 111.13.1, "How to Cache Results in a ReadQuery"](#)), the cache invalidation policy allows the cached query result set to expire, based on a time-to-live or daily-expiry. This invalidation time is calculated from the time of the query execution that cached the query result set for the specific set of query parameters.

[Example 111–22](#) shows how to configure a `ReadQuery` so that a `TimeToLiveCacheInvalidationPolicy` is applied to all the objects returned by the query and cached in the query's internal cache.

**Example 111–22 Configuring a CacheInvalidationPolicy on a ReadQuery for the Query's Internal Cache**

```
// The TimeToLiveCacheInvalidationPolicy applies to all objects returned by the query and
// cached in the query's internal cache
readQuery.setQueryResultsCachePolicy(
    new QueryResultsCachePolicy(new TimeToLiveCacheInvalidationPolicy(1000))
```

);  
For more information, see [Section 102.2.5, "Cache Invalidation"](#).

---

# Introduction to TopLink Support for Oracle Spatial

This chapter provides an overview of the TopLink support for Oracle Spatial, as well as demonstrates the ways to extend TopLink to support the mapping and querying of Oracle Spatial columns (`MDSYS.SDO_GEOMETRY`).

For more information about Oracle Spatial, see <http://www.oracle.com/technology/products/spatial/index.html>

This chapter includes the following sections:

- [TopLink Support for Oracle Spatial](#)
- [Using Structure Converters](#)
- [Using JGeometry](#)

## 112.1 TopLink Support for Oracle Spatial

TopLink provides support for direct mappings of database columns of type `MDSYS.SDO_GEOMETRY` to attributes of the `oracle.spatial.geometry.JGeometry` data type.

TopLink also provides support for spatial operators (see [Section 112.3.3, "How to Perform Queries Using Spatial Operator Expressions"](#)) through the TopLink expression framework (see [Section 110, "Introduction to TopLink Expressions"](#)), as well as for custom object types that wrap `SDO_GEOMETRY`.

For information on using TopLink structure converter with application servers other than OC4J, see the relevant server documentation.

For information on configuring OC4J application server to use the TopLink structure converter, see *Oracle Fusion Middleware Administration and Application Deployment Guide for Oracle Containers for Java EE*

## 112.2 Using Structure Converters

In TopLink, a `oracle.toplink.platform.database.DatabasePlatform` (see [Section 96.1.3.1, "Database Platforms"](#)) stores a list of structure converters.

To create a custom converter, implement the `oracle.toplink.platform.database.converters.StructConverter` interface and register it on your direct-to-field mapping (see [Section 27.3, "Direct-to-Field Mapping"](#)).

To use the `StructConverter`, do the following:

1. Configure the database platform (see [Section 112.2.1, "How to Configure the Database Platform to Use Structure Converters"](#)).
2. Set up a mapping (see [Section 112.2.2, "How to Set Up Mappings Using Structure Converters"](#)).

### 112.2.1 How to Configure the Database Platform to Use Structure Converters

TopLink uses a database platform (see [Section 96.1.3.1, "Database Platforms"](#)) to control the usage of database vendor-specific and version-specific operations such as SQL dialect, stored procedure calls, sequencing, as well as platform-specific type handling. You need to configure the platform to allow TopLink to use the advanced features of the database.

To add your structure converter to the DatabasePlatform, call `addStructConverter(StructConverter converter)` method of the DatabasePlatform. Call this method within your TopLink session (server or database) prior to the session login (see [Section 89.3, "Configuring a Session Login"](#)).

### 112.2.2 How to Set Up Mappings Using Structure Converters

Use direct-to-field mappings (see [Section 27.3, "Direct-to-Field Mapping"](#)) to map your STRUCT types. For each mapping that maps to the type defined by the structure converter, set its field type to the STRUCT data type, as follows:

```
mapping.setFieldType(java.sql.Types.STRUCT);
```

## 112.3 Using JGeometry

To use the `oracle.spatial.geometry.JGeometry`, do the following:

1. Configure the database platform (see [Section 112.3.1, "How to Configure the Database Platform to Use JGeometry"](#)).
2. Set up a mapping (see [Section 112.3.2, "How to Map JGeometry Attributes"](#)).

You can query your mapped entities with expressions that use Spatial operators. For more information, see [Section 112.3.3, "How to Perform Queries Using Spatial Operator Expressions"](#).

### 112.3.1 How to Configure the Database Platform to Use JGeometry

To configure the database platform, add a structure converter in a form of the `oracle.toplink.platform.database.oracle.converters.JGeometryConverter` as follows:

```
databasePlatform.addStructConverter(new JGeometryConverter());
```

You must configure this platform within your TopLink session prior to the session login (see [Section 89.3, "Configuring a Session Login"](#)).

### 112.3.2 How to Map JGeometry Attributes

Use direct-to-field mappings (see [Section 27.3, "Direct-to-Field Mapping"](#)) to map your STRUCT types. For each mapping that maps to the type defined by the structure converter (`JGeometry`), set its field type to the STRUCT data type, as follows:

```
mapping.setFieldType(java.sql.Types.STRUCT);
```



### 112.3.3 How to Perform Queries Using Spatial Operator Expressions

With the configured database platform, you can read and write persistent entities with JGeometry attributes mapped to SDO\_GEOMETRY columns. With this support, you can query for these mapped entities with native SQL queries using Oracle Spatial operators (see [http://download-west.oracle.com/docs/cd/B19306\\_01/appdev.102/b14255/sdo\\_operat.htm#i76448](http://download-west.oracle.com/docs/cd/B19306_01/appdev.102/b14255/sdo_operat.htm#i76448)).

Spatial operators are special SQL functions supported by Oracle Database to enable querying and comparison of columns containing geometry types. The spatial operators take the following format:

```
<SPATIAL-OP>(geometry1, geometry2, parameters) = 'TRUE'
```

For more information on spatial operators, see *Oracle Spatial API Documentation*.

In its `oracle.toplink.expressions.spatial` package, TopLink provides the expression support for the following Spatial operators:

- SDO\_WITHIN\_DISTANCE
- SDO\_RELATE
- SDO\_FILTER
- SDO\_NN

Use the following methods of the `oracle.toplink.expressions.spatial.SpatialExpressionFactory` class to build expressions that use Spatial operators:

- `withinDistance`
- `relate`
- `filter`
- `nearestNeighbor`

All these methods have the following common set of parameters:

1. an expression (`oracle.toplink.expressions.Expression`) that points to JGeometry;
2. JGeometry object or an Expression;
3. an `oracle.toplink.expressions.spatial.SpatialParameters` object that defines the parameters to the function call.

The `SpatialParameters` class provides convenience methods that let you set the parameters representing the following:

- minimum resolution;
- maximum resolution;
- units;
- distance;
- query type;
- masks;
- String of parameters.

**Example 112–1** demonstrates how to construct a Spatial operator expression, and then relate it to an existing JGeometry with `SpatialParameters` created using a String.

**Example 112–1 Relating an Expression Using String of Spatial Parameters**

```
SpatialParameters parameters =  
    new SpatialParameters("MASK=ANYINTERACT QUERYTYPE=WINDOW");  
Expression selectionCriteria =  
    SpatialExpressionFactory.relate(expressionBuilder.get("geometry"),  
                                   rectangle,  
                                   parameters);
```

[Example 112–2](#) demonstrates how to relate two expressions with `SpatialParameters` constructed using convenience methods.

**Example 112–2 Relating Two Expressions**

```
SpatialParameters parameters = new SpatialParameters();  
parameters.setQueryType(  
    SpatialParameters.QueryType.WINDOW.setMask(Mask.ANYINTERACT);  
Expression selectionCriteria =  
    SpatialExpressionFactory.relate(expressionBuilder1.get("geometry"),  
                                   expressionBuilder2.get("geometry"),  
                                   parameters);
```

# Part XXV

---

## Transactions

This part describes how to use the TopLink unit of work to transactionally perform create, read, update, and delete operations with and without an external transaction processor. It contains the following chapters:

- [Chapter 113, "Introduction to TopLink Transactions"](#)

This chapter describes how to use the unit of work, the TopLink wrapper for a transaction, and how TopLink integrates with transaction management and other important query concepts.
- [Chapter 114, "Using Basic Unit of Work API"](#)

This chapter explains how to use basic TopLink unit of work options.
- [Chapter 115, "Using Advanced Unit of Work API"](#)

This chapter explains how to use advanced TopLink unit of work options.



---

---

## Introduction to TopLink Transactions

This chapter explains how transactions are implemented in TopLink.

This chapter includes the following sections:

- [Unit of Work Architecture](#)
- [Unit of Work Concepts](#)
- [Unit of Work API](#)
- [Example Model Object and Schema](#)

### 113.1 Unit of Work Architecture

A database **transaction** is a set of operations (create, read, update, or delete) that either succeed or fail as a single operation. The database discards, or *rolls back*, unsuccessful transactions, leaving the database in its original state. Transactions may be *internal* (that is, provided by TopLink) or *external* (that is, provided by a source external to the application, such as an application server).

In TopLink, transactions are contained in the unit of work object. You acquire a unit of work from a session (see [Section 114.1, "Acquiring a Unit of Work"](#)) and using its API, you can control transactions directly or through a Java 2 Enterprise Edition (Java EE) application server transaction controller such as the Java Transaction API (JTA).

Transactions execute in their own context, or logical space, isolated from other transactions and database operations.

The transaction context is demarcated; that is, it has a defined structure that includes the following:

- A **begin** point, where the operations within the transaction begin. At this point, the transaction begins to execute its operations.
- A **commit** point, where the operations are complete and the transaction attempts to formalize changes on the database.

The degree to which concurrent (parallel) transactions on the same data are allowed to interact is determined by the level of *transaction isolation* configured. ANSI/SQL defines four levels of database transaction isolation, as shown in [Table 113–1](#). Each offers a trade-off between performance and resistance from the following unwanted actions:

- **Dirty read**: a transaction reads uncommitted data written by a concurrent transaction.
- **Nonrepeatable read**: a transaction rereads data and finds it has been modified by some other transaction that was committed after the initial read operation.

- Phantom read: a transaction reexecutes a query and the returned data has changed due to some other transaction that was committed after the initial read operation.

**Table 113–1 Transaction Isolation Levels**

Transaction Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read
Read Uncommitted	Yes	Yes	Yes
Read Committed	No	Yes	Yes
Repeatable Read	No	No	Yes
Serializable	No	No	No

As a transaction is committed, the database maintains a log of all changes to the data. If all operations in the transaction succeed, the database allows the changes; if any part of the transaction fails, the database uses the log to roll back the changes.

Like any transaction, a unit of work transaction provides the following:

- [Unit of Work Transaction Context](#)
- [Unit of Work Transaction Demarcation](#)
- [Unit of Work Transaction Isolation](#)

### 113.1.1 Unit of Work Transaction Context

Unit of work operations occur within a unit of work *context*, in which writes are isolated from the database until commit time. The unit of work executes changes on copies, or *clones*, of objects in its own internal cache, and if successful, applies changes to objects in the database and the session cache.

### 113.1.2 Unit of Work Transaction Demarcation

In a standalone TopLink application, your application demarcates transactions using the unit of work.

If your application includes a Java EE container that provides container-managed transactions, your application server demarcates transactions using its own transaction service. You configure TopLink to integrate with the container's transaction service by specifying a TopLink external transaction controller.

A TopLink external transaction controller class integrates the unit of work with an application server's transaction service. Using an external transaction controller, your application can participate in transactions that span multiple data sources and that are managed by the application server. The external transaction controller coordinates messages and callbacks between the application server's transaction service and the unit of work.

When you configure your application to use an external transaction controller (see [Section 89.9, "Configuring the Server Platform"](#)), the unit of work executes as part of an external transaction. The unit of work still manages its own internal operations, but it waits for the external transaction to tell it to write changes back to the database and to the session cache.

Note that because the transaction happens outside of the unit of work context and is controlled by the application server's transaction service, errors can be more difficult to diagnose and fix because exceptions may occur outside of your application code, for example, during application server initiated call-backs.

You can integrate the unit of work with the following:

- [JTA Controlled Transactions](#)
- [OTS Controlled Transactions](#)
- [CMP-Controlled Transactions](#)

#### 113.1.2.1 JTA Controlled Transactions

The Java Transaction API (JTA) is the application programming interface you use to interact with a transaction manager.

Using JTA, your application can participate in a distributed transaction. A transaction manager that implements JTA provides transaction management and connection pooling and enables your application to interact with multiple data sources transparently by using JTA.

For more information, see [Section 115.13, "Integrating the Unit of Work with an External Transaction Service"](#).

#### 113.1.2.2 OTS Controlled Transactions

The CORBA Object Transaction Service (OTS) specification is part of the Common Object Request Brokers Architecture (CORBA) Object Services model and is the standard for Object Request Broker (ORB) implementations. Some servers implement the Java APIs for the OTS rather than for JTA (see [Section 113.1.2.1, "JTA Controlled Transactions"](#)).

Use TopLink OTS support with the unit of work to directly access the Java OTS interfaces of servers that do not support JTA.

To integrate your application with an OTS transaction service, you must configure your application to use a custom server platform (see [Section 89.9, "Configuring the Server Platform"](#)).

For more information, see [Section 115.13, "Integrating the Unit of Work with an External Transaction Service"](#).

#### 113.1.2.3 CMP-Controlled Transactions

Entity beans that use container-managed persistence may participate in transactions that are either client demarcated or container demarcated.

A client demarcated transaction occurs when a client of an entity bean directly sets up transaction boundaries using the `javax.transaction.UserTransaction` interface.

A container demarcated transaction occurs when the container automatically wraps an invocation on an EJB in a transaction based upon the transaction attributes supplied in the EJB deployment descriptor.

In transactions involving EJB, TopLink waits until the transaction begins its two-phase commit process before updating the database. This allows for the following:

- SQL optimizations that ensure only changed data is written to the data source
- Proper ordering of updates to allow for database constraints

For more information, see [Section 115.14, "Integrating the Unit of Work with CMP"](#).

### 113.1.3 Unit of Work Transaction Isolation

The unit of work does not directly participate in database transaction isolation. Because the unit of work may execute queries outside the database transaction (and, by interacting with the cache, outside the database itself), the database does not have control over this data and its visibility.

However, by default, TopLink provides a degree of transaction isolation regardless of what database transaction isolation has been configured on the underlying database.

Each unit of work instance operates on its own copy (clone) of registered objects (see [Section 113.2.4, "Clones and the Unit of Work"](#)). In this case, because the unit of work provides an API that allows querying to be done on object changes within a unit of work (see [Section 115.4, "Using Conforming Queries and Descriptors"](#)), the unit of work provides read committed operations.

Optimistic locking, optimistic read locking, or pessimistic locking can be used to further manage concurrency (see [Section 113.3.1.2, "Locking and the Unit of Work"](#)).

Changes are committed to the database only when the unit of work `commit` method is called (either directly or by way of an external transaction controller).

For detailed information on configuring and using TopLink to achieve a particular level of transaction isolation and transaction isolation level limitations, see [Section 115.15, "Database Transaction Isolation Levels"](#).

## 113.2 Unit of Work Concepts

This section introduces transaction concepts unique to TopLink, including the following:

- [Unit of Work Benefits](#)
- [Unit of Work Life Cycle](#)
- [Unit of Work and Change Policy](#)
- [Clones and the Unit of Work](#)
- [Nested and Parallel Units of Work](#)
- [Commit and Rollback Transactions](#)
- [Primary Keys](#)
- [Unit of Work Optimization](#)

### 113.2.1 Unit of Work Benefits

The TopLink unit of work simplifies transactions and improves transactional performance. It is the preferred method of writing to a database in TopLink because it performs the following:

- Sends a minimal amount of SQL to the database during the commit by updating only the exact changes down to the field level
- Reduces database traffic by isolating transaction operations in their own memory space
- Optimizes cache coordination, in applications that use multiple caches, by passing change sets (rather than objects) between caches
- Isolates object modifications in their own transaction space to allow parallel transactions on the same objects



- Ensures referential integrity and minimizes deadlocks by automatically maintaining SQL ordering
- Orders database insert, update, and delete operations to maintain referential integrity for mapped objects
- Resolves bidirectional references automatically
- Frees the application from tracking or recording its changes
- Simplifies persistence with *persistence by reachability* (see [Section 114.5, "Associating a New Source to an Existing Target Object"](#))

## 113.2.2 Unit of Work Life Cycle

TopLink uses the unit of work as follows:

1. The client application acquires a unit of work from a session object.
2. The client application queries TopLink to obtain a cache object it wants to modify, and then registers the cache object with the unit of work.
3. The unit of work registers the object according to the object's change policy. For more information about how change policy affects registration, see [Section 113.2.3, "Unit of Work and Change Policy"](#).

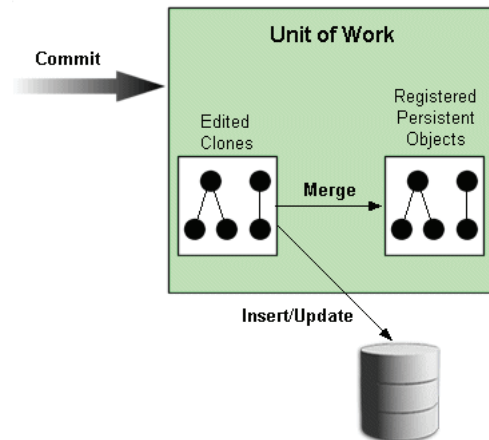
By default, as each object is registered, the unit of work accesses the object from the session cache or database, and creates a backup clone and a working clone (see [Section 113.2.4, "Clones and the Unit of Work"](#)). The unit of work returns the working clone to the client application. If change tracking is used, the unit of work does not create backup clones and intercepts the changes through weaving (see [Section 12.14, "Optimizing Using Weaving"](#)).

4. The client application modifies the working object returned by the unit of work.
5. The client application (or external transaction controller) commits the transaction.
6. The unit of work calculates the change set for each registered object according to the object's change policy. For more information about how change policy affects change set calculation, see [Section 113.2.3, "Unit of Work and Change Policy"](#).

By default, at commit time, the unit of work compares the working clones to the backup clones and calculates the change set (that is, determines the minimum changes required). The comparison is done with a backup clone so that concurrent changes to the same objects will not result in incorrect changes being identified. The unit of work then attempts to commit any new or changed objects to the database.

If the commit transaction succeeds, the unit of work merges changes into the shared session cache. Otherwise, no changes are made to the objects in the shared cache. For more details, see [Section 113.2.6, "Commit and Rollback Transactions"](#).

If there are no changes, the unit of work does not start a new transaction.

**Figure 113–1 The Life Cycle of a Unit of Work**

Example 113–1 shows the default life cycle in code.

#### Example 113–1 Unit of Work Life Cycle

```
// The application reads a set of objects from the database
List employees = session.readAllObjects(Employee.class);

// The application specifies an employee to edit
...
Employee employee = (Employee) employees.get(index);

try {
    // Acquire a unit of work from the session
    UnitOfWork uow = session.acquireUnitOfWork();
    // Register the object that is to be changed. Unit of work returns a clone
    // of the object and makes a backup copy of the original employee
    Employee employeeClone = (Employee)uow.registerObject(employee);
    // Make changes to the employee clone by adding a new phoneNumber.
    // If a new object is referred to by a clone, it does not have to be
    // registered. Unit of work determines it is a new object at commit time
    PhoneNumber newPhoneNumber = new PhoneNumber("cell", "212", "765-9002");
    employeeClone.addPhoneNumber(newPhoneNumber);
    // Commit the transaction: unit of work compares the employeeClone with
    // the backup copy of the employee, begins a transaction, and updates the
    // database with the changes. If successful, the transaction is committed
    // and the changes in employeeClone are merged into employee. If there is an
    // error updating the database, the transaction is rolled back and the
    // changes are not merged into the original employee object
    uow.commit();
}
catch (DatabaseException ex) {
    // If the commit fails, the database is not changed. The unit of work should
    // be thrown away and application-specific action taken
}
// After the commit, the unit of work is no longer valid. Do not use further
```

### 113.2.3 Unit of Work and Change Policy

The unit of work tracks changes for a registered object based on the change policy you configure for the object's descriptor. If there are no changes, the unit of work will not start a database transaction.

Table 113–2 lists the change policies that TopLink provides.

**Table 113–2 TopLink Change Policies**

Change Policy	Applicable to...
Deferred Change Detection Policy	Wide range of object change characteristics. The default change policy.
Object-Level Change Tracking Policy	Objects with few attributes or with many attributes and many changed attributes.
Attribute Change Tracking Policy	Objects with many attributes and few changed attributes. The most efficient change policy. The default change policy for JPA or EJB 2.n CMP on OC4J.

For CMP applications deployed to an application server for which TopLink provides CMP integration (see [Section 8.1, "Introduction to the Application Server Support"](#)), when you configure a descriptor for an entity bean with container-managed persistence with an `ObjectChangeTrackingPolicy`, TopLink code generates a concrete subclass to implement the TopLink `ChangeTracker` interface at deploy time (see [Section 119.30.1.2, "Configuring Object Change Tracking Policy"](#)).

For more information, see [Section 119.30, "Configuring Change Policy"](#).

### 113.2.3.1 Deferred Change Detection Policy

The `DeferredChangeDetectionPolicy` is the change policy that all persistent objects use by default.

This option provides good unit of work commit performance for a wide range of object change characteristics.

When you register in a unit of work an object whose descriptor is configured with a `DeferredChangeDetectionPolicy` (see [Section 119.30.1.1, "Configuring Deferred Change Detection Policy"](#)), a backup clone is made of the object (see [Section 113.2.4, "Clones and the Unit of Work"](#)) and at commit time, the unit of work computes changes by making an attribute-by-attribute comparison between the backup clone and the original object.

This change policy is applicable to all mapping types.

### 113.2.3.2 Object-Level Change Tracking Policy

The `ObjectChangeTrackingPolicy` optimizes the unit of work commit transaction by including objects in the change set calculation only if at least one attribute has changed.

This option provides improved unit of work commit performance for objects with few attributes, or with many attributes and many changed attributes.

When you register in a unit of work an object whose descriptor is configured with `ObjectChangeTracking` change policy, a backup clone is made of the object and at commit time, the unit of work computes changes by comparing the backup to the current object if and only if at least one attribute is changed (if the object's `hasChanges` method returns `true`). If a registered object has no changes, the unit of work does not compare it to the backup clone.

---

**Note:** If you modify an object's field through reflection, TopLink will not detect the change. However, if you disable change tracking, TopLink *will* detect the change.

---

This change policy is applicable to a subset of mapping types (see [Section 113.2.3.4, "Change Policy Mapping Support"](#)).

TopLink provides different levels of support for this change policy depending on the EJB version and application server you are using:

**113.2.3.2.1 EJB CMP and JPA** For CMP applications deployed to an application server for which TopLink provides CMP integration (see [Section 8.1, "Introduction to the Application Server Support"](#)), as well as for JPA applications, when you configure a descriptor for an entity bean with container-managed persistence (or a JPA entity) with an `ObjectChangeTrackingPolicy`, TopLink code generates a concrete subclass to implement the TopLink `ChangeTracker` interface at deploy time (see [Section 119.30.1.2, "Configuring Object Change Tracking Policy"](#)).

### 113.2.3.3 Attribute Change Tracking Policy

The `AttributeChangeTrackingPolicy` optimizes the unit of work commit transaction by tracking all object changes at the attribute level. This eliminates two unit of work operations: backup clone creation and change detection through comparison.

This option provides improved unit of work commit performance for objects with many attributes, and few changed attributes. Generally, this is the most efficient change policy.

This change policy is applicable to a subset of mapping types (see [Section 113.2.3.4, "Change Policy Mapping Support"](#)).

---

---

**Note:** You cannot use the `AttributeChangeTrackingPolicy` if you are using any instance of `FieldsLockingPolicy` (see [Section 16.4.4, "Optimistic Field Locking Policies"](#)).

---

---

TopLink provides different levels of support for this change policy:

**113.2.3.3.1 JPA Entities** For JPA entities, you can configure TopLink to automatically weave attribute level change tracking.

TopLink only supports change tracking with lazy collection relationships, not with eager collection relationship.

For more information, see "What You May Need to Know About Weaving JPA Entities" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#What\\_You\\_May\\_Need\\_to\\_Know\\_About\\_Weaving\\_JPA\\_Entities](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#What_You_May_Need_to_Know_About_Weaving_JPA_Entities).

**113.2.3.3.2 Plain Old Java Object (POJO) Classes** For POJO classes, you can configure TopLink to automatically weave attribute level change tracking.

TopLink can weave both transparent indirect container indirection (lazy loading) and change tracking for collection mappings. If you manually configure a collection mapping with nontransparent indirection (either value holder indirection or proxy indirection), TopLink does not automatically weave change tracking.

For more information, see [Section 2.10.5, "What You May Need to Know About Weaving and POJO Classes"](#).

**113.2.3.3.3 EJB CMP on OC4J** By default, TopLink uses change tracking for CMP.

### 113.2.3.4 Change Policy Mapping Support

TopLink supports alternative change tracking policies (policies other than `DeferredChangeDetectionPolicy`) for attributes that use any of the following mapping types:

- [Direct-to-Field Mapping](#)
- [Transformation Mapping](#) (immutable mappings only)
- [One-to-One Mapping](#)
- [Variable One-to-One Mapping](#)
- [One-to-Many Mapping](#)
- [Many-to-Many Mapping](#)
- [Direct Collection Mapping](#)
- [Direct Map Mapping](#)
- [Aggregate Object Mapping](#)
- [EIS Transformation Mapping](#) (immutable mappings only)

TopLink uses the `DeferredChangeDetectionPolicy` (see [Section 113.2.3.1, "Deferred Change Detection Policy"](#)) for attributes that use any other type of mapping.

If a transformation mapping maps a mutable value, TopLink must clone and compare the value in a unit of work (see [Section 119.29, "Configuring Copy Policy"](#)).

By default, TopLink assumes that all transformation mappings are mutable. If the mapping maps a simple immutable value, you can improve unit of work performance by configuring the `IsMutable` option to `false`.

Mutable basic mappings affect the overhead of change tracking. TopLink can only weave an attribute change tracking policy for immutable mappings.

For more information, see [Section 2.8.11, "Mutability"](#).

## 113.2.4 Clones and the Unit of Work

When using the `DeferredChangeDetectionPolicy` or the `ObjectLevelChangeTrackingPolicy` (see [Section 113.2.3.1, "Deferred Change Detection Policy"](#)), the unit of work maintains the following two copies of the original objects registered with it:

- working clones;
- backup clones.

After you change the working clones and the transaction is committed, the unit of work compares the working clones to the backup clones, and writes any changes to the database. The unit of work uses clones to allow parallel units of work (see [Section 113.2.5, "Nested and Parallel Units of Work"](#)) to exist, a requirement in multiuser three-tier applications.

The TopLink cloning process is efficient in that it clones only the mapped attributes of registered objects, and stops at indirection (lazily loaded) objects unless you trigger the indirection. For more information, see [Section 121.3, "Configuring Indirection \(Lazy Loading\)"](#).

You can customize the cloning process using the descriptor's copy policy. For more information, see [Section 119.29, "Configuring Copy Policy"](#).

You should discontinue the use of the unit of work clones after the transaction has been committed, as it is beyond the scope of a server request. If you choose to continue using the clones, be aware that these objects may include a reference to the unit of work and not let the garbage collection to proceed until they are released. For more information, see [Section 115.6, "Resuming a Unit of Work After Commit"](#).

## 113.2.5 Nested and Parallel Units of Work

You can use TopLink to create the following:

- [Nested Unit of Work](#)
- [Parallel Unit of Work](#)

For additional information and examples on using nested and parallel units of work, see [Section 115.8, "Using a Nested or Parallel Unit of Work"](#).

### 113.2.5.1 Nested Unit of Work

You can nest a unit of work (the *child*) within another unit of work (the *parent*). A nested unit of work does not commit changes to the database. Instead, it passes its changes to the parent unit of work, and the parent attempts to commit the changes at commit time. Nesting units of work lets you break a large transaction into smaller isolated transactions, and ensures that:

- Changes from each nested unit of work commit or fail as a group.
- Failure of a nested unit of work does not affect the commit or rollback transaction of other operations in the parent unit of work.
- Changes are presented to the database as a single transaction.

### 113.2.5.2 Parallel Unit of Work

You can modify the same objects in multiple unit of work instances in parallel because the unit of work manipulates copies of objects. TopLink resolves any concurrency issues when the Units of Work commits the changes.

## 113.2.6 Commit and Rollback Transactions

When a unit of work transaction is committed, it either succeeds, or fails and rolls back. A commit transaction can be initiated by your application or by a Java EE container.

### 113.2.6.1 Commit Transactions

At commit time, the unit of work compares the working clones and backup clones to calculate the change set (that is, to determine the minimum changes required). Changes include updates to or deletion of existing objects, and the creation of new objects. The unit of work then begins a database transaction, and attempts to write the changes to the database. If all changes commit successfully on the database, the unit of work merges the changed objects into the session cache. If any one of the changes fail on the database, the unit of work rolls back any changes on the database, and does not merge changes into the session cache.

The unit of work calculates commit order using foreign key information from one-to-one and one-to-many mappings. If you encounter constraint problems during a commit transaction, verify your mapping definitions. The order in which you register objects with the `registerObject` method does not affect the commit order.

**113.2.6.1.1 Commit and JTA** When your application uses JTA, the unit of work commit transaction acts differently than in a non-JTA application. In most cases, the unit of work attaches itself to an external transaction. If no transaction exists, the unit of work creates a transaction. This distinction affects commit activity as follows:

- *If the unit of work attaches to an existing transaction*, the unit of work ignores the `commit` call. The transaction commits the unit of work when the entire external transaction is complete.
- *If the unit of work starts the external transaction*, the transaction treats the unit of work `commit` call as a request to commit the external transaction. The external transaction then calls its own commit code on the database.

In either case, only the external transaction can call `commit` on the database because it owns the database connection.

For more information, see [Section 115.13, "Integrating the Unit of Work with an External Transaction Service"](#).

### 113.2.6.2 Rollback Transactions

A unit of work commit transaction must succeed or fail as a unit. Failure in writing changes to the database causes the unit of work to roll back the database to its previous state. Nothing changes in the database, and the unit of work does not merge changes into the session cache.

**113.2.6.2.1 Rollback and JTA** In a JTA environment, the unit of work does not own the database connection. In this case, the unit of work sends the rollback call to the external transaction rather than the database, and the external transaction treats the rollback call as a request to roll the transaction back.

For more information, see [Section 115.13, "Integrating the Unit of Work with an External Transaction Service"](#).

## 113.2.7 Primary Keys

You cannot modify the primary key attribute of an object in a unit of work. This is an unsupported operation and doing so will result in unexpected behaviour (exceptions or database corruption).

To replace one instance of an object with unique constraints with another, see [Section 115.10.1, "How to Use the `setShouldPerformDeletesFirst` Method of the Unit of Work"](#).

## 113.2.8 Unit of Work Optimization

By default, the unit of work performs change set calculation efficiently for a wide range of object change characteristics. However, there are various ways you can use the unit of work to enhance application performance.

One way to improve performance is to consider using an alternative change policy (see [Section 113.2.3, "Unit of Work and Change Policy"](#)).

For more performance options, see [Section 12.13, "Optimizing the Unit of Work"](#).

## 113.3 Unit of Work API

You do not instantiate an instance of `oracle.toplink.sessions.UnitOfWork`. Rather, you acquire a unit of work from an instance of `oracle.toplink.sessions.Session` or from another unit of work.

For more information on creating sessions, see [Chapter 88, "Creating a Session"](#).

For more information on acquiring a unit of work, see [Section 114.1, "Acquiring a Unit of Work"](#).

For more information on using the basic API of the unit of work, see [Chapter 114, "Using Basic Unit of Work API"](#).

For more information on using the advanced API of the unit of work, see [Chapter 115, "Using Advanced Unit of Work API"](#).

### 113.3.1 Unit of Work as Session

The unit of work extends the interface `oracle.toplink.sessions.Session`, and implements all the usual session API. When using session API from a unit of work, you should consider the following:

- [Reading and Querying Objects with the Unit of Work](#)
- [Locking and the Unit of Work](#)

#### 113.3.1.1 Reading and Querying Objects with the Unit of Work

A unit of work offers the same set of database access methods as a regular session.

When called from a unit of work, these methods access the objects in the unit of work, register the selected objects automatically, and return clones.

Although this makes it unnecessary for you to call the `registerObject` and `registerAllObjects` methods, be aware of the restrictions on registering objects described in [Section 114.2, "Creating an Object"](#) and [Section 114.5, "Associating a New Source to an Existing Target Object"](#).

**113.3.1.1.1 Reading Objects with the Unit of Work** As with regular sessions, you use the `readObject` and `readAllObjects` methods to read objects from the database.

**113.3.1.1.2 Querying Objects with the Unit of Work** You can execute queries in a unit of work with the `executeQuery` method.

---

---

**Note:** Because a unit of work manages changes to existing objects and the creation of new objects, modifying queries such as `InsertObjectQuery` or `UpdateObjectQuery` are not necessary and therefore are not supported by the unit of work.

---

---

#### 113.3.1.2 Locking and the Unit of Work

For information on locking API generic to all sessions, see the following:

- [Section 2.3.5, "Locking"](#)
- [Section 119.26, "Configuring Locking Policy"](#)

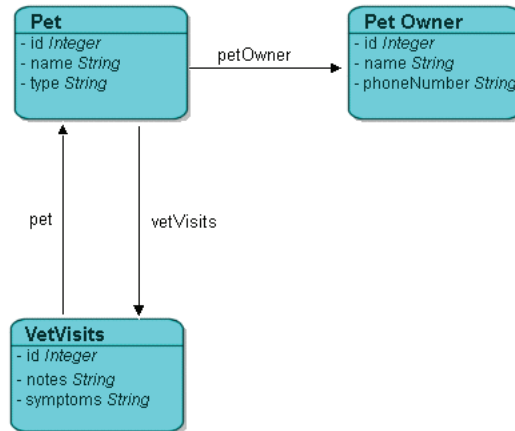
For information on locking API specific to a unit of work, see [Section 115.11, "Using Optimistic Read Locking with the `forceUpdateToVersionField` Method"](#).



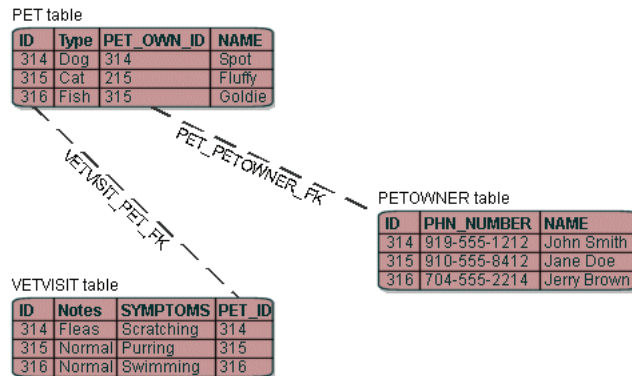
## 113.4 Example Model Object and Schema

Throughout the chapters in this part, the following object model and schema are used in the examples provided. The example object model appears in [Figure 113–2](#) and the example entity-relationship (data model) diagram appears in [Figure 113–3](#).

**Figure 113–2 Example Object Model**



**Figure 113–3 Example Data Model**





---

---

## Using Basic Unit of Work API

This chapter explains the essential unit of work API calls that you are most likely to use throughout the development cycle.

This chapter includes the following sections:

- [Acquiring a Unit of Work](#)
- [Creating an Object](#)
- [Modifying an Object](#)
- [Associating a New Target to an Existing Source Object](#)
- [Associating a New Source to an Existing Target Object](#)
- [Associating an Existing Source to an Existing Target Object](#)
- [Deleting Objects](#)

For more information, see [Chapter 115, "Using Advanced Unit of Work API"](#).

### 114.1 Acquiring a Unit of Work

This example shows how to acquire a unit of work from a client session object.

```
Server server =
    (Server) SessionManager.getManager().getSession(
        sessionName, MyServerSession.class.getClassLoader());
Session session = (Session) server.acquireClientSession();
UnitOfWork uow = session.acquireUnitOfWork();
```

You can acquire a unit of work from any session type. For more information about acquiring sessions at run time, see [Section 87.2.4, "Acquiring a Session at Run Time with the Session Manager"](#).

Note that you do not need to create a new session and log in before every transaction. The recommended pattern is to acquire a client session per client access (or thread), and then acquire the necessary unit of work from this client session.

The unit of work is valid until the `commit` or `release` method is called. After a `commit` or `release` transaction, a unit of work is not valid even if the transaction fails and is rolled back.

A unit of work remains valid after the `commitAndResume` method is called as described in [Section 115.6, "Resuming a Unit of Work After Commit"](#).

When using a unit of work with JTA, you should also use the advanced API `getActiveUnitOfWork` method as described in [Section 115.13, "Integrating the Unit of Work with an External Transaction Service"](#).

## 114.2 Creating an Object

When you create new objects in the unit of work, use the `registerObject` method to ensure that the unit of work writes the objects to the database at commit time.

The unit of work calculates commit order using foreign key information from one-to-one and one-to-many mappings. If you encounter constraint problems during a commit transaction, verify your mapping definitions. The order in which you register objects with the `registerObject` method does not affect the commit order.

[Example 114–1](#) and [Example 114–2](#) show how to create and persist a simple object (without relationships) using the clone returned by the unit of work `registerObject` method.

### **Example 114–1 Creating an Object: Preferred Method**

```
UnitOfWork uow = session.acquireUnitOfWork();
    Pet pet = new Pet();
    Pet petClone = (Pet)uow.registerObject(pet);
    petClone.setId(100);
    petClone.setName("Fluffy");
    petClone.setType("Cat");
uow.commit();
```

[Example 114–2](#) shows a common alternative.

### **Example 114–2 Creating an Object: Alternative Method**

```
UnitOfWork uow = session.acquireUnitOfWork();
    Pet pet = new Pet();
    pet.setId(100);
    pet.setName("Fluffy");
    pet.setType("Cat");
    uow.registerObject(pet);
uow.commit();
```

Both approaches produce the following SQL:

```
INSERT INTO PET (ID, NAME, TYPE, PET_OWN_ID) VALUES (100, 'Fluffy', 'Cat', NULL)
```

[Example 114–1](#) is preferred: it gets you into the pattern of working with clones and provides the most flexibility for future code changes. Working with combinations of new objects and clones can lead to confusion and unwanted results.

## 114.3 Modifying an Object

In [Example 114–3](#), a `Pet` is read prior to a unit of work: the variable `pet` is the cache copy clone for that `Pet`. Inside the unit of work, register the cache copy to get a working copy clone. You then modify the working copy clone and commit the unit of work.

### **Example 114–3 Modifying an Object**

```
// Read in any pet
Pet pet = (Pet)session.readObject(Pet.class);
UnitOfWork uow = session.acquireUnitOfWork();
Pet petClone = (Pet) uow.registerObject(pet);
petClone.setName("Furry");
uow.commit();
```

[Example 114–4](#) shows how to take advantage of the fact that you can query through a unit of work and get back clones, saving the registration step. However, the drawback is that we do not have a handle to the cache copy clone.

If you wanted to do something with the updated `Pet` after the commit transaction, you would have to query the session to get it (remember that after a unit of work is committed, its clones are invalid and should not be used).

**Example 114–4 Modifying an Object: Skipping the Registration Step**

```
UnitOfWork uow = session.acquireUnitOfWork();
Pet petClone = (Pet) uow.readObject(Pet.class);
petClone.setName("Furry");
uow.commit();
```

Both approaches produce the following SQL:

```
UPDATE PET SET NAME = 'Furry' WHERE (ID = 100)
```

Take care when querying through a unit of work. All objects read in the query are registered in the unit of work and therefore will be checked for changes at commit time. Rather than do a `ReadAllQuery` through a unit of work, it is better for performance to design your application to do the `ReadAllQuery` through a session, and then register in a unit of work only the objects that need to be changed.

## 114.4 Associating a New Target to an Existing Source Object

This section explains how you can associate a new target to an existing source object, including the following:

- [How to Associate a New Target to an Existing Source Object in a Unidirectional Relationship: Reference to the New Cache Object After Commit not Required](#)
- [How to Associate a New Target to an Existing Source Object in a Unidirectional Relationship: Reference to the New Cache Object After Commit Required](#)
- [How to Associate a New Target to an Existing Source Object in a Bidirectional Relationship: Query for Target Before Commit not Required](#)
- [How to Associate a New Target to an Existing Source Object in a Bidirectional Relationship: Query for Target Object Before Commit Required](#)

Deciding which approach to use depends on whether or not your code requires a reference to the cache copy clone of the new object after the unit of work is committed, and on how adaptable to change you want your code to be.

---

**Note:** You cannot use `UnitOfWork` methods `registerObject`, `registerNewObject`, or `registerExistingObject` with an aggregate object (see [Section 22.2.1.2, "Creating Relational Aggregate Descriptors"](#)). Doing so will raise a `ValidationException` or other errors at commit time. For more information, see [Section 115.1.4, "How to Work with Aggregates"](#).

---

### 114.4.1 How to Associate a New Target to an Existing Source Object in a Unidirectional Relationship: Reference to the New Cache Object After Commit not Required

[Example 114–5](#) shows how to associate a new target with an existing source in a unidirectional relationship without retaining a reference to the cache object.

When the `Pet` object is read using the unit of work, `TopLink` automatically registers it. Because there is a unidirectional relationship between the `Pet` object and the new `PetOwner` and `VetVisit` objects, you do not need to register the new `PetOwner` or `VetVisit` objects. `TopLink` can reach these new objects through the registered `Pet` object and automatically detect that they are new objects.

**Example 114–5 Associating Without Reference to the Cache Object**

```
UnitOfWork uow = session.acquireUnitOfWork();
Pet petClone = (Pet)uow.readObject(Pet.class);

PetOwner petOwner = new PetOwner();
petOwner.setId(400);
petOwner.setName("Donald Smith");
petOwner.setPhoneNumber("555-1212");

VetVisit vetVisit = new VetVisit();
vetVisit.setId(500);
vetVisit.setNotes("Pet was shedding a lot.");
vetVisit.setSymptoms("Pet in good health.");
vetVisit.setPet(petClone);

petClone.setPetOwner(petOwner);
petClone.getVetVisits().add(vetVisit);
uow.commit();
```

This executes the following proper SQL:

```
INSERT INTO PETOWNER (ID, NAME, PHN_NBR) VALUES (400, 'Donald Smith', '555-1212')
UPDATE PET SET PET_OWN_ID = 400 WHERE (ID = 100)
INSERT INTO VETVISIT (ID, NOTES, SYMPTOMS, PET_ID) VALUES (500, 'Pet was shedding a lot.',
'Pet in good health.', 100)
```

When associating new objects to existing objects, the unit of work treats the new object as if it were a clone. That is, after the commit transaction:

```
petOwner != session.readObject(petOwner)
```

Therefore, after the unit of work commit transaction, the variables `vetVisit` and `petOwner` no longer point to their respective cache objects; they point at working copy clones.

If you need the cache object after the unit of work commit transaction, you must query for it or create the association with a reference to the cache object (as described in [Section 114.4.2, "How to Associate a New Target to an Existing Source Object in a Unidirectional Relationship: Reference to the New Cache Object After Commit Required"](#)).

If there was a bidirectional relationship between the source and target objects, you must take more care when registering them (see [Section 114.4.3, "How to Associate a New Target to an Existing Source Object in a Bidirectional Relationship: Query for Target Before Commit not Required"](#)).

For more information, see [Section 115.1, "Registering and Unregistering Objects"](#).

---

**Note:** You cannot use `UnitOfWork` methods `registerObject`, `registerNewObject`, or `registerExistingObject` with an aggregate object (see [Section 22.2.1.2, "Creating Relational Aggregate Descriptors"](#)). Doing so will raise a `ValidationException` or other errors at commit time. For more information, see [Section 115.1.4, "How to Work with Aggregates"](#).

---

## 114.4.2 How to Associate a New Target to an Existing Source Object in a Unidirectional Relationship: Reference to the New Cache Object After Commit Required

[Example 114–6](#) shows how to associate a new target with an existing source in a unidirectional relationship and retain a reference to the cache object.

When the `Pet` object is read using the unit of work, `TopLink` automatically registers it. Because there is a unidirectional relationship between the `Pet` object and the new `PetOwner` and `VetVisit` objects, you do not need to register the new `PetOwner` or

VetVisit objects. TopLink can reach these new objects through the registered Pet object and automatically detect that they are new objects.

However, by using UnitOfWork method registerObject, you can retain a handle to the post-commit cache objects in case your code needs to continue using them after commit: for example, to display their new contents in a GUI.

If there was a bidirectional relationship between the source and target objects, you must take more care when registering them (see [Section 114.4.4, "How to Associate a New Target to an Existing Source Object in a Bidirectional Relationship: Query for Target Object Before Commit Required"](#)).

#### **Example 114–6 Associating With Reference to the Cache Object**

```
UnitOfWork uow = session.acquireUnitOfWork();
Pet petClone = (Pet)uow.readObject(Pet.class);

PetOwner petOwner = new PetOwner();
PetOwner petOwnerClone = (PetOwner)uow.registerObject(petOwner);
petOwnerClone.setId(400);
petOwnerClone.setName("Donald Smith");
petOwnerClone.setPhoneNumber("555-1212");

VetVisit vetVisit = new VetVisit();
VetVisit vetVisitClone = (VetVisit)uow.registerObject(vetVisit);
vetVisitClone.setId(500);
vetVisitClone.setNotes("Pet was shedding a lot.");
vetVisitClone.setSymptoms("Pet in good health.");
vetVisitClone.setPet(petClone);

petClone.setPetOwner(petOwnerClone);
petClone.getVetVisits().add(vetVisitClone);
uow.commit();
```

Now, after the unit of work commit transaction:

```
petOwner == session.readObject(petOwner)
```

This means that we have a handle to the cache copy after the commit transaction, rather than a clone.

For more information, see [Section 115.1, "Registering and Unregistering Objects"](#).

---

**Note:** You cannot use UnitOfWork methods registerObject, registerNewObject, or registerExistingObject with an aggregate object (see [Section 22.2.1.2, "Creating Relational Aggregate Descriptors"](#)). Doing so will raise a ValidationException or other errors at commit time. For more information, see [Section 115.1.4, "How to Work with Aggregates"](#).

---

### **114.4.3 How to Associate a New Target to an Existing Source Object in a Bidirectional Relationship: Query for Target Before Commit not Required**

Consider an Employee class implemented, as [Example 114–7](#) shows. Note that the setManager method modifies the Employee instance you pass into it.

#### **Example 114–7 Employee Class**

```
public class Employee {

    private Collection managedEmployees = new ArrayList();
    private Employee myManager;
```

```

...

public setManager(Employee manager) {
    myManager = manager;
    manager.addManagedEmployee(this);
}

public addManagedEmployee(Employee employee) {
    managedEmployees.add(employee);
}

...
}

```

[Example 114–8](#) shows how to register a new object when a bidirectional relationship exists such as that between manager and employee.

Because `Employee` method `setManager` modifies the `Employee` you pass in (as [Example 114–7](#) shows), you must pass in `managerClone` that `registerObject` returns.

After you call `setManager`, you establish the bidirectional relationship between `newEmployee` and `managerClone`. Because `newEmployee` is reachable from the `manager` object already registered with the unit of work, `TopLink` can automatically detect that it is a new object. Consequently, you do not need to register `newEmployee` at all and it is, in fact, an error to call `registerObject` on `newEmployee` in this case.

If your code must be able to query for the new child object prior to commit, see [Section 114.4.4, "How to Associate a New Target to an Existing Source Object in a Bidirectional Relationship: Query for Target Object Before Commit Required"](#).

If you need the cache object after the unit of work commit transaction, in this case, you must query for it.

#### **Example 114–8 Resolving Issues When Adding New Objects**

```

// Get an employee read from the parent session of the unit of work
Employee manager = (Employee)session.readObject(Employee.class);

// Acquire a unit of work
UnitOfWork uow = session.acquireUnitOfWork();

// Register the manager to get its clone
Employee managerClone = (Employee)uow.registerObject(manager);

// Create a new employee
Employee newEmployee = new Employee();
newEmployee.setFirstName("Spike");
newEmployee.setLastName("Robertson");

/* INCORRECT: Do not associate the new employee with the original manager. This will cause a
QueryException when TopLink detects this error during commit */
//newEmployee.setManager(manager);

/* CORRECT: Associate the new object with the clone. Note that in this example, the
setManager method is maintaining the bidirectional managedEmployees relationship and adding
the new employee to its managedEmployees. At commit time, the unit of work will detect that
this is a new object and will take the appropriate action */
newEmployee.setManager(managerClone);

/* INCORRECT: Do not register the newEmployee: this will create two copies and cause a
QueryException when TopLink detects this error during commit */
//uow.registerObject(newEmployee);

// Commit the unit of work

```



```
uow.commit();
```

For more information, see [Section 115.1, "Registering and Unregistering Objects"](#)).

---



---

**Note:** You cannot use `UnitOfWork` methods `registerObject`, `registerNewObject`, or `registerExistingObject` with an aggregate object (see [Section 22.2.1.2, "Creating Relational Aggregate Descriptors"](#)). Doing so will raise a `ValidationException` or other errors at commit time. For more information, see [Section 115.1.4, "How to Work with Aggregates"](#).

---



---

#### 114.4.4 How to Associate a New Target to an Existing Source Object in a Bidirectional Relationship: Query for Target Object Before Commit Required

Consider an `Employee` class implemented, as [Example 114-7](#) shows. Note that the `setManager` method modifies the `Employee` instance you pass into it.

##### **Example 114-9 Employee Class**

```
public class Employee
{
    private Collection managedEmployees = new ArrayList();
    private Employee myManager;

    ...

    public setManager(Employee manager)
    {
        myManager = manager;
        manager.addManagedEmployee(this);
    }

    public addManagedEmployee(Employee employee)
    {
        managedEmployees.add(employee);
    }

    ...
}
```

[Example 114-8](#) shows how to register a new object when a bidirectional relationship exists such as that between manager and employee.

##### **Example 114-10 Resolving Issues When Adding New Objects**

```
// Get an employee read from the parent session of the unit of work
Employee manager = (Employee)session.readObject(Employee.class);

// Acquire a unit of work
UnitOfWork uow = session.acquireUnitOfWork();

// Register the manager to get its clone
Employee managerClone = (Employee)uow.registerObject(manager);

// Create a new employee
Employee newEmployee = new Employee();
newEmployee.setFirstName("Spike");
newEmployee.setLastName("Robertson");

/* INCORRECT: Do not associate the new employee with the original manager. This will cause a
QueryException when TopLink detects this error during commit */
//newEmployee.setManager(manager);
```

```

/* CORRECT: Associate the new object with the clone. Note that in this example, the
setManager method is maintaining the bidirectional managedEmployees relationship and adding
the new employee to its managedEmployees. At commit time, the unit of work will detect that
this is a new object and will take the appropriate action */
newEmployee.setManager(managerClone);

/* INCORRECT: Do not register the newEmployee: this will create two copies and cause a
QueryException when TopLink detects this error during commit */
//uow.registerObject(newEmployee);

/* CORRECT: In the above setManager call, if the managerClone's managedEmployees was not
maintained by the setManager method, then you should call registerObject before the new
employee is related to the manager. If in doubt, you could use the registerNewObject method
to ensure that the newEmployee is registered in the unit of work. The registerNewObject
method registers the object, but does not make a clone */
uow.registerNewObject(newEmployee);

// Commit the unit of work
uow.commit();

```

Because `Employee` method `setManager` modifies the `Employee` you pass in (as [Example 114-7](#) shows), you must pass in `managerClone` that `registerObject` returns.

After you call `setManager`, you establish the bidirectional relationship between `newEmployee` and `managerClone`. Because `newEmployee` is reachable from the `manager` object already registered with the unit of work, `TopLink` can automatically detect that it is a new object. Consequently, you do not need to register `newEmployee` at all and it is, in fact, an error to call `registerObject` on `newEmployee` in this case.

If your code must be able to query for the new child object prior to commit, register the new object using `UnitOfWork` method `registerNewObject`. Unlike `registerObject`, this method does not create a clone.

Another difference between `registerNewObject` and `registerObject` is that `registerNewObject` does not cascade registration to child objects. If you call `registerNewObject` on a parent object, you must also call `registerNewObject` on new child instances if your code must be able to query for the new child object prior to commit and you prefer not to use conforming queries.

If you need the cache object after the unit of work commit transaction, you must query for it.

For more information, see [Section 115.1, "Registering and Unregistering Objects"](#).

---



---

**Note:** You cannot use `UnitOfWork` methods `registerObject`, `registerNewObject`, or `registerExistingObject` with an aggregate object (see [Section 22.2.1.2, "Creating Relational Aggregate Descriptors"](#)). Doing so will raise a `ValidationException` or other errors at commit time. For more information, see [Section 115.1.4, "How to Work with Aggregates"](#).

---



---

## 114.5 Associating a New Source to an Existing Target Object

This section describes how to associate a new source object with an existing target object with one-to-many and one-to-one relationships.

`TopLink` follows all relationships of all registered objects (deeply) in a unit of work to calculate what is new and what has changed. This is known as **persistence by reachability**. In [Section 114.4, "Associating a New Target to an Existing Source Object"](#), we saw that when you associate a new target with an existing source, you can choose

to register the object or not. If you do not register the new object, it is still reachable from the source object (which is a clone, hence it is registered). However, when you need to associate a new source object with an existing target, you must register the new object. If you do not register the new object, then it is not reachable in the unit of work, and TopLink will not write it to the database.

For example, the code shown in [Example 114–11](#) shows how to create a new Pet and associate it with an existing PetOwner.

**Example 114–11 Associating a New Source to an Existing Target Object**

```
UnitOfWork uow = session.acquireUnitOfWork();
PetOwner existingPetOwnerClone =
    (PetOwner)uow.readObject(PetOwner.class);
```

```
Pet newPet = new Pet();
Pet newPetClone = (Pet)uow.registerObject(newPet);
newPetClone.setId(900);
newPetClone.setType("Lizzard");
newPetClone.setName("Larry");
newPetClone.setPetOwner(existingPetOwnerClone);
uow.commit();
```

This generates the following proper SQL:

```
INSERT INTO PET (ID, NAME, TYPE, PET_OWN_ID) VALUES (900, 'Larry', 'Lizzard', 400)
```

In this situation, you should register the new object and work with the working copy of the new object. If you associate the new object with the PetOwner clone without registering, it will not be written to the database.

---

**Note:** You cannot use UnitOfWork methods `registerObject`, `registerNewObject`, or `registerExistingObject` with an aggregate object (see [Section 22.2.1.2, "Creating Relational Aggregate Descriptors"](#)). Doing so will raise a `ValidationException` or other errors at commit time. For more information, see [Section 115.1.4, "How to Work with Aggregates"](#).

---

If you fail to register the clone and accidentally associate the cache version of the existing object with the new object, then TopLink will generate an error which states that you have associated the cache version of an object ("from a parent session") with a clone from this unit of work. You must work with working copies in units of work.

For more information, see the following:

- [Section 114.4, "Associating a New Target to an Existing Source Object"](#)
- [Section 114.6, "Associating an Existing Source to an Existing Target Object"](#)

## 114.6 Associating an Existing Source to an Existing Target Object

This section explains how to associate an existing source object with an existing target object with one-to-many and one-to-one relationships.

As shown in [Example 114–12](#), associating existing objects with each other in a unit of work is as simple as associating objects in Java. Just remember to only work with working copies of the objects.

**Example 114–12 Associating an Existing Source to Existing Target Object**

```
// Associate all VetVisits in the database to a Pet from the database
```

```

UnitOfWork uow = session.acquireUnitOfWork();
Pet existingPetClone = (Pet)uow.readObject(Pet.class);
List allVetVisitClones;
allVetVisitClones = uow.readAllObjects(VetVisit.class);
Iterator iter = allVetVisitClones.elements();
while(iter.hasNext()) {
    VetVisit vetVisitClone = (VetVisit)iter.next();
    existingPetClone.getVetVisits().add(vetVisitClone);
    vetVisitClone.setPet(existingPetClone);
};
uow.commit();

```

The most common error when associating existing objects is failing to work with the working copies. If you accidentally associate a cache version of an object with a working copy you will get an error at commit time indicating that you associated an object from a parent session (the cache version) with a clone from this unit of work.

[Example 114–13](#) shows another example of associating an existing source to an existing target object.

#### **Example 114–13 Associating Existing Objects**

```

// Get an employee read from the parent session of the unit of work
Employee employee = (Employee)session.readObject(Employee.class)

// Acquire a unit of work
UnitOfWork uow = session.acquireUnitOfWork();
Project project = (Project) uow.readObject(Project.class);

/* When associating an existing object (read from the session) with a clone, we
must make sure we register the existing object and assign its clone into a unit of
work */

/* INCORRECT: Cannot associate an existing object with a unit of work clone. A
QueryException will be thrown */
//project.setTeamLeader(employee);

/* CORRECT: Instead register the existing object then associate the clone */
Employee employeeClone = (Employee)uow.registerObject(employee);
project.setTeamLeader(employeeClone);
uow.commit();

```

For more information, see the following:

- [Section 114.4, "Associating a New Target to an Existing Source Object"](#)
- [Section 114.5, "Associating a New Source to an Existing Target Object"](#)

## 114.7 Deleting Objects

To delete objects in a unit of work, use the `deleteObject` or `deleteAllObjects` method. When you delete an object that is not already registered in the unit of work, the unit of work registers the object automatically.

When you delete an object, TopLink deletes the object's privately owned child parts, because those parts cannot exist without the owning (parent) object. At commit time, the unit of work generates SQL to delete the objects, taking database constraints into account.

When you delete an object, you must take your object model into account. You may need to set references to the deleted object to null (for an example, see [Section 114.7.1, "How to Use the `privateOwnedRelationship` Attribute"](#)).

This section explains how to delete objects within a unit of work, including the following:

- [How to Use the `privateOwnedRelationship` Attribute](#)
- [How to Explicitly Delete from the Database](#)
- [What You May Need to Know About the Order in which Objects Are Deleted](#)

### 114.7.1 How to Use the `privateOwnedRelationship` Attribute

Relational databases do not have garbage collection like a Java Virtual Machine (JVM) does. To delete an object in Java you just remove the reference to the object. To delete a row in a relational database, you must explicitly delete it. Rather than tediously manage when to delete data in the relational database, use the mapping attribute `privateOwnedRelationship` to have TopLink manage the garbage collection in the relational database for you.

As shown in [Example 114–14](#), when you create a mapping using Java, use its `privateOwnedRelationship` method to tell TopLink that the referenced object is privately owned: that is, the referenced child object cannot exist without the parent object.

#### **Example 114–14** *Specifying a Mapping as Privately Owned*

```
OneToOneMapping petOwnerMapping = new OneToOneMapping();
petOwnerMapping.setAttributeName("petOwner");
petOwnerMapping.setReferenceClass(com.top.uowprimer.model.PetOwner.class);
petOwnerMapping.privateOwnedRelationship();
petOwnerMapping.addForeignKeyFieldName("PET.PET_OWN_ID", "PETOWNER.ID");
descriptor.addMapping(petOwnerMapping);
```

When you create a mapping using TopLink Workbench, you can select the **Private Owned** check box under the **General** tab.

When you tell TopLink that a relationship is privately owned, you are specifying the following:

- If the source of a privately owned relationship is deleted, then delete the target.
- If you remove the reference to a target from a source, then delete the target.

Do not configure privately owned relationships to objects that might be shared. An object should not be the target in more than one relationship if it is the target in a privately owned relationship.

The exception to this rule is the case when you have a many-to-many relationship in which a relation object is mapped to a relation table and is referenced through a one-to-many relationship by both the source and the target. In this case, if the one-to-many mapping is configured as privately owned, then when you delete the source, all the association objects will be deleted.

Consider [Example 114–15](#).

#### **Example 114–15** *Privately Owned Relationships*

```
// If the Pet-PetOwner relationship is privateOwned
// then the PetOwner will be deleted at uow.commit()
// otherwise, just the foreign key from PET to PETOWNER will
// be set to null. The same is true for VetVisit
UnitOfWork uow = session.acquireUnitOfWork();
Pet petClone = (Pet)uow.readObject(Pet.class);
petClone.setPetOwner(null);
```

```
VetVisit vvClone =
    (VetVisit)petClone.getVetVisits().get(0);
vvClone.setPet(null);
petClone.getVetVisits().remove(vvClone);
uow.commit();
```

If the relationships from Pet to PetOwner and from Pet to VetVisit are not privately owned, this code produces the following SQL:

```
UPDATE PET SET PET_OWN_ID = NULL WHERE (ID = 150)
UPDATE VETVISIT SET PET_ID = NULL WHERE (ID = 350)
```

If the relationships are privately owned, this code produces the following SQL:

```
UPDATE PET SET PET_OWN_ID = NULL WHERE (ID = 150)
UPDATE VETVISIT SET PET_ID = NULL WHERE (ID = 350)
DELETE FROM VETVISIT WHERE (ID = 350)
DELETE FROM PETOWNER WHERE (ID = 250)
```

## 114.7.2 How to Explicitly Delete from the Database

If there are cases where you have objects that will not be garbage collected through privately owned relationships (especially root objects in your object model), then you can explicitly tell TopLink to delete the row representing the object using the `deleteObject` API, as shown in [Example 114–16](#).

### **Example 114–16** Explicitly Deleting

```
UnitOfWork uow = session.acquireUnitOfWork();
pet petClone = (Pet)uow.readObject(Pet.class);
uow.deleteObject(petClone);
uow.commit();
```

The preceding code generates the following SQL:

```
DELETE FROM PET WHERE (ID = 100)
```

## 114.7.3 What You May Need to Know About the Order in which Objects Are Deleted

The unit of work does not track changes or the order of operations. It is intended to insulate you from having to modify your objects in the order the database requires.

By default, at commit time, the unit of work correctly puts in order all insert and update operations using the constraints defined by your schema. After all insert and update operations are done, the unit of work will issue the necessary delete operations.

Constraints are inferred from one-to-one and one-to-many mappings. If you have no such mappings, you can add additional constraint knowledge to TopLink as described in [Section 115.10, "Controlling the Order of Delete Operations"](#).

---

---

## Using Advanced Unit of Work API

This chapter explains the advanced unit of work API calls and techniques that you are most likely to use later in the development cycle.

This chapter includes the following sections:

- [Registering and Unregistering Objects](#)
- [Declaring Read-Only Classes](#)
- [Writing Changes Before Commit Time](#)
- [Using Conforming Queries and Descriptors](#)
- [Merging Changes in Working Copy Clones](#)
- [Resuming a Unit of Work After Commit](#)
- [Reverting a Unit of Work](#)
- [Using a Nested or Parallel Unit of Work](#)
- [Using a Unit of Work with Custom SQL](#)
- [Controlling the Order of Delete Operations](#)
- [Using Optimistic Read Locking with the `forceUpdateToVersionField` Method](#)
- [Implementing User and Date Auditing with the Unit of Work](#)
- [Integrating the Unit of Work with an External Transaction Service](#)
- [Integrating the Unit of Work with CMP](#)
- [Database Transaction Isolation Levels](#)
- [Troubleshooting a Unit of Work](#)

For more information about the available methods for the `UnitOfWork`, see *Oracle Fusion Middleware Java API Reference for Oracle TopLink*.

### 115.1 Registering and Unregistering Objects

The unit of work provides a number of object registration options.

This section describes the following:

- [How to Create and Register a New Object in One Step Using `UnitOfWork` Method `newInstance`](#)
- [How to Use the `registerAllObjects` Method](#)
- [How to Use Registration and Existence Checking](#)

- [How to Work with Aggregates](#)
- [How to Unregister Working Clones](#)
- [What You May Need to Know About Object Registration](#)

### 115.1.1 How to Create and Register a New Object in One Step Using UnitOfWork Method newInstance

[Example 115–1](#) shows how to use the unit of work `newInstance` method to create a new `Pet` object, register it with the unit of work, and return a clone, all in one step. If you are using a factory design pattern to create your objects (and have specified this in the query builder), the `newInstance` method will use the appropriate factory.

#### **Example 115–1** *Creating and Registering an Object in One Step*

```
UnitOfWork uow = session.acquireUnitOfWork();
Pet petClone = (Pet)uow.newInstance(Pet.class);
petClone.setId(100);
petClone.setName("Fluffy");
petClone.setType("Cat");
uow.commit();
```

### 115.1.2 How to Use the registerAllObjects Method

The `registerAllObjects` method takes a `Collection` of objects as an argument and returns a `Collection` of clones. This lets you register many objects at once as shown in [Example 115–2](#).

---

---

**Note:** You cannot use `UnitOfWork` methods `registerObject`, `registerNewObject`, or `registerExistingObject` with an aggregate object (see [Section 22.2.1.2, "Creating Relational Aggregate Descriptors"](#)). Doing so will raise a `ValidationException` or other errors at commit time. For more information, see [Section 115.1.4, "How to Work with Aggregates"](#).

---

---

#### **Example 115–2** *Using registerAllObjects*

```
UnitOfWork uow = session.acquireUnitOfWork();
Collection toRegister = new ArrayList(2);
VetVisit vv1 = new VetVisit();
vv1.setId(70);
vv1.setNotes("May have flu");
vv1.setSymptoms("High temperature");
toRegister.add(vv1);

VetVisit vv2 = new VetVisit();
vv2.setId(71);
vv2.setNotes("May have flu");
vv2.setSymptoms("Sick to stomach");
toRegister.add(vv2);

uow.registerAllObjects(toRegister);
uow.commit();
```



### 115.1.3 How to Use Registration and Existence Checking

When you register an object with the unit of work, TopLink runs an existence check to determine whether or not the object exists. TopLink uses this information at commit time to determine whether to perform an insert or an update operation. You can specify the default existence checking policy for a project as a whole (see [Section 117.7, "Configuring Existence Checking at the Project Level"](#)) or on a per-descriptor basis ([Section 119.17, "Configuring Cache Existence Checking at the Descriptor Level"](#)). By default, TopLink uses the check cache existence checking policy. If you use any existence checking policy other than check cache, then you can use the way you register your objects to your advantage to reduce the time it takes TopLink to register an object.

This section explains how to use one of the following existence checking policies to accelerate object registration:

- [Using Check Database](#)
- [Using Assume Existence](#)
- [Using Assume Nonexistence](#)

#### 115.1.3.1 Using Check Database

If you configure a class's descriptor with an existence checking policy of check database, TopLink will check the database for existence for all instances of that class registered in a unit of work. However, if you know that an object is new or existing, rather than use the basic `registerObject` method, you can use `registerNewObject` or `registerExistingObject` to bypass the existence check. TopLink will not check the database for existence on objects that you have registered with these methods. It will automatically perform an insert operation if `registerNewObject` is called, or an update operation if `registerExistingObject` is called.

#### 115.1.3.2 Using Assume Existence

If you configure a class's descriptor with an existence checking policy of assume existence, TopLink will assume that all instances of that class registered with a unit of work exist and TopLink will always perform an update operation to the database on all such registered objects; even new objects that you registered with `registerObject` method. However, if you use the `registerNewObject` method on the new object, TopLink knows to perform an insert operation in the database even though the existence checking policy says assume existence.

#### 115.1.3.3 Using Assume Nonexistence

If you configure a class's descriptor with an existence checking policy of assume nonexistence then TopLink assumes that all instances of that class registered with a unit of work do not exist and will always perform an insert operation on the database, even on objects read from the database. However, if you use the `registerExistingObject` method on existing objects, TopLink knows to perform an update operation on the database.

### 115.1.4 How to Work with Aggregates

Aggregate mapped objects should never be registered in a TopLink unit of work—doing so will generate an exception. Aggregate cloning and registration is automatic based on the owner of the aggregate object. In other words, if you register

the owner of an aggregate, the aggregate is automatically cloned. When you get a working copy of an aggregate owner, its aggregate is also a working copy.

When working with aggregates, you should always use an aggregate *within the context of its owner*:

- If you get an aggregate from a working clone owner, then the aggregate is a working clone.
- If you get an aggregate from a cache version owner, then the aggregate is the cache version.

For more information about aggregate objects, see [Section 22.2.1.2, "Creating Relational Aggregate Descriptors"](#).

### 115.1.5 How to Unregister Working Clones

The unit of work `unregisterObject` method lets you unregister a previously registered object from a unit of work. An unregistered object will be ignored in the unit of work, and any uncommitted changes made to the object up to that point will be discarded.

In general, this method is rarely used. It can be useful if you create a new object, but then decide to delete it in the same unit of work (which is not recommended).

### 115.1.6 What You May Need to Know About Object Registration

[Table 115–1](#) summarizes the `UnitOfWork` object registration methods.

**Table 115–1 UnitOfWork Object Registration API**

Task	UnitOfWork Method	Result
Create new object without children	<code>registerObject</code>	New object is reference to cache object after commit.
Create new object without children	<code>registerNewObject</code>	New object is reference to cache object after commit. Performance enhancement: no clone created.
Create new child object for existing parent: unidirectional relationship	Parent: <code>registerObject</code> Child: no registration necessary	<code>TopLink</code> cascades registration to all new child objects reachable from the registered object: no need to register new child objects.
Create new child object for existing parent: unidirectional relationship	Parent: <code>registerObject</code> Child: <code>registerObject</code>	New object is reference to cache object after commit.
Create new child object for existing parent: bidirectional relationship	Parent: <code>registerObject</code> Child: no registration necessary	<code>TopLink</code> cascades registration to all new child objects reachable from the registered object: no need to register new child objects.
Create new child object for existing parent: bidirectional relationship	Parent: <code>registerObject</code> Child: <code>registerNewObject</code>	New object can be queried prior to commit (without using conforming query). Calling <code>registerObject</code> on child is an error.
Modify existing object known to exist in the database.	<code>registerExistingObject</code>	Performance enhancement: no call to <code>Descriptor</code> method <code>doesExist</code> .
Modify a Collection of objects	<code>registerAllObjects</code>	Convenience method: equivalent to calling <code>registerObject</code> for each object in the Collection.

If a new object is reachable from a clone, you do not need to register it.

When working with new objects, remember the following:

- Only reachable or registered objects will be persisted.

- Reachable new objects or objects that have been registered with `registerNewObject` are considered to be working copies in the unit of work.
- If you call `registerObject` with a new object, the clone the method returns, and the argument you pass in, are considered the cache version.

The `registerNewObject` method registers a new object as if it was a clone. At commit time, the unit of work creates another instance of the object to be the cache version of that object. Use the `registerNewObject` method in situations where the following applies:

- You do not need a handle to the cache version of the object after the commit transaction and you do not want to work with clones of new objects.
- You must pass a clone into the constructor of a new object, then register the new object.

Note that if you call `registerNewObject` on an object, `TopLink` does not cascade registration to new children of that object. Children will be persisted but you cannot query them before commit (unless you use conforming queries).

To make children visible to queries before commit, you must do one of the following:

- register the parent using `registerObject`
- register each child using `registerNewObject` if you register the parent using `registerNewObject`

For more information, see the following:

- [Section 114.4, "Associating a New Target to an Existing Source Object"](#)
- [Section 114.5, "Associating a New Source to an Existing Target Object"](#)
- [Section 114.6, "Associating an Existing Source to an Existing Target Object"](#)
- [Section 115.4, "Using Conforming Queries and Descriptors"](#)

## 115.2 Declaring Read-Only Classes

You can declare a class as read-only within the context of a unit of work. Clones are neither created nor merged for such classes, thus improving performance. Such classes are ineligible for changes in the unit of work.

When a unit of work registers an object, it traverses and registers the entire object tree. If the unit of work encounters a read-only class, it does not traverse that branch of the tree, and does not register objects referenced by the read-only class, so those classes are ineligible for changes in the unit of work. The read-only classes are cached and must not be changed by the user.

This section describes the following:

- [How to Configure Read-Only Classes for a Single Unit of Work](#)
- [How to Configure Default Read-Only Classes](#)
- [How to Declare Read-Only Descriptors](#)

Alternatively, you can configure an object-level ready query as a read-only query. For more information, see [Section 108.7.1.4, "Read-Only Query"](#).

### 115.2.1 How to Configure Read-Only Classes for a Single Unit of Work

For example, suppose class A owns a class B, and class C extends class B. You acquire a unit of work in which you know only instances of class A will change: you know that no class Bs will change. Before registering an instance of class B, use the following:

```
myUnitOfWork.addReadOnlyClass(B.class);
```

You can then proceed with your transaction: registering class A objects, modifying their working copies, and committing the unit of work.

At commit time, the unit of work will not have to compare backup clones with the working clones for instances of class B (even if instances were registered explicitly or implicitly). This can improve unit of work performance if the object tree is very large.

Note that if you register an instance of class C, the unit of work does not create or merge clones for this object; any changes made to class C are not persisted because class C extends class B and class B was identified as read-only.

To identify multiple classes as read-only, add them to a `Vector` and use the following code:

```
myUnitOfWork.addReadOnlyClasses(myVectorOfClasses);
```

Note that a nested unit of work inherits the set of read-only classes from the parent unit of work. For more information on using a nested unit of work, see [Section 115.8, "Using a Nested or Parallel Unit of Work"](#).

### 115.2.2 How to Configure Default Read-Only Classes

To establish a default set of read-only classes for all units of work, use the project method `setDefaultReadOnlyClasses(Vector)`. After you call this method, all new units of work include the `Vector` of read-only classes.

### 115.2.3 How to Declare Read-Only Descriptors

When you declare a class as read-only, the read-only declaration extends to its descriptors. You can declare a descriptor as read-only at development time, using either Java code, Oracle JDeveloper, or TopLink Workbench. This option improves performance by excluding the read-only descriptors from unit of work registration and editing.

To flag descriptors as read-only in Java code, call the `setReadOnly` method on the descriptor as follows:

```
descriptor.setReadOnly();
```

To declare a descriptor as read-only in TopLink Workbench, select the **Read Only** check box for the specific descriptor.

For more information, see [Section 119.3, "Configuring Read-Only Descriptors"](#).

## 115.3 Writing Changes Before Commit Time

By default, when you call the unit of work `commit` method, TopLink writes your changes to the data source and commits your changes.

Alternatively, you can perform a two-stage or partial commit transaction by using the unit of work `writeChanges` method prior to calling `commit` (either directly or by way of an external transaction service).

When you call the unit of work `writeChanges` method, the unit of work commit process begins, and all changes are written out to the data source. However, the data source transaction is not committed, nor will changes be merged into the shared

session cache. To finalize your changes, the unit of work `commit` method must still be called (either directly or by way of an external transaction service).

After you call the unit of work `writeChanges` method, any attempt to register objects or execute object-level queries will throw an exception. You may execute report queries, noncaching queries, and data read and modify queries.

If any exception is thrown, the transaction will be rolled back (or marked rollback only) and you cannot recover the unit of work.

You can call this method only once. You cannot use this method to write out changes in an incremental fashion.

You can use the unit of work `writeChanges` method to address a variety of transaction issues, including the following:

- As an alternative to conforming (see [Section 115.4.4.1, "Using Unit of Work Method writeChanges Instead of Conforming"](#))
- To handle external transaction issues (see [Section 115.13.4, "How to Use the Unit of Work to Handle External Transaction Timeouts and Exceptions"](#))

## 115.4 Using Conforming Queries and Descriptors

Because queries are executed on the database, querying though a unit of work will not, by default, include new, uncommitted objects in a unit of work. The unit of work will not spend time executing your query against new, uncommitted objects in the unit of work unless you explicitly tell it to. If you have uncommitted changes, this can pose a problem in a unit of work. Uncommitted changes not yet written to the database cannot influence which result set gets returned.

**Conforming** is a query feature that lets you include new, changed, or deleted objects in queries within a unit of work prior to committing. This lets you to query against your relative logical or transaction view of the database.

Before you use conforming, be aware of its limitations (see [Section 115.4.1, "How to Use Conforming"](#)) and make sure that conforming is actually necessary. For example, consider the alternative described in [Section 115.4.4, "What You May Need to Know About Conforming Query Alternatives"](#).

This section explains the following:

- [How to Use Conforming](#)
- [How to Use Conforming Queries](#)
- [How to Use Conforming Descriptors](#)
- [What You May Need to Know About Conforming Query Alternatives](#)

---

---

**Note:** By default, TopLink suppresses exceptions thrown during the memory search stage of conforming. For more information on handling exceptions during conforming, see [Section 115.16.4.2, "Handling Exceptions During Conforming"](#).

---

---

### 115.4.1 How to Use Conforming

When using conforming, follow the guidelines that this section describes to ensure that conforming queries return the correct results:

- [Ensuring that the Query Supports Conforming](#)

- [Considering how Conforming Affects Database Results](#)
- [Registering New Objects and Instantiate Relationships](#)

#### 115.4.1.1 Ensuring that the Query Supports Conforming

Conforming is supported by the following queries:

- `ReadObjectQuery`
- `ReadAllQuery`
- Expressions
- EJB QL
- Query by example
- Query by selection object or primary key (only new or deleted objects apply)

Conforming is *not* supported by the following queries:

- `ReportQuery`
- `DataReadQuery`
- `DataReadQuery` (deleted objects can be conformed)
- `StoredProcedureCall` (deleted objects can be conformed)
- `EISCall` (deleted objects can be conformed)
- Expression or EJB QL queries that use database-specific functions, or subselects
- Parallel expressions.

#### 115.4.1.2 Considering how Conforming Affects Database Results

When conforming is used on a `ReadAllQuery`, the database result is first queried. If the unit of work has not yet committed any changes to the database, this result will not reflect the unit of work changes. The database results are then conformed in memory using the following criteria:

- Registered new objects that conform to the query are added.
- Modified existing objects that no longer conform are removed.
- Modified existing objects that conform are added.
- Deleted objects are removed.

---

---

**Note:** If new objects are not explicitly registered, they are not conformed. Also, if removed object are not explicitly deleted, they are not conformed.

---

---

If the query uses ordering, ordering of conformed results is not maintained and conformed instances are added to the front of the result. To apply ordering, store the result in memory using `Collections` method `sort`, or a `TreeSet` result collection class. When using conforming on a `ReadObjectQuery`, first query the unit of work for a conforming object: if the conforming object is found, it is returned and the database access is avoided; if the conforming object is not found, the database is queried. If the unit of work has not yet committed any changes to the database, this result does not reflect the unit of work changes. The database results are then conformed in memory using the following criteria:

- If the database result no longer conforms, null is returned.
- If the database result has been deleted, null is returned.

---

**Note:** If the database result returns multiple results, only the first result is considered, because it is an instance of the `ReadAllQuery` and only a single result is expected. If the first result no longer conforms, null is returned, even if there were potential valid conforming results.

If you expect the query to return multiple results, use a `ReadAllQuery`.

---

### 115.4.1.3 Registering New Objects and Instantiate Relationships

If a new object is only related to an existing object, and not explicitly registered, queries for this object are not able to conform it. If you remove, but do not explicitly delete a privately owned object, this object cannot be conformed.

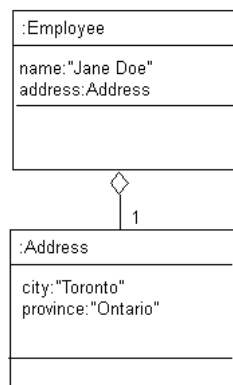
If a query traverses relationships (uses joins) and the related objects are changed, the query can only conform these objects if both of the following conditions are met:

- The source objects have been registered in the unit of work.
- The source objects' relationship has been instantiated.

`TopLink` provides a conforming option that forces an instantiation of indirection (lazy loading). However, you use this option with caution as it can cause an increased database access.

Consider [Example 115–1](#). In this example, you have `Employee` objects with an `address` attribute configured for indirection (see [Section 17.2.4, "Indirection \(Lazy Loading\)"](#)) mapped by a one-to-one mapping to an `Address` object.

**Figure 115–1 Conforming Example**



You want to read all employees who live in Ottawa, but first, you need to modify some of the `Address` objects to change city from Toronto to Ottawa. Jane Doe is one such employee.

First, using the `UnitOfWork`, you read all `Address` objects and change some `city` attributes (including Jane's) from Toronto to Ottawa. Then you run a conforming query to get all employees who live in Ottawa. However, for the following reasons Jane is not included in the results, even though she now lives in Ottawa:

- Jane is not returned from the database because the transaction has not yet been committed and in the database, her address still says Toronto.
- Jane cannot be added to the conformed result in memory because she is not registered in the `UnitOfWork` cache.

Conforming only recognizes *explicit* changes. In this example, Jane Doe's `Employee` object was only *implicitly* changed. In order to be considered explicitly changed, an `Employee` must meet the following criteria:

- Be registered in the `UnitOfWork`.
- Have its `address` attribute changed: in this example, indirection (lazy loading) must be triggered for the `address` attribute.

The correct way to handle this example would be as follows:

1. Using the `UnitOfWork`, read in all employees.  
All these `Employee` objects are now registered with the `UnitOfWork`
2. Using the same `UnitOfWork`, access the employees' addresses, instantiating the indirect relationships.
3. Modify the employees' addresses, changing some of the addresses to be in Ottawa.
4. Run the conforming query on employees with addresses inside Ottawa.  
All employees with addresses in Ottawa are returned, including both employees that were in Ottawa originally and employees whose addresses were changed in this transaction.
5. Commit the transaction.

If you do not register all employees whose address may be changed, and instantiate their address relationship, the conforming query will not include Jane.

An alternate approach is to use short transactions: the safest conforming query is one made immediately after a commit. For example:

1. Using the `UnitOfWork`, read in all addresses outside of Ottawa.
2. Modify the addresses, changing some of the addresses to be in Ottawa
3. Commit the transaction.
4. Using the `UnitOfWork`, read in all employees inside Ottawa.

## 115.4.2 How to Use Conforming Queries

Assume that a single `Pet` of type `Cat` already exists on the database. Examine the code shown in [Example 115-3](#).

### **Example 115-3 Using Conforming Queries**

```
UnitOfWork uow = session.acquireUnitOfWork();
Pet pet2 = new Pet();
Pet petClone = (Pet)uow.registerObject(pet2);
petClone.setId(200);
petClone.setType("Cat");
petClone.setName("Mouser");

ReadAllQuery readAllCats = new ReadAllQuery();
readAllCats.setReferenceClass(Pet.class);
ExpressionBuilder builder = new ExpressionBuilder();
Expression catExp = builder.get("type").equal("Cat");
```



```
readAllCats.setSelectionCriteria(catExp);

List allCats = (List)uow.executeQuery(readAllCats);

System.out.println("All 'Cats' read through UOW are: " + allCats);
uow.commit();
```

This produces the following output:

```
All 'Cats' read through UOW are: [Pet type Cat named Fluffy id:100]
If you tell the query readAllCats to include new objects:
```

```
readAllCats.conformResultsInUnitOfWork();
The output would be as follows:
```

```
All 'Cats' read through UOW are: [Pet type Cat named Fluffy id:100, Pet type Cat
named Mouser id:200]
```

### 115.4.3 How to Use Conforming Descriptors

TopLink's support for conforming queries in the unit of work can be specified at the descriptor level.

You can define a descriptor directly to always conform results in the unit of work so that all queries performed on this descriptor conform its results in the unit of work by default. You can specify this either within code, from Oracle JDeveloper, or TopLink Workbench.

You can configure a descriptor to always conform in the unit of work using Oracle JDeveloper, TopLink Workbench, or Java code.

To configure a descriptor to always conform in the unit of work in Java code, use Descriptor method `setShouldAlwaysConformResultsInUnitOfWork`, passing in an argument of `true`.

To configure a descriptor to always conform in the unit of work using TopLink, see [Section 119.4, "Configuring Unit of Work Conforming at the Descriptor Level"](#).

### 115.4.4 What You May Need to Know About Conforming Query Alternatives

This section describes alternatives to conforming that may meet your needs without the performance penalty imposed by conforming. This section describes the following:

- [Using Unit of Work Method `writeChanges` Instead of Conforming](#)
- [Using Unit of Work Properties Instead of Conforming](#)

#### 115.4.4.1 Using Unit of Work Method `writeChanges` Instead of Conforming

Using `UnitOfWork` method `writeChanges`, you can write uncommitted changes to the data source: you can execute report queries, noncaching queries, and data read and modify queries against these changes (see [Example 115-4](#)).

##### **Example 115-4 Using Unit of Work Method `writeChanges`**

```
UnitOfWork uow = session.acquireUnitOfWork();
Pet pet = new Pet();
Pet petClone = (Pet)uow.registerObject(pet);
petClone.setId(100);
petClone.setName("Fluffy");
petClone.setType("Cat");

uow.writeChanges();
```

```
// Use uow to perform report, noncaching, and data read and modify queries  
// against the changes made so far
```

```
uow.commit();
```

However, you can call `writeChanges` only once; any attempt to register objects or to execute object-level queries will throw an exception.

For more information, see [Section 115.3, "Writing Changes Before Commit Time"](#)

#### 115.4.4.2 Using Unit of Work Properties Instead of Conforming

Sometimes, you need to provide other code modules with access to new objects created in a unit of work. Conforming can be used to provide this access. However, the following alternative is significantly more efficient.

Somewhere a unit of work is acquired from a session and is passed to multiple modules for portions of the requisite processing:

```
UnitOfWork uow = session.acquireUnitOfWork();
```

In the module that creates the new employee, note the following:

```
Pet newPet = new Pet();  
Pet newPetClone = (Pet)uow.registerObject(newPet);  
uow.setProperty("NEW PET", newPet);
```

In other modules where `newPet` needs to be accessed for further modification, it can simply be extracted from the unit of work's properties:

```
Pet newPet = (Pet) uow.getProperty("NEW PET");  
newPet.setType("Dog");
```

Conforming queries are ideal if you are not sure if an object has been created yet or the criteria is dynamic.

However, for situations where the quantity of objects is finite and well known, using unit of work properties is a simple and more efficient solution.

## 115.5 Merging Changes in Working Copy Clones

In a three-tier application, the client and server exchange objects using a serialization mechanism such as RMI or CORBA. When the client changes an object and returns it to the server, you cannot register this serialized object into a unit of work directly. On the server, you must merge the serialized object with the original object in the session cache.

Using the unit of work methods listed in [Table 115–2](#), you can merge a deserialized object into your session cache. Each method takes the serialized object as an argument and returns the original object.

Before doing so, you must ensure that the source object is in your session cache. Attempting to merge a deserialized object into a session cache that does not yet contain the object will result in a descriptor exception. To avoid this, Oracle recommends that you first read the object instance that the deserialized object represents. If you are using a coordinated cache or your application is running in a cluster, the session you merge into may not yet contain your original object. By performing a read operation first, you guarantee that the object will be in the cache before you merge.

**Table 115–2 Unit of Work Merge Methods**

Method	Purpose	Used When
<code>mergeClone</code>	Merges the serialized object and all its privately owned parts (excluding non-private references from it to independent objects) into the working copy clone.	The client edits the object but not its relationships, or marks its independent relationships as transient.
<code>mergeCloneWithReferences</code>	Merges the serialized object and all references into the working copy clone.	The client edits the object and the targets of its relationships and has not marked any attributes as transient.
<code>shallowMergeClone</code>	Merges only serialized object changes to attributes mapped with direct mappings into the working copy clone.	The client edits only the object's direct attributes or has marked all of the object's relationships as transient.
<code>deepMergeClone</code>	Merges the serialized object and everything connected to it (the entire object tree where the serialized object is the root) into the working copy clone.	The client traverses all relationships of the objects and makes changes.  Note: Use <code>deepMergeClone</code> with caution. If two different copies of an object are in the same tree, <code>TopLink</code> will merge one set of changes over the other. You should not have any transient attributes in any of your related objects.

Note that if your three-tier client is sufficiently complex, consider using the `TopLink` remote session (see [Section 87.9, "Remote Sessions"](#)). It automatically handles merging and lets you use a unit of work on the client.

You can merge clones with both existing and new objects. Because clones do not appear in the cache and may not have a primary key, you can merge new objects only once within a unit of work. If you need to merge a new object more than once, call the unit of work `setShouldNewObjectsBeCached` method, and ensure that the object has a valid primary key; you can then register the object.

[Example 115–5](#) shows one way to update the original object with the changes contained in the corresponding serialized object (`rmiClone`) received from a client.

**Example 115–5 Merging a Serialized Object**

```
update(Object original, Object rmiClone) {
    original = uow.registerObject(original);
    uow.mergeCloneWithRefereneces(rmiClone);
    uow.commit();
}
```

For more information, see [Section 17.2.4.7, "Indirection, Serialization, and Detachment"](#).

## 115.6 Resuming a Unit of Work After Commit

At commit time, a unit of work and its contents expire: you must not use the unit of work nor its clones even if the transaction failed and rolled back.

However, `TopLink` offers the following API that lets you continue working with a unit of work and its clones:

- `commitAndResume`: Commits the unit of work, but does not invalidate it or its clones.

- `commitAndResumeOnFailure`: Commits the unit of work. If the commit transaction succeeds, the unit of work expires. However, if the commit transaction fails, this method does not invalidate the unit of work or its clones. This method lets you modify the registered objects in a failed unit of work and retry the commit transaction.

You should resume a unit of work only in an application that makes repeated changes to the same, small dataset. Reusing the same unit of work while accessing different datasets may result in poor performance.

[Example 115–6](#) shows how to use the `commitAndResume` method.

**Example 115–6 Using the `commitAndResume` Method**

```
UnitOfWork uow = session.acquireUnitOfWork();
PetOwner petOwnerClone =
    (PetOwner)uow.readObject(PetOwner.class);
petOwnerClone.setName("Mrs. Newowner");
uow.commitAndResume();
petOwnerClone.setPhoneNumber("KL5-7721");
uow.commit();
```

The `commitAndResume` call produces the following SQL:

```
UPDATE PETOWNER SET NAME = 'Mrs. Newowner' WHERE (ID = 400)
```

Then, the `commit` call produces the following SQL:

```
UPDATE PETOWNER SET PHN_NBR = 'KL5-7721' WHERE (ID = 400)
```

## 115.7 Reverting a Unit of Work

Under certain circumstances, you may want to abandon some or all changes to clones in a unit of work, but not abandon the unit itself. The following options exist for reverting all or part of the unit of work:

- `revertObject`: Abandons changes to a specific working copy clone in the unit of work
- `revertAndResume`: Uses the backup copy clones to restore all clones to their original states, deregister any new objects, and reinstate any deleted objects.

## 115.8 Using a Nested or Parallel Unit of Work

You can use a unit of work within another unit of work (nesting), or you can use two or more units of work with the same objects in parallel.

This section describes the following:

- [How to Use Parallel Unit of Work](#)
- [How to Use Nested Unit of Work](#)

### 115.8.1 How to Use Parallel Unit of Work

To start multiple units of work that operate in parallel, call the `acquireUnitOfWork` method multiple times on the session. The units of work operate independently of one another and maintain their own cache.

## 115.8.2 How to Use Nested Unit of Work

To nest units of work, call the `acquireUnitOfWork` method on the parent unit of work. This creates a child unit of work with its own cache. If a child unit of work commits, it updates the parent unit of work rather than the database. If the parent does not commit, the changes made to the child are not written to the database.

`TopLink` does not update the database or the cache until the outermost unit of work is committed. You must commit or release the child unit of work before you can commit its parent.

Working copy clones from one unit of work are not valid in another units of work: not even between an inner and outer unit of work. You must register objects at all levels of a unit of work where they are used.

[Example 115–7](#) shows how to use nested units of work.

### **Example 115–7 Using Nested Units of Work**

```
UnitOfWork outerUOW = session.acquireUnitOfWork();
Pet outerPetClone = (Pet)outerUOW.readObject(Pet.class);

UnitOfWork innerUOWa = outerUOW.acquireUnitOfWork();
Pet innerPetCloneA =
    (Pet)innerUOWa.registerObject(outerPetClone);
innerPetCloneA.setName("Muffy");
innerUOWa.commit();

UnitOfWork innerUOWb = outerUOW.acquireUnitOfWork();
Pet innerPetCloneB =
    (Pet)innerUOWb.registerObject(outerPetClone);
innerPetCloneB.setName("Duffy");
innerUOWb.commit();
outerUOW.commit();
```

## 115.9 Using a Unit of Work with Custom SQL

You can execute native SQL or invoke a stored procedure within a unit of work by using unit of work method `executeNonSelectingCall`, or by executing a `DataModifyQuery`. This makes the unit of work begin its database transaction early and execute the call to the data immediately.

If you release the unit of work, it will roll back the database changes. If you commit the unit of work and the commit succeeds, the unit of work will commit the changes to the database.

You can execute a `DataModifyQuery` only in a unit of work or a database session. You cannot execute a `DataModifyQuery` in a client or server session directly.

You can execute a `DataReadQuery` or use session method `executeSelectingCall` in any session type because these do not modify data.

[Example 115–8](#) illustrates using `SQLCall` with the unit of work method `executeNonSelectingCall`.

### **Example 115–8 Using the executeNonSelectingCall Method**

```
uow.executeNonSelectingCall(new SQLCall(mySqlString));
```

---

---

**WARNING:** Allowing an unverified SQL string to be passed into methods makes your application vulnerable to SQL injection attacks.

---

---

## 115.10 Controlling the Order of Delete Operations

Section 114.7, "Deleting Objects" explained that TopLink always properly arranges (orders) the SQL based on the mappings and foreign keys in your object model and schema. You can control the order of delete operations if you know how to do the following:

- [How to Use the `setShouldPerformDeletesFirst` Method of the Unit of Work](#)
- [How to Use the `addConstraintDependencies` Method of the Descriptor](#)

### 115.10.1 How to Use the `setShouldPerformDeletesFirst` Method of the Unit of Work

By default, TopLink does insert and update operations first, before delete operations, to ensure that referential integrity is maintained. This is the preferred approach.

If you are forced to replace an object with unique constraints by deleting it and inserting a replacement, you may cause a constraint violation if the insert operation occurs before the delete operation. In this case, call `setShouldPerformDeletesFirst` to perform the delete operation before the insert operation.

### 115.10.2 How to Use the `addConstraintDependencies` Method of the Descriptor

The constraints used by TopLink to determine delete order are inferred from one-to-one and one-to-many mappings. If you do not have such mappings, you can add constraint knowledge to TopLink using the descriptor `addConstraintDependencies(Class)` method.

For example, suppose you have a composition of objects: A contains B (one-to-many, privately owned) and B has a one-to-one, nonprivate relationship with C. You want to delete A (and in doing so the included Bs) but before deleting the Bs, for some of them (not all) you want to delete the associated object C.

There are the following possible solutions:

1. You can use the `deleteAllObjects` method without the `addConstraintDependencies` method (see [Section 115.10.3, "How to Use the `deleteAllObjects` Method Without the `addConstraintDependencies` Method"](#))
2. You can use the `deleteAllObjects` method with the `addConstraintDependencies` method (see [Section 115.10.4, "How to Use the `deleteAllObjects` Method with the `addConstraintDependencies` Method"](#))

### 115.10.3 How to Use the `deleteAllObjects` Method Without the `addConstraintDependencies` Method

In the first option, you do not identify the one-to-many (A to B) relationship as privately owned. When deleting an A object, you must delete all of its B objects, as well as any C objects, as the following example shows:

```
uow.deleteObject(existingA);
uow.deleteAllObjects(existingA.getBs());
// delete one of the Cs
```

```
uow.deleteObject(((B) existingA.getBs().get(1)).getC());
```

This option produces the following SQL:

```
DELETE FROM B WHERE (ID = 2)
DELETE FROM B WHERE (ID = 1)
DELETE FROM A WHERE (ID = 1)
DELETE FROM C WHERE (ID = 1)
```

### 115.10.4 How to Use the `deleteAllObjects` Method with the `addConstraintDependencies` Method

In the second option, keep the one-to-many (A to B) relationship privately owned and add a constraint dependency from A to C, as the following example shows:

```
session.getDescriptor(A.class).addConstraintDependencies(C.class);
```

Now the delete code would be as follows:

```
uow.deleteObject(existingA);
// delete one of the Cs
uow.deleteObject(((B) existingA.getBs().get(1)).getC());
```

This option produces the following SQL:

```
DELETE FROM B WHERE (A = 1)
DELETE FROM A WHERE (ID = 1)
DELETE FROM C WHERE (ID = 1)
```

In both cases, the B object is deleted before A and C. The main difference is that the second option generates fewer SQL statements, as it knows that it is deleting the entire set of Bs related from A.

## 115.11 Using Optimistic Read Locking with the `forceUpdateToVersionField` Method

If your descriptors are configured to use an optimistic version locking policy (see [Section 16.4.1, "Optimistic Version Locking Policies"](#)) or field locking policy (see [Section 16.4.4, "Optimistic Field Locking Policies"](#)), use the unit of work method `forceUpdateToVersionField` to solve either or both of the following problems:

- You want an `OptimisticLockingException` thrown at commit time if an object you read in a transaction has changed since it was registered even though you have not changed the object in your transaction (see [Section 115.11.1, "How to Force a Check of the Optimistic Read Lock"](#)).
- You modify an object in a transaction in such a way that its version field is not updated but you want to alert other threads of the change by way of the version field (see [Section 115.11.2, "How to Force a Version Field Update"](#)).

For example, you change a privately owned object that has its own database table so the parent object's version field is not, by default, updated. In this case, you can use `forceUpdateToVersionField` to update the parent's version field.

As an alternative to this approach, consider [Section 16.4.2, "Optimistic Version Locking Policies and Cascading"](#).

To remove `forceUpdateToVersionField` configuration from an object before a commit operation, use the unit of work method `removeForceUpdateToVersionField` (see [Section 115.11.3, "How to Disable the `forceUpdateToVersionField` Configuration"](#)).

### 115.11.1 How to Force a Check of the Optimistic Read Lock

When you read an object with the unit of work, optimistic lock checking is not applied to that object at commit time unless you change the object. However, there are times when you want your transaction to fail if the state of an object has changed since it was read, even though you have not modified the object.

[Example 115–9](#) shows a transaction that updates a mortgage rate by multiplying the central bank prime rate by 1.25. The transaction forces an optimistic read lock on the central prime rate at commit time to ensure that the prime rate has not changed since the transaction began. Note that in this example, the transaction does not increment the version of the unchanged object (the central prime rate).

#### **Example 115–9 Optimistic Read Lock with No Version Increment**

```
try {
    UnitOfWork uow = session.acquireUnitOfWork();
    MortgageRate cloneMortgageRate = (MortgageRate)
        uow.registerObject(mortgageRate);
    CentralPrimeRate cloneCentralPrimeRate = (CentralPrimeRate)
        uow.registerObject(CentralPrimeRate);
    // change the Mortgage Rate
    cloneMortgageRate.setRate(cloneCentralPrimeRate.getRate() * 1.25);
    // optimistic read lock check on Central prime rate with no version update
    uow.forceUpdateToVersionField(cloneCentralPrimeRate, false);
    uow.commit();
}
catch(OptimisticLockException exception) {
    // refresh the out-of-date object
    session.refreshObject(exception.getObject());
    // Retry
}
```

For another example that forces both optimistic locking and a version field update, see [Example 115–10](#) in [Section 115.11.2, "How to Force a Version Field Update"](#).

### 115.11.2 How to Force a Version Field Update

The unit of work considers an object changed when you modify its direct-to-field or aggregate object mapping attribute. Adding, removing, or modifying objects related to the source object does not render the source object changed for the purposes of the unit of work. In other words, when a relationship is changed in a one-to-many or one-to-one target foreign key mapping, by default, the version field (if any) of the affected object is not changed.

If you configure a descriptor to refresh the cache only if the database version is newer than the cache version (using descriptor method `onlyRefreshCacheIfNewerVersion`), and such a relationship changes, you will not be able to refresh the object at all. Because the version has not changed, the unit of work method `refreshObject` and even a query with `refreshIdentityMapResults` option set to `true` cannot refresh the object.

Using the unit of work method `forceUpdateToVersionField` passing in both the unit of work copy clone and `true` value will ensure that the object's version field is updated when such a change is made. It will also ensure that changes to the object before it is refreshed will result in optimistic locking exceptions, preventing the writing of stale data (see [Section 115.11.1, "How to Force a Check of the Optimistic Read Lock"](#)).

[Example 115–10](#) and [Example 115–11](#) show transactions executing in separate threads that access the same customer object concurrently. The unit of work method



forceUpdateToVersionField is used to ensure that changes to the customer object in one thread are detected by the other threads.

[Example 115–10](#) shows a transaction in which an *invoice* thread calculates an invoice for a customer. [Example 115–11](#) shows a transaction in which another thread, the *service* thread, adds a service to the same customer or modifies the current service. In either case, the *service* thread must inform the *invoice* thread, which adds the changes to the invoice.

#### **Example 115–10 Optimistic Read Lock with Version Increment: Service Thread**

**/\* The following code represents the service thread. Notice that the thread forces a version update \*/**

```
try {
    UnitOfWork uow = session.acquireUnitOfWork();
    Customer cloneCustomer = (Customer) uow.registerObject(customer);
    Service cloneService = (Service) uow.registerObject(service);
    // add a service to customer
    cloneService.setCustomer(cloneCustomer);
    cloneCustomer.getServices().add(cloneService);
    /* Modify the customer version to inform other application that
       the customer has changed */
    uow.forceUpdateToVersionField(cloneCustomer, true);
    uow.commit();
}
catch (OptimisticLockException exception) {
    // refresh out-of-date object
    session.refreshObject(exception.getObject());
    // retry
}
}
```

#### **Example 115–11 Optimistic Read Lock with Version Increment: Invoice Thread**

**/\* The following code represents the invoice thread, and calculates a bill for the customer. Notice that it does not force an update to the version \*/**

```
try {
    UnitOfWork uow = session.acquireUnitOfWork();
    Customer cloneCustomer = (Customer) uow.registerObject(customer);
    Invoice cloneInvoice = (Invoice) uow.registerObject(new Invoice());
    cloneInvoice.setCustomer(cloneCustomer);
    // calculate service charge
    int total = 0;
    for(Enumeration enum = cloneCustomer.getServices().elements();
        enum.hasMoreElements()) {
        total += ((Service) enum.nextElement()).getCost();
    }
    cloneInvoice.setTotal(total);
    /* Force optimistic lock checking on the customer to guarantee a valid
       calculation */
    uow.forceUpdateToVersionField(cloneCustomer, false);
    uow.commit();
}
catch(OptimisticLockException exception) {
    // refresh the customer and its privately owned parts
    session.refreshObject(cloneCustomer);
    /* If the customer's services are not privately owned then use a
       ReadObjectQuery to refresh all parts */
    ReadObjectQuery query = new ReadObjectQuery(customer);
    /* Refresh the cache with the query's result and cascade refreshing

```

```
        to all parts including customer's services */
    query.refreshIdentityMapResult();
    query.cascadeAllParts();
    // refresh from the database
    query.dontCheckCache();
    session.executeQuery(query);
    // retry
}
```

### 115.11.3 How to Disable the `forceUpdateToVersionField` Configuration

The `forceUpdateToVersionField` configuration that you apply to an object stays in effect for the lifetime of your unit of work. After you commit your transaction, `forceUpdateToVersionField` configuration no longer applies.

To remove `forceUpdateToVersionField` configuration from an object before commit time, use the unit of work method `removeForceUpdateToVersionField`. TopLink will not apply optimistic read locking to the object unless you change it in this transaction (that is, unless you modify its direct-to-field or aggregate object mapping attribute).

## 115.12 Implementing User and Date Auditing with the Unit of Work

Auditing data source changes is a common requirement: when a user commits a change to the data source, the application updates a field in the data source to record the user who made the change and the date.

For example, suppose each row in your database schema includes fields `lastUpdateBy` (to record the user name of the user who commits a change) and `lastUpdateOn` (to record the date of the change).

You can use `UnitOfWork` method `setProperty` to record the name of the user who acquires the `UnitOfWork` and implement a descriptor event listener for `AboutToUpdateEvent` descriptor events that extracts the property and updates the `lastUpdateBy` and `lastUpdateOn` fields.

For more information, see the following:

- [Section 114.1, "Acquiring a Unit of Work"](#)
- [Section 119.24, "Configuring a Domain Object Method as an Event Handler"](#)
- [Section 119.25, "Configuring a Descriptor Event Listener as an Event Handler"](#)

## 115.13 Integrating the Unit of Work with an External Transaction Service

To support transactions managed by an application server's external transaction service, TopLink supports external connection pools and external transaction controller classes for supported servers. This lets you incorporate external transaction service support into your application, and use the unit of work with transactions managed externally by the server. For more information, see [Section 113.1.2, "Unit of Work Transaction Demarcation"](#).

To integrate a unit of work with an external transaction service, you must enable the use of the following:

- an external transaction controller (see [Section 89.9, "Configuring the Server Platform"](#))

- an external connection pool (see [Section 97.4, "Configuring External Connection Pooling"](#))

After you configure external connection pool and external transaction controller support, you use a unit of work in your TopLink application as you would typically, with few exceptions. This section describes these exceptions as follows:

- [How to Acquire a Unit of Work with an External Transaction Service](#)
- [How to Use a Unit of Work when an External Transaction Exists](#)
- [How to Use a Unit of Work when No External Transaction Exists](#)
- [How to Use the Unit of Work to Handle External Transaction Timeouts and Exceptions](#)

### 115.13.1 How to Acquire a Unit of Work with an External Transaction Service

You use a unit of work to commit changes to a data source even when using an external transaction service. To ensure that only one unit of work is associated with a given transaction, use the `getActiveUnitOfWork` method to acquire a unit of work, as shown in [Example 115–12](#).

---



---

**Note:** Although there are other ways to commit changes to a data source using an external transaction service, Oracle recommends using the `getActiveUnitOfWork` method.

---



---

The `getActiveUnitOfWork` method searches for an existing external transaction in the following way:

- If there is an active external transaction and a unit of work is already associated with it, return this unit of work.
- If there is an active external transaction with no associated unit of work, then acquire a new unit of work, associate it with the transaction, and return it.
- If there is no active external transaction in progress, return null.

If TopLink returns a unit of work that is not null, use it exactly as you would typically: the only exception is that you do not call the `commit` method (see [Section 115.13.2, "How to Use a Unit of Work when an External Transaction Exists"](#)).

If TopLink returns a null unit of work, start an external transaction explicitly through the `UserTransaction` interface.

#### **Example 115–12 Using a Unit of Work with an External Transaction Service**

```
// Read in any pet
Pet pet = (Pet)clientSession.readObject(Pet.class);
UnitOfWork uow = clientSession.getActiveUnitOfWork();
if (uow == null) {
    throw new RuntimeException("No transaction started");
}
Pet petClone = (Pet)uow.registerObject(pet);
petClone.setName("Furry");
```

### 115.13.2 How to Use a Unit of Work when an External Transaction Exists

When `getActiveUnitOfWork` returns a unit of work that is not null, you are associated with an existing external transaction. Use the unit of work as usual.

As the external transaction was not started by the unit of work, issuing a `commit` on it will not cause the external transaction to be committed. The unit of work will defer to the application or container that began the transaction. When the external transaction does get committed by the container, `TopLink` receives synchronization callbacks at key points during the commit transaction.

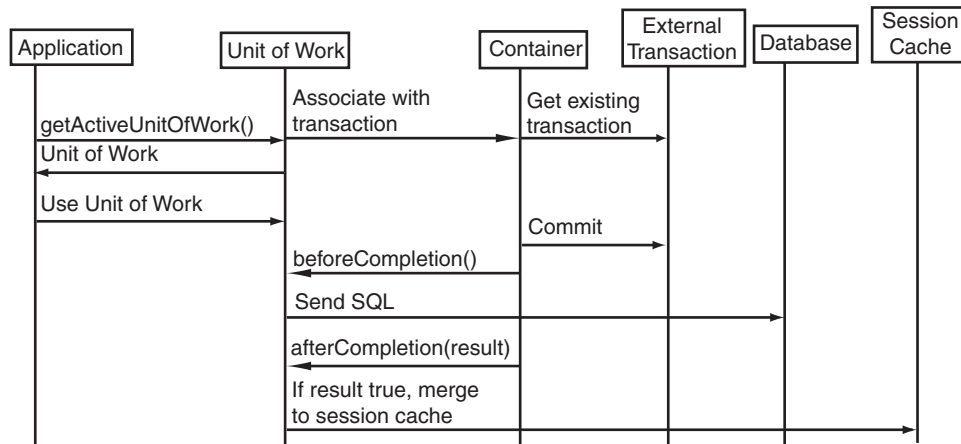
The unit of work sends the required SQL to the database when it receives the `beforeCompletion` callback.

The unit of work uses the `Boolean` argument received from the `afterCompletion` callback to determine if the commit was successful (`true`) or not (`false`).

If the commit transaction was successful, the unit of work merges changes to the session cache. If the commit transaction was unsuccessful, the unit of work discards the changes.

Figure 115–2 shows the life cycle of a unit of work when an external transaction exists.

**Figure 115–2 Unit of Work when an External Transaction Exists**



### 115.13.3 How to Use a Unit of Work when No External Transaction Exists

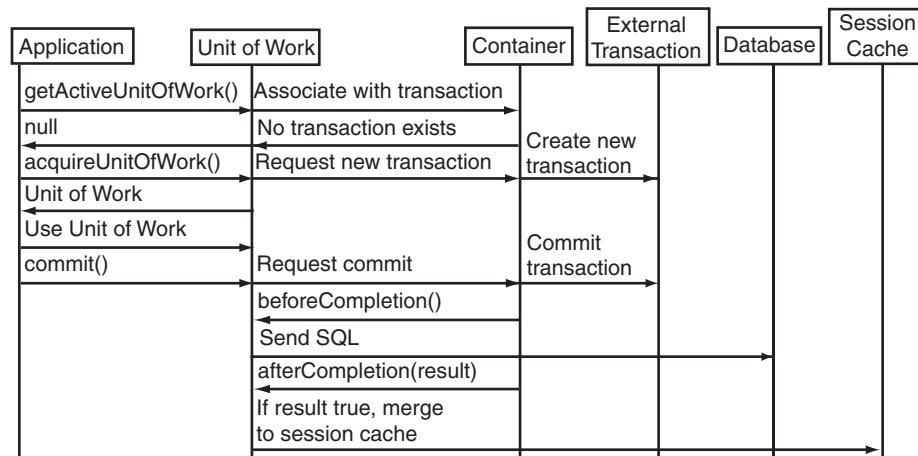
When the `getActiveUnitOfWork` method returns a null unit of work, there is no existing external transaction. You must start a new external transaction.

Do this either by starting an external transaction explicitly using the `UserTransaction` interface, or by acquiring a new unit of work using the `acquireUnitOfWork` method on the server session.

Use the unit of work as usual.

Once the modifications to registered objects are complete, you must commit the transaction either explicitly through the `UserTransaction` interface or by calling the unit of work `commit` method.

The transaction synchronization callbacks are then invoked on, and the database updates and cache merge occur based upon those callbacks.

**Figure 115-3 Unit of Work when No External Transaction Exists**

### 115.13.4 How to Use the Unit of Work to Handle External Transaction Timeouts and Exceptions

This section describes the following two common problems with external transactions, and the ways to handle them:

- [Handling External Transaction Commit Timeouts](#)
- [Handling External Transaction Commit Exceptions](#)

#### 115.13.4.1 Handling External Transaction Commit Timeouts

When an external transaction is committed, the external transaction service expects each transaction owner to commit its portion of the overall transaction within a finite amount of time. If any individual transaction exceeds this timeout interval, the external transaction service will fail the specific transaction and roll it back (or mark it rollback only).

If your transaction is large and its commit transaction may exceed the external transaction service timeout interval, use `UnitOfWork` method `writeChanges` to write changes to the data source before committing the external transaction. This will reduce the time it takes for your part of the global transaction to commit.

For more information about the `UnitOfWork` method `writeChanges`, including restrictions and warnings, see [Section 115.3, "Writing Changes Before Commit Time"](#).

#### 115.13.4.2 Handling External Transaction Commit Exceptions

When you use the unit of work with an external transaction service, commit exceptions may not be thrown until long after your application thread calls its `UnitOfWork` method `commit` and returns. In this case, commit exceptions are thrown to the client of the container-managed transaction (CMT) call, forcing the client to handle this server-side failure.

You can use the `UnitOfWork` method `writeChanges` to write changes to the data source before the external transaction commits. This allows your application thread to catch and handle most exceptions that could be thrown at the time the external transaction service commits the global transaction.

For more information about the `UnitOfWork` method `writeChanges`, including restrictions and warnings, see [Section 115.3, "Writing Changes Before Commit Time"](#).

For more information on handling unit of work exceptions in general, see [Section 115.16.4, "How to Handle Exceptions"](#).

## 115.14 Integrating the Unit of Work with CMP

All modifications to persistent beans should be carried out in the context of a transaction.

Modifying entity beans without a transaction can lead to an inconsistent state, potentially corrupting the values in the TopLink cache. Because of this, TopLink does not support modifying a bean through its remote interface when no transaction is active. If you attempt to do so, TopLink simply does not write changes to the database.

Although TopLink does not let you modify an entity bean through its remote interface without a transaction, TopLink *does* let you invoke methods on its home interface that change the state in the underlying database without a transaction. For example, you may invoke remove and create methods on the home interface of an entity bean without a transaction.

To integrate TopLink transactions and the unit of work with container-managed persistence, you must consider the following:

- CMP transaction attribute (see [Section 115.14.1, "How to Use CMP Transaction Attribute"](#))
- Local transactions (see [Section 115.14.2, "How to Use Local Transactions"](#))
- Nondeferred changes (see [Section 115.14.3, "How to Use Nondeferred Changes"](#))

### 115.14.1 How to Use CMP Transaction Attribute

To ensure that all modifications to persistent beans are carried out in the context of a transaction, transactional attributes must be properly specified in the bean deployment descriptors.

The transaction may be either client-controlled or container-controlled.

Client-controlled transactions are started explicitly by your application by way of the `javax.transaction.UserTransaction` interface.

Container-controlled transactions are started implicitly by the container to satisfy the transaction attribute configuration when a bean method is invoked in the absence of a client-controlled transaction.

[Table 115–3](#) shows what transaction (if any) an EJB method invocation uses depending on how its transaction attribute is configured and whether or not a client-controlled transaction exists at the time the method is invoked.

Oracle recommends that you do not make modifications to entity beans under conditions identified as "Use no transaction" in [Table 115–3](#). Oracle also recommends that you avoid using the `SUPPORTS` transaction attribute because it leads to a nontransactional state whenever the client does not explicitly provide a transaction.

**Table 115–3 EJB Transaction State by Transaction Attribute**

Transaction Attribute	Client-Controlled Transaction Exists	Client-Controlled Transaction Does Not Exist
NotSupported	Use no transaction	Use no transaction
Supports	Use client-controlled transaction	Use no transaction
Required	Use client-controlled transaction	Use container-controlled transaction

**Table 115-3 (Cont.) EJB Transaction State by Transaction Attribute**

Transaction Attribute	Client-Controlled Transaction Exists	Client-Controlled Transaction Does Not Exist
RequiresNew	Use client-controlled transaction	Use container-controlled transaction
Mandatory	Use client-controlled transaction	Exception raised
Never	Exception raised	Use no transaction

Depending on the EJB container you use, you may be able to write without a container-controlled transaction (see [Section 115.13, "Integrating the Unit of Work with an External Transaction Service"](#)). In this case, TopLink automatically uses a transaction of its own, referred to as a local transaction (see [Section 115.14.2, "How to Use Local Transactions"](#)).

### 115.14.2 How to Use Local Transactions

Some EJB containers, such as, for example, OC4J, support writing without an active JTA transaction.

If you execute a bean method outside a JTA transaction while the transaction attribute (see [Section 115.14.1, "How to Use CMP Transaction Attribute"](#)) is set to `Supports`, `NotSupported`, or `Never`, TopLink performs the operation within a local unit of work and commits the unit of work at the end of the method. This unit of work is referred to as a local transaction.

The reason for this is because the update semantics in the EJB specification are left undefined for these scenarios, and a proper transactional model demands that a transaction be active before being able to modify data. TopLink also requires change operations to occur within a unit of work to ensure that the session cache remains consistent.

### 115.14.3 How to Use Nondeferred Changes

Some EJB containers, such as, for example, OC4J, support nondeferred changes: the ability to modify the data source immediately as you change the persistent fields of an entity bean.

Using nondeferred changes, you can achieve backwards compatibility with the native behavior of some EJB containers, such as, for example, OC4J, and you can accommodate advanced applications that rely on the database and entity changes being synchronized for such things as triggers or stored procedures based on transient state within the transaction, deletion and creation of rows with the same primary key, or other complex queries that depend on transient transaction state.

Nondeferred changes have the disadvantage of being the least efficient approach: they produce the greatest number of data source interactions.

By default, TopLink defers all changes until commit time. This is the most efficient approach that produces the least number of data source interactions.

For more information, see [Section 16.2.3.1, "Nondeferred Changes"](#).

## 115.15 Database Transaction Isolation Levels

Achieving a particular database transaction isolation level in a TopLink application is more involved than simply using the `DatabaseLogin` method `setTransactionIsolation`.

In a typical TopLink application and in Java EE applications that require persistence in general, a variety of factors affect when database transaction isolation levels apply and to what extent a particular database transaction isolation can be achieved.

This section describes these factors and provides guidelines on configuring and using TopLink to achieve each database transaction isolation level to the extent possible given these factors.

This section includes the following:

- [What You May Need to Know About General Factors Affecting Transaction Isolation Level](#)
- [What You May Need to Know About Read Uncommitted Level](#)
- [What You May Need to Know About Read Committed Level](#)
- [What You May Need to Know About Repeatable Read Levels](#)
- [What You May Need to Know About Serializable Read Levels](#)

### 115.15.1 What You May Need to Know About General Factors Affecting Transaction Isolation Level

This section describes some of the important factors and variables that may affect the degree to which your TopLink application can achieve a particular database transaction isolation level. These factors include the following:

- [External Applications](#)
- [TopLink Coordinated Cache](#)
- [DatabaseLogin Method setTransactionIsolation](#)
- [Reading Through the Write Connection](#)
- [Managing Cache Access](#)
- [CMP and External Transactions](#)

#### 115.15.1.1 External Applications

In many cases, your TopLink application is not the only application that can update to the database. External, non-TopLink applications, can also update the database at any time.

In this case, your TopLink application must use the `ObjectLevelReadQuery` method `refreshIdentityMapResult` (see [Section 108.16.5, "How to Refresh the Cache"](#)) or `Descriptor` methods `alwaysRefreshCache` and `disableCacheHits` (see [Section 119.9, "Configuring Cache Refreshing"](#)).

For more information, see [Section 115.15.1.5, "Managing Cache Access"](#).

If the external application can update a version field in the database, your TopLink application could use `alwaysRefreshCache` in conjunction with `Descriptor` method `onlyRefreshCacheIfNewerVersion` to ensure that refresh operations are performed only when required. Another, recommended way to achieve this, is to use the descriptor isolated cache option (see [Section 102.2.7, "Cache Isolation"](#)), as well as cache invalidation (see [Section 102.2.5, "Cache Invalidation"](#)).

#### 115.15.1.2 TopLink Coordinated Cache

Consider multiple TopLink applications (each running on its own application server instance) configured to use a distributed, coordinated cache (as described in



[Section 102.3, "Cache Coordination"](#)). A TopLink application instance first commits changes to its own cache before the change is distributed to other caches. Because cache coordination is not instantaneous, there is a possibility that one TopLink application instance may read an older version of an object from its cache before a cache coordination message is received.

To provide your TopLink application with the most up-to-date version of an object use the descriptor isolated cache option (see [Section 102.2.7, "Cache Isolation"](#)), as well as cache invalidation (see [Section 102.2.5, "Cache Invalidation"](#)).

You can also avoid stale data by using Descriptor methods `alwaysRefreshCache` and `disableCacheHits`. For more information on the `disableCacheHits` method, see [Section 115.15.1.5, "Managing Cache Access"](#).

---

---

**Caution:** Using Descriptor methods `alwaysRefreshCache` and `disableCacheHits` will result in frequent database hits. Use only when absolutely necessary.

---

---

### 115.15.1.3 DatabaseLogin Method `setTransactionIsolation`

Use the `DatabaseLogin` method `setTransactionIsolation` to configure the database transaction isolation level that TopLink applies to any database connection it obtains, for example:

```
databaseLogin.setTransactionIsolation(DatabaseLogin.TRANSACTION_SERIALIZABLE);
```

This method sets the transaction isolation level used for both database read and write operations on the database connections obtained from either an internal or external connection pool (see [Section 87.2.1.2, "Connection Pools"](#)), for both internal transactions and external transactions as in the case of CMP.

However, with TopLink, by default read operations use a different database connection than write operations, typically obtained from an external connection pool, or may use the cache, bypassing the database entirely. Thus, with TopLink, by default, read operations are always performed outside the transaction or unit of work, even if you perform the read operation within a transaction or unit of work. Although database transaction isolation applies to both read and write connections, the read is not performed as part of the transaction. Therefore, the read operation overrides the transaction isolation set on the database.

Depending on the level of transaction isolation you are trying to achieve, you may require that the same transaction isolation be applied to both read and write operations. You must take special action to make TopLink use the same connection for both read and write operations. For more information, see [Section 115.15.1.4, "Reading Through the Write Connection"](#).

### 115.15.1.4 Reading Through the Write Connection

Recall that TopLink, by default, performs read operations with a different database connection than used for write operations ([Section 115.15.1.3, "DatabaseLogin Method `setTransactionIsolation`"](#)). However, from the perspective of database transaction isolation, there is a one-to-one relationship between transaction and database connection: that is, all database operations (including read operations) must use the same database connection in order to achieve a particular database transaction isolation level.

In general, when TopLink performs a read operation, if a write connection already exists, TopLink will use the write connection for the read operation. This is called

"reading through the write connection." If a write connection does not yet exist, TopLink will acquire another connection and use that for the read operation.

You can configure TopLink to allocate a write connection early using any of the following:

- [Pessimistic Locking Query](#)
- [Unit of Work Method `beginTransactionEarly`](#)
- [ConnectionPolicy Method `setShouldUseExclusiveConnection`](#)

---

---

**Caution:** Depending on the database transaction isolated level reading through the write connection may lock the object being read. This will affect performance and reduce concurrency. Oracle recommends that you do not use these advanced techniques unless strict database transaction isolation is absolutely necessary.

---

---

For more information, see [Section 115.15.1.6, "CMP and External Transactions"](#).

**115.15.1.4.1 Pessimistic Locking Query** When you use pessimistic locking (`ObjectLevelReadQuery` methods `acquireLocks` or `acquireLocksWithoutWaiting` or `Session` method `refreshAndLockObject`), TopLink does the following:

- Allocates a write connection used for both read and write operations.
- Always reads from the database.
- Always updates the cache with the database version.

**115.15.1.4.2 Unit of Work Method `beginTransactionEarly`** This method is advanced API. If you call `beginTransactionEarly` on an instance of a unit of work, all read operations should be performed through that instance of the unit of work.

This method starts a database transaction immediately: any objects you read will lock data in the database before commit time, reducing concurrency.

**115.15.1.4.3 ConnectionPolicy Method `setShouldUseExclusiveConnection`** Client sessions can access the data source using a connection pool or an exclusive connection. To use an exclusive connection, acquire your client session using a `ConnectionPolicy` (see [Section 90.4.2, "How to Acquire a Client Session that Uses Exclusive Connections"](#)).

If you are using isolated client sessions (see [Section 87.5, "Isolated Client Sessions"](#)), you can use exclusive connections for reading isolated data. In this case, you can configure TopLink to acquire an exclusive connection from the write connection pool and use it for both writing and reading isolated data. However, TopLink still acquires a shared connection from the read connection pool for reading nonisolated data.

For more information, see [Exclusive Write Connections](#).

### 115.15.1.5 Managing Cache Access

By default, TopLink uses the shared session cache as much as possible. Doing so increases concurrency and improves performance. However, to achieve a particular transaction isolation level, you may need to avoid the cache using some or all the following:

- [Isolated Client Session Cache](#)

- [ReadObjectQuery](#)
- [ReadAllQuery](#)
- [Descriptor Method disableCacheHits](#)
- [DatabaseQuery Method dontMaintainCache](#)

**115.15.1.5.1 Isolated Client Session Cache** This method always goes to the database for the initial read operation of an object whose descriptor is configured as isolated. By avoiding the shared session cache, you do not need to use the more complicated descriptor and query APIs to disable cache hits or always refresh. For more information about isolated client sessions, see [Section 87.5, "Isolated Client Sessions"](#). This is particularly useful for achieving serializable transaction isolation (see [Section 115.15.5, "What You May Need to Know About Serializable Read Levels"](#)).

**115.15.1.5.2 ReadObjectQuery** This API goes to the database unless it is a primary key-based query, in which case it will go to the cache first. For information on how to avoid the cache entirely in this case, see [Section 115.15.1.5.4, "Descriptor Method disableCacheHits"](#).

**115.15.1.5.3 ReadAllQuery** This API always goes to the databases. For information on how to avoid the cache entirely in this case, see the description of the `Descriptor` method `alwaysRefreshCache` in [Section 102.4.2, "Cache Refresh API"](#).

**115.15.1.5.4 Descriptor Method disableCacheHits** This API allows for cache hits on primary key, read-object queries to be disabled. This can be used with the `Descriptor` method `alwaysRefreshCache` to ensure queries always go to the database.

**115.15.1.5.5 DatabaseQuery Method dontMaintainCache** This is a query-level means of preventing objects from being added to the shared session cache. Using an isolated client session (see [Section 115.15.1.5.1, "Isolated Client Session Cache"](#)) is a simpler approach to achieving the same ends.

### 115.15.1.6 CMP and External Transactions

In general, the transaction isolation information in this section applies to both CMP and non-CMP applications, with the following exception:

For application servers other than OC4J, when using a TopLink application with CMP, Oracle recommends that you configure your container to use separate read and write connection pools, and to associate only the write connections with an external transaction. This means the read connections do not participate in the transaction.

---



---

**Note:** OC4J always uses the same connection pool for reading and writing: it uses JTA connections from that pool for writing, and non-JTA connections from the pool for reading.

---



---

However, because TopLink treats EJB finders as just another type of query, you can use your descriptor configuration to exploit the options described in [Section 115.15.1.4, "Reading Through the Write Connection"](#). For example, if you configure a descriptor to use pessimistic locking (see [Section 119.26, "Configuring Locking Policy"](#)), then when its finder is invoked it will allocate a write connection early and both read and write operations will use the same connection.

Refer to [Section 96.1.1, "Externally Managed Transactional Data Sources"](#) for more information on external transactions with transactional data sources.

### 115.15.2 What You May Need to Know About Read Uncommitted Level

Oracle does not recommend using this transaction isolation level.

In general, a read uncommitted operation is not necessary. Using TopLink, a transaction isolation of read committed gives you better performance than read uncommitted but with greatly improved data integrity.

### 115.15.3 What You May Need to Know About Read Committed Level

Using the unit of work guarantees that you will read only committed data in the shared session cache or committed data in the database.

### 115.15.4 What You May Need to Know About Repeatable Read Levels

To achieve repeatable read operations, you must use a unit of work, you must register all objects in the unit of work (both objects you intend to modify and objects you intend only to read), and you must use `ObjectLevelReadQuery` method `conformResultsInUnitOfWork` or `Descriptor` method `alwaysConformResultsInUnitOfWork`.

By doing so, each time you query a registered object, you will get the version of the object as it currently is in your unit of work.

### 115.15.5 What You May Need to Know About Serializable Read Levels

To achieve serializable transaction isolation with TopLink, Oracle recommends that you use an isolated client session (see [Section 87.5, "Isolated Client Sessions"](#)) as follows:

1. Configure the database transaction isolation as serializable.
2. Configure objects as isolated (see [Section 117.11, "Configuring Cache Isolation at the Project Level"](#) or [Section 119.13, "Configuring Cache Isolation at the Descriptor Level"](#)).
3. Use the `UnitOfWork` method `beginTransactionEarly` (see [Section 115.15.1.4.2, "Unit of Work Method beginTransactionEarly"](#)).

If you are only concerned about the write aspect of serializable, optimistic locking is sufficient.

To prevent phantom read transactions (that is, when a transaction detects that new records that have been added to the database after the transaction started), use the `ReadQuery` method `cacheQueryResults`.

## 115.16 Troubleshooting a Unit of Work

This section examines common unit of work problems and debugging techniques, and describes the following:

- [How to Avoid the Use of Post-Commit Clones](#)
- [How to Determine Whether or Not an Object Is the Cache Object](#)
- [How to Dump the Contents of a Unit of Work](#)
- [How to Handle Exceptions](#)

- [How to Validate a Unit of Work](#)

### 115.16.1 How to Avoid the Use of Post-Commit Clones

A common unit of work error is holding on to clones after commit time. Typically the clones are stored in a static variable and you incorrectly believe that this object is the cache copy. This leads to problems when another unit of work makes changes to the object and what you believe is the cache copy is not updated (because a unit of work updates only the cache copy, not old clones).

Consider the error in [Example 115–13](#). In this example you get a handle to the cache copy of a `Pet` and store it in the static `CACHE_PET`. We get a handle to a working copy clone and store it in the static `CLONE_PET`. In a future unit of work, the `Pet` is changed.

If you incorrectly store global references to clones from units of work, you often expect them to be updated when the cache object is changed in a future unit of work. Only the cache copy is updated.

#### **Example 115–13 Incorrect Use of Handle to Clone**

```
// Read a Pet from the database, store in static
CACHE_PET = (Pet)session.readObject(Pet.class);

// Put a clone in a static. This is a bad idea and is a common error
UnitOfWork uow = session.acquireUnitOfWork();
CLONE_PET = (Pet)uow.readObject(Pet.class);
CLONE_PET.setName("Hairy");
uow.commit();
//Later, the pet is changed again
UnitOfWork anotherUow = session.acquireUnitOfWork();
Pet petClone = (Pet)anotherUow.registerObject(CACHE_PET);
petClone.setName("Fuzzy");
anotherUow.commit();

// If you incorrectly stored the clone in a static and thought it should be
// updated when it is later changed, you would be wrong: only the cache copy is
// updated; NOT OLD CLONES
System.out.println("CACHE_PET is" + CACHE_PET);
System.out.println("CLONE_PET is" + CLONE_PET);
The two System.out calls produce the following output:
```

```
CACHE_PET isPet type Cat named Fuzzy id:100
CLONE_PET isPet type Cat named Hairy id:100
```

### 115.16.2 How to Determine Whether or Not an Object Is the Cache Object

In [Section 114.3, "Modifying an Object"](#), it was noted that it is possible to read any particular instance of a class by executing:

```
session.readObject(Class);
```

There is also a `readObject` method that takes an object as an argument: this method is equivalent to doing a `ReadObjectQuery` on the primary key of the object passed in. For example, the following code is equivalent to the code in the subsequent example:

```
session.readObject(pet);
```

The following is equivalent to the preceding code:

```
ReadObjectQuery query = new ReadObjectQuery();
query.setReferenceClass(Pet.class);
```

```
ExpressionBuilder builder = new ExpressionBuilder();
Expression exp = builder.get("id").equal(pet.getId());
query.setSelectionCriteria(exp);
session.executeQuery(query);
```

Also note that primary key-based queries, by default, will return what is in the cache without going to the database. As a result, you can use very quick and simple method for accessing the cache copy of an object, as shown in [Example 115-14](#).

**Example 115-14 Testing If an Object Is the Cache Object**

```
//Here is a test to see if an object is the cache copy
boolean cached = CACHE_PET == session.readObject(CACHE_PET);
boolean cloned = CLONE_PET == session.readObject(CLONE_PET);
System.out.println("Is CACHE_PET the Cache copy of the object: " + cached);
System.out.println("Is CLONE_PET the Cache copy of the object: " + cloned);
```

This code produces the following output:

```
Is CACHE_PET the Cache copy of the object: true
Is CLONE_PET the Cache copy of the object: false
```

### 115.16.3 How to Dump the Contents of a Unit of Work

The unit of work has several debugging methods to help you analyze performance or track down problems with your code. The most useful is `printRegisteredObjects`, which prints all the information about known objects in the unit of work. Use this method to see how many objects are registered and to make sure objects you are working on are registered.

To use this method, you must have log messages enabled for the session that the unit of work is from. Session log messages are disabled by default. To enable log messages, use the session `logMessages` method. To disable log messages, use the session `dontLogMessages` method, as shown in [Example 115-15](#).

**Example 115-15 Dumping the Contents of a Unit of Work**

```
session.logMessages(); // Enable log messages
UnitOfWork uow = session.acquireUnitOfWork();
Pet petClone = (Pet)uow.readObject(Pet.class);
petClone.setName("Mop Top");
```

```
Pet pet2 = new Pet();
pet2.setId(200);
pet2.setName("Sparky");
pet2.setType("Dog");
uow.registerObject(pet2);
```

```
uow.printRegisteredObjects();
uow.commit();
session.dontLogMessages(); // Disable log messages
```

This example produces the following output:

```
UnitOfWork identity hashCode: 32373
Deleted Objects:
```

```
All Registered Clones:
  Key: [100] Identity Hash Code:13901 Object: Pet type Cat named Mop Top id:100
  Key: [200] Identity Hash Code:16010 Object: Pet type Dog named Sparky id:200
```

```
New Objects:
  Key: [200] Identity Hash Code:16010 Object: Pet type Dog named Sparky id:200
```

## 115.16.4 How to Handle Exceptions

This section explains how to handle the following:

- [Handling Exceptions at Commit Time](#)
- [Handling Exceptions During Conforming](#)

### 115.16.4.1 Handling Exceptions at Commit Time

TopLink exceptions are instances of `RuntimeException`, which means that methods that throw them do not have to be placed in a `try-catch` block.

However, the unit of work `commit` method is one that should be called within a `try-catch` block to deal with problems that may arise.

[Example 115–16](#) shows one way to handle unit of work exceptions:

#### **Example 115–16 Handling Unit of Work Commit Exceptions**

```
UnitOfWork uow = session.acquireUnitOfWork();
Pet petClone = (Pet)uow.registerObject(newPet);
petClone.setName("Assume this name is too long for a database constraint");
// Assume that the name argument violates a length constraint on the database.
// This will cause a DatabaseException on commit
try {
    uow.commit();
}
catch (TopLinkException tle) {
    System.out.println("There was an exception: " + tle);
}
```

This code produces the following output:

```
There was an exception: EXCEPTION [ORACLEAS TOPLINK-6004]:
oracle.toplink.exceptions.DatabaseException
```

If you use optimistic locking, you must catch exceptions at commit time because the exception raised is the indication that there was an optimistic locking problem.

Optimistic locking allows all users to access a given object, even if it is currently in use in a transaction or unit of work. When the unit of work attempts to change the object, the database checks to ensure that the object has not changed since it was initially read by the unit of work. If the object has changed, the database raises an exception, and the unit of work rolls back the transaction. For more information, see [Section 113.3.1.2, "Locking and the Unit of Work"](#).

If you are using an external transaction service, exceptions may be thrown long after your `UnitOfWork` code has returned. Using `UnitOfWork` method `writeChanges`, you can catch and handle most exceptions before the external transaction is committed. For more information, see [Section 115.13.4.2, "Handling External Transaction Commit Exceptions"](#).

### 115.16.4.2 Handling Exceptions During Conforming

You can conform query results in a unit of work across one-to-many relationships and a combination of both one-to-one and one-to-many relationships. [Example 115–17](#) illustrates a query across two levels of relationships, one-to-many and one-to-one.

#### **Example 115–17 Querying Across Two Levels of Relationship**

```
Expression exp =
    bldr.anyOf("managedEmployees").get("address").get("city").equal("Perth");
```

By default, any exceptions thrown during conforming are suppressed. However, you can use the `UnitOfWork` method `setShouldThrowConformExceptions` to make

the unit of work throw all conforming exceptions. This method takes one `int` argument with the following values:

- 0—do not throw conform exceptions (default)
- 1—throw all conform exceptions

For more information on customizing exception handling when using conforming and in-memory queries, see [Section 108.16.2.3, "Handling Exceptions Resulting from In-Memory Queries"](#).

## 115.16.5 How to Validate a Unit of Work

The unit of work validates object references at commit time. If an object registered in a unit of work references other unregistered objects, this violates object transaction isolation, and causes TopLink validation to raise an exception.

Although referencing unregistered objects from a registered object can corrupt the session cache, there are applications in which you want to disable validation. TopLink offers the following APIs to toggle validation:

- `dontPerformValidation`: disables validation
- `performFullValidation`: enables validation

### 115.16.5.1 Validating the Unit of Work Before Commit Time

If the unit of work detects an error when merging changes into the session cache, it throws a `QueryException`. Although this exception specifies the invalid object and the reason it is invalid, it may still be difficult to determine the cause of the problem.

In this case, you can use the `validateObjectSpace` method to test registered objects and provide the full stack trace of all traversed objects. This may help you more easily find the problem. You can call this method at any time on a unit of work.



# Part XXVI

---

## Creation and Configuration of Projects

This part describes the TopLink artifact used to contain mapping and data source-specific information. It contains the following chapters.

- [Chapter 116, "Creating a Project"](#)

This chapter contains procedures for creating TopLink projects.

- [Chapter 117, "Configuring a Project"](#)

This chapter explains how to configure TopLink project options common to two or more project types.



---

---

## Creating a Project

This chapter describes how to create TopLink projects.

This chapter includes the following sections:

- [Introduction to the Project Creation](#)
- [Working with Projects](#)
- [Exporting Project Information](#)

For information on the various types of projects available, see [Section 15.1, "TopLink Project Types"](#).

### 116.1 Introduction to the Project Creation

You can create a project using Oracle JDeveloper, TopLink Workbench, or Java code.

Oracle recommends using either Oracle JDeveloper or TopLink Workbench to create projects and generate deployment XML or Java source versions of the project for use at run time. For more information, see [Section 116.1.1, "How to Create a Project Using Oracle JDeveloper"](#) and [Section 116.1.2, "How to Create a Project Using TopLink Workbench"](#).

Alternatively, you can create projects in Java code. For an EIS project that uses a record type other than XML, you must use Java code. For more information, see [Section 116.1.3, "How to Create a Project Using Java"](#) and Oracle Fusion Middleware Java API Reference for Oracle TopLink.

For information on how to create a project using Java, see [Section 116.1.3, "How to Create a Project Using Java"](#).

#### 116.1.1 How to Create a Project Using Oracle JDeveloper

When you create a TopLink project using Oracle JDeveloper, your mapping information is stored in the TopLink map. The TopLink map contains the information about how classes map to database tables. Use the TopLink editor to edit each component of the mappings, including:

- Database information, such as driver, URL, and login information.
- Mapping defaults, such as identity map and cache options.

The TopLink editor in Oracle JDeveloper supports the following project types:



- Relational project



- XML project



- EIS project

## 116.1.2 How to Create a Project Using TopLink Workbench

When you create a project using TopLink Workbench, all project information is stored in the project file (.mwp file). This file references additional XML data files that contain the information about how the Java classes map to database tables or XML elements.

Using TopLink Workbench, you can export this information as a TopLink project XML file (that is, the deployment XML file) that is read in by the TopLink runtime. You can also export this information as a Java class. For more information, see [Section 116.3, "Exporting Project Information"](#).

TopLink Workbench displays projects and their contents in the Navigator window. When you select a project, its attributes are displayed in the Editor window. See [Section 5.3.3, "How to Use the Navigator"](#) for more information. TopLink Workbench supports the following project types:



- Relational project



- XML project



- EIS project

### 116.1.2.1 Creating New TopLink Workbench Projects

This section includes information on creating a new TopLink Workbench project. To create a new project from an existing persistence application, such as, for example, OC4J, see [Chapter 8, "Integrating TopLink with an Application Server"](#). To create a new project from JAXB, see [Section 47.1.1, "TopLink Support for Java Architecture for XML Binding \(JAXB\)"](#).

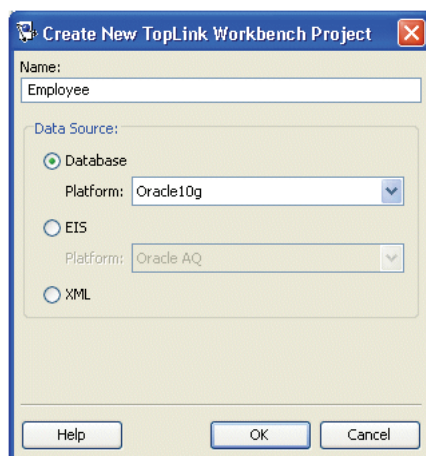
To create a new TopLink Workbench project, use this procedure:



1. Click **New** on the toolbar and select **Project**. The Create New TopLink Workbench Project dialog box appears.

You can also create a new project by choosing **File > New > Project** from the menu.

**Figure 116–1 Create New TopLink Workbench Project Dialog Box**



Use the following information to enter data in each field of this dialog box:

Field	Description
Name	Enter the name of the TopLink Workbench project. This project name will also become the name of the .mwp file.
Data Source	Use these options to specify the type of project to create, and its data source.
Database	Select <b>Database</b> to create an relational project to a relational database.  Use the <b>Platform</b> list to select the specific database platform.  See <a href="#">Chapter 18, "Introduction to Relational Projects"</a> for more information.
EIS	Select <b>EIS</b> to create an EIS project to a nonrelational data source using XML records.  Use the <b>Platform</b> list to specify the JCA adapter to use.  See <a href="#">Chapter 71, "Introduction to EIS Projects"</a> for more information.
XML	Select <b>XML</b> to create a nontransactional, nonpersistent XML project to an XML schema.  Alternatively, you can generate both an XML project and object model classes (see <a href="#">Section 48.2, "Creating an XML Project from an XML Schema"</a> ).  See <a href="#">Chapter 47, "Introduction to XML Projects"</a> for more information.

For more project information, continue with the following:

- Configure the project (see [Chapter 117, "Configuring a Project"](#)).
- Add mappings and descriptors (see [Chapter 16, "Introduction to Descriptors"](#) and [Chapter 17, "Introduction to Mappings"](#)).
- Export the project for use with the TopLink runtime (see [Section 116.3, "Exporting Project Information"](#)).

### 116.1.3 How to Create a Project Using Java

To create a project using Java code, use this procedure:

1. Implement a project class that extends the `oracle.toplink.sessions.Project` class (see [Example 116–1](#)).
2. Compile the project class.

#### **Example 116–1 Specifying a TopLink Project in Code**

```
/**
 * The class EmployeeProject is an example of an Oracle TopLink project defined in
 * Java code. The individual parts of the project - the Login and the descriptors,
 * are built inside of methods that are called by the constructor. Note that
 * EmployeeProject extends the class oracle.toplink.sessions.Project
 */
public class EmployeeProject extends oracle.toplink.sessions.Project {

/**
 * Supply a zero-argument constructor that initializes all aspects of the project.
 * Make sure that the login and all the descriptors are initialized and added to
 * the project. Project-level properties, such as the name of the project, should
 * be specified here
```

```
*/
public EmployeeProject() {
    setName("EmployeeProject");
    applyLogin();

    addDescriptor(buildAddressDescriptor());
    addDescriptor(buildEmployeeDescriptor());
    addDescriptor(buildPhoneNumberDescriptor());
}

// Data source information
public void applyLogin() {
    DatabaseLogin login = new DatabaseLogin();

    // use platform appropriate for underlying database
    login.usePlatform(
        new oracle.toplink.platform.database.oracle.Oracle9Platform());
    login.setDriverClassName("oracle.jdbc.OracleDriver");
    login.setConnectionString("jdbc:oracle:thin:@HOST:PORT:SID");
    login.setUserName("USER NAME");
    login.setEncryptedPassword("PASSWORD, ENCRYPTED");

    // Configuration Properties
    setDatasourceLogin(login);
}

/**
 * Descriptors are built by defining table info, setting properties
 * (caching, etc.) and by adding mappings to the descriptor
 */

// SECTION: DESCRIPTOR
public ClassDescriptor buildAddressDescriptor() {

    RelationalDescriptor descriptor = new RelationalDescriptor();

    // specify the class to be made persistent
    descriptor.setJavaClass(examples.servletjsp.model.Address.class);

    // specify the tables to be used and primary key
    descriptor.addTableName("ADDRESS");
    descriptor.addPrimaryKeyFieldName("ADDRESS.ADDRESS_ID");

    // Descriptor Properties
    descriptor.useSoftCacheWeakIdentityMap();
    descriptor.setIdentityMapSize(100)
    descriptor.useRemoteSoftCacheWeakIdentityMap()
    descriptor.setRemoteIdentityMapSize(100)
    descriptor.setSequenceNumberFieldName("ADDRESS.ADDRESS_ID")
    descriptor.setSequenceNumberName("ADD_SEQ");
    descriptor.setAlias("Address");

    // Mappings
    DirectToFieldMapping cityMapping = new DirectToFieldMapping();
    cityMapping.setAttributeName("city");
    cityMapping.setFieldName("ADDRESS.CITY");
    descriptor.addMapping(cityMapping);

    // Additional mappings are added to the descriptor using the addMapping method
```

```
return descriptor;
}
```

---



---

**Note:** Using TopLink Workbench provides a starting point for a custom project class. For more information, see [Section 19.6.1, "How to Export Project Java Source Using TopLink Workbench"](#).

---



---

## 116.2 Working with Projects

Using TopLink Workbench, you can perform the following project functions:

- [How to Open Existing Projects](#)
- [How to Save Projects](#)
- [How to Generate the Project Status Report](#)

See [Chapter 117, "Configuring a Project"](#) for additional information on working with TopLink Workbench projects.

### 116.2.1 How to Open Existing Projects

Use this procedure to open an existing project:

---



---

**Caution:** For most prior release projects, simply opening the project in TopLink Workbench will upgrade your project. However, to upgrade release 9.0.3 (and earlier) projects, you must follow specific upgrade procedures and use the Package Rename tool.

Refer to *Oracle TopLink Release Notes* for more information.

---



---



1. Click **Open Project** on the toolbar. The Choose a File dialog box appears. You can also open a project by choosing **File > Open** from the menu.

---



---

**Note:** The **File** menu option contains a list of recently opened projects. You can select one of these projects to open. See [Section 5.4.1, "How to Use General Preferences"](#) for information on customizing this list.

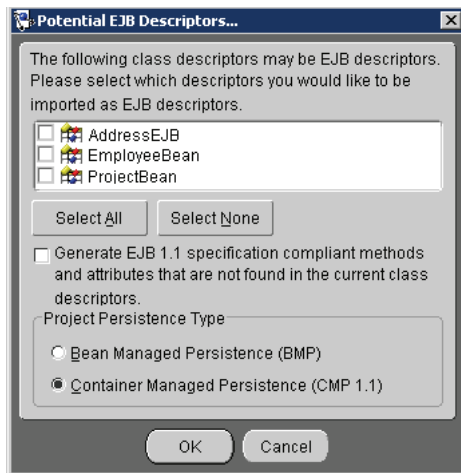
---



---

2. Select the TopLink Workbench project file ( `.mwp` ) to open, and click **Open**. TopLink Workbench displays the project information.

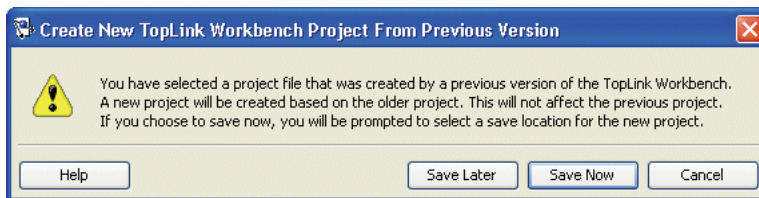
If you open a TopLink Workbench version *3.n* project that contains EJB information, the Potential EJB Descriptors dialog box appears.

**Figure 116–2 Potential EJB Descriptors Dialog Box**

3. Select which of the descriptors should be imported as EJB descriptors, the project persistence type, and click **OK**.

You can also specify whether or not TopLink Workbench generates methods and attributes that comply with the EJB specification, if they are not found within the current class descriptor(s).

If you open a TopLink Workbench version 9.0.3 (or later) project, the Create New TopLink Workbench Project from Previous Version dialog box appears.

**Figure 116–3 Create New TopLink Workbench Project From Previous Version Dialog Box**

To convert the old project to the current format and view the project immediately, click **Save Later**.

To convert the old project to the current format and save it to a new location and then view the project, click **Save Now**.

## 116.2.2 How to Save Projects

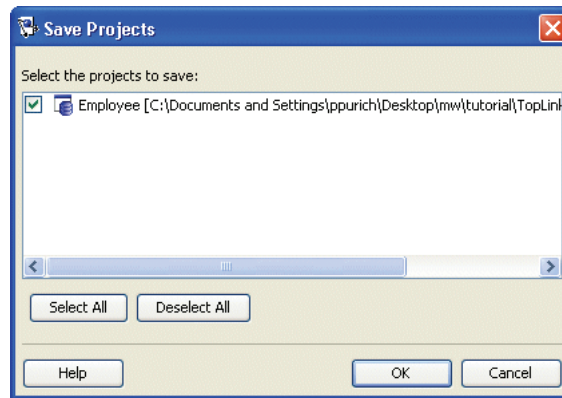
TopLink Workbench does not automatically save your project. Be sure to save your project often to avoid losing data.

To save your project(s), use this procedure:



1. Click **Save** or **Save All** to save your project(s).  
You can also save a project by choosing **File > Save** or **File > Save All** from the menu.
2. If you close TopLink Workbench while there are currently unsaved changes, the Save Project dialog box appears.



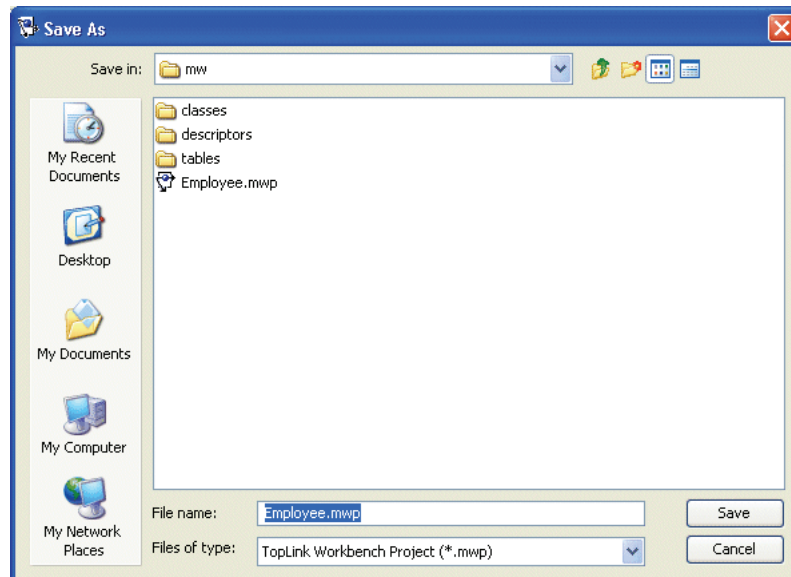
**Figure 116–4 Save Projects Dialog Box**

3. Select the project(s) to save and click **OK**.  
Click **Select All** to select all the available projects.

### 116.2.2.1 Saving Projects with a New Name or Location

To save your project with a different name or location, use this procedure:

1. Choose **File > Save As**. The Save As dialog box appears.

**Figure 116–5 Save As Dialog Box**

2. Select a name and location, then click **Save**.

---

**Caution:** Do not rename the `.mwp` file outside of TopLink Workbench. To rename a project, use the **Save As** option.

---

## 116.2.3 How to Generate the Project Status Report

Use the project status report to display a list of all warnings and errors in the TopLink Workbench project. This report is similar to the Problems window (see [Section 5.3](#),

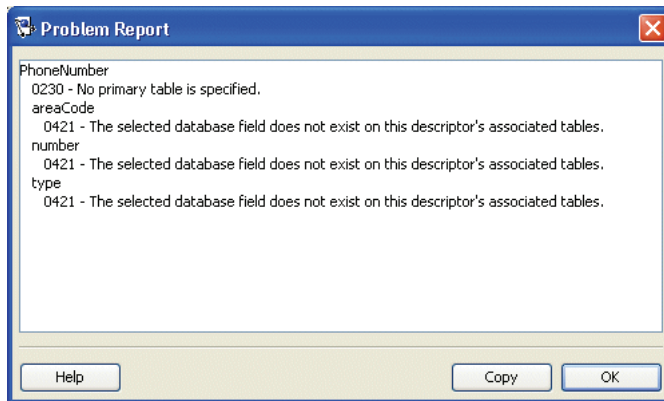


"Using TopLink Workbench"), but lets you easily copy and paste the errors into documents or messages. To generate the project status report, use this procedure:

1. Right-click the **Problems** label above the **Problems** window and select **Problem Report**. The Project Status Report dialog box appears, displaying the status of each TopLink Workbench project.

You can also generate the project status report by selecting **Tools > Problem Report** from the menu.

**Figure 116–6 Problem Report Dialog Box**



See [Section A.3, "TopLink Workbench Error Reference"](#) for information on each reported error.

To copy the report to another application, click **Copy**.

## 116.3 Exporting Project Information

To use your project with the TopLink Foundation Library at run time, you must either generate deployment XML or export the project to Java source code.

For all project types, TopLink Workbench can generate and export the following project information:

- Deployment information (see [Section 116.3.1, "How to Export Deployment XML Information Using TopLink Workbench"](#)) (`project.xml` file)
- Model Java source (see [Section 116.3.2, "How to Export Model Java Source Using TopLink Workbench"](#))

---

**Note:** When exporting Java source and deployment XML, TopLink Workbench writes the database password (if applicable) using Java Cryptography Extension (JCE) encryption. For information on how to specify password encryption options, see [Section 97.3, "Configuring Password Encryption"](#).

---

### 116.3.1 How to Export Deployment XML Information Using TopLink Workbench

To export your deployment XML file (`project.xml`), use this procedure (see [Chapter 9, "Creating TopLink Files for Deployment"](#) for detailed information):



1. Select the project and click **Export Deployment XML**.

You can also right-click the project in the **Navigator** and choose **Export > Project Deployment XML** from the context menu or choose **Selected > Export > Project Deployment XML** from the menu.

If you have not defined deployment and source code generation defaults (see [Chapter 117, "Configuring a Project"](#)) TopLink Workbench prompts for a file name and directory.

---

---

**Note:** If your project contains errors, the `project.xml` may not be valid. See [Section A.3, "TopLink Workbench Error Reference"](#) for information on each reported error.

---

---

To generate the deployment XML file that is compatible with projects prior to this release, see [Section 20.11, "Configuring Deprecated Direct Mappings"](#).

### 116.3.2 How to Export Model Java Source Using TopLink Workbench

To generate the project model's Java source code, use this procedure:

1. Right-click the project, package, or specific descriptor in the **Navigator** and choose **Export > Export Model Java Source** from the context menu. TopLink Workbench creates a `.java` file for each selected descriptor.

You can also choose **Workbench > Export > Export Model Java Source** or **Selected > Export > Model Java Source** from the menu or click **Generate Source Code** on the **Class** tab. See [Section 5.7.2.1, "Configuring Class Information"](#) for more information.

2. Click **Generate Source Code** to generate the project's model Java source.

If you have not defined deployment and source code generation defaults (see [Section 117, "Configuring a Project"](#)) TopLink Workbench prompts for a root directory.

---

---

**Note:** If your TopLink Workbench project uses UTF-8 character set, you must use a compatible JDK when compiling the exported Java source.

---

---



---



---

## Configuring a Project

This chapter describes how to configure TopLink project options common to two or more project types.

This chapter includes the following sections:

- [Configuring Common Project Options](#)
- [Configuring Project Save Location](#)
- [Configuring Project Classpath](#)
- [Configuring Method or Direct Field Access at the Project Level](#)
- [Configuring Persistence Type](#)
- [Configuring Default Descriptor Advanced Properties](#)
- [Configuring Existence Checking at the Project Level](#)
- [Configuring Project Deployment XML Options](#)
- [Configuring Model Java Source Code Options](#)
- [Configuring Cache Type and Size at the Project Level](#)
- [Configuring Cache Isolation at the Project Level](#)
- [Configuring Cache Coordination Change Propagation at the Project Level](#)
- [Configuring Cache Expiration at the Project Level](#)
- [Configuring Project Comments](#)

Table 117–1 lists the types of TopLink projects that you can configure and provides a cross-reference to the type-specific chapter that lists the configurable options supported by that type.

**Table 117–1** *Configuring TopLink Projects*

<b>If you are creating...</b>	<b>See also...</b>
Relational Projects	<a href="#">Chapter 20, "Configuring a Relational Project"</a>
EIS Projects	<a href="#">Chapter 73, "Configuring an EIS Project"</a>
XML Projects	<a href="#">Chapter 49, "Configuring an XML Project"</a>

Table 117–2 lists the configurable options shared by two or more TopLink project types.

For more information, see the following:

- [Section 116.1, "Introduction to the Project Creation"](#)

- [Chapter 15, "Introduction to Projects"](#)

## 117.1 Configuring Common Project Options

Table 117–2 lists the configurable options shared by two or more TopLink project types. In addition to the configurable options described here, you must also configure the options described for the specific TopLink project types (see Section 15.1, "TopLink Project Types"), as shown in Table 117–1.

**Table 117–2 Common Project Options**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Save location (see Section 117.2, "Configuring Project Save Location")		✓	
Classpath (see Section 117.3, "Configuring Project Classpath")		✓	
Method or direct field access (see Section 117.4, "Configuring Method or Direct Field Access at the Project Level")	✓	✓	
Persistence type (see Section 117.5, "Configuring Persistence Type")	✓	✓	
Default descriptor advanced properties (see Section 117.6, "Configuring Default Descriptor Advanced Properties")	✓	✓	
Existence checking (see Section 117.7, "Configuring Existence Checking at the Project Level")	✓	✓	✓
Deployment XML options (see Section 117.8, "Configuring Project Deployment XML Options")		✓	
Model Java source code options (see Section 117.9, "Configuring Model Java Source Code Options")	✓	✓	
Deprecated direct mappings (see Section 20.11, "Configuring Deprecated Direct Mappings")	✓	✓	
Cache type and size (see Section 117.10, "Configuring Cache Type and Size at the Project Level")	✓	✓	✓
Cache isolation (see Section 117.11, "Configuring Cache Isolation at the Project Level")	✓	✓	✓
Cache coordination change propagation (see Section 117.12, "Configuring Cache Coordination Change Propagation at the Project Level")	✓	✓	✓
Cache expiration (see Section 117.13, "Configuring Cache Expiration at the Project Level")	✓	✓	✓
Comments (see Section 117.14, "Configuring Project Comments")	✓	✓	

## 117.2 Configuring Project Save Location

You can configure a project save location only when using TopLink Workbench.

Table 117–3 summarizes which projects support a project save location.

**Table 117–3 Project Support for Project Save Location**

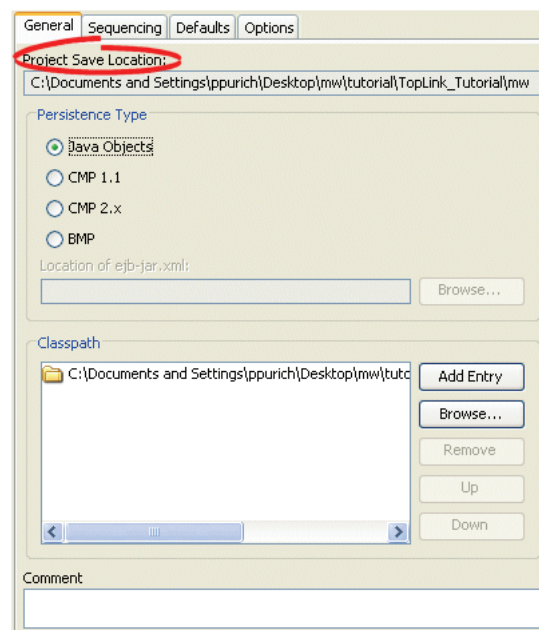
Descriptor	How to Use Oracle JDeveloper	How to Configure Project Save Location Using TopLink Workbench	How to Use Java
Relational Projects		✓	

**Table 117–3 (Cont.) Project Support for Project Save Location**

Descriptor	How to Use Oracle JDeveloper	How to Configure Project Save Location Using TopLink Workbench	How to Use Java
EIS Projects		✓	
XML Projects		✓	

### 117.2.1 How to Configure Project Save Location Using TopLink Workbench

The Project Save Location field on the project's General tab is for display only. This field shows the full directory path for the project. All relative locations used in the project are based on this location.

**Figure 117–1 General Tab, Project Save Location**

To select a new location, right-click on the project in the **Navigator** and select **Save As** from the context menu. See [Section 116.2.2, "How to Save Projects"](#) for more information.

## 117.3 Configuring Project Classpath

The TopLink project uses a classpath—a set of directories, JAR files, and ZIP files—when importing Java classes and defining object types.

[Table 117–4](#) summarizes which projects support project classpath configuration.

**Table 117–4 Project Support for Project Classpath**

Descriptor	How to Use Oracle JDeveloper	How to Configure Project Classpath Using TopLink Workbench	How to Use Java
Relational Projects		✓	

**Table 117-4 (Cont.) Project Support for Project Classpath**

Descriptor	How to Use Oracle JDeveloper	How to Configure Project Classpath Using TopLink Workbench	How to Use Java
EIS Projects		✓	
XML Projects		✓	

Do not include JDBC drivers or other elements required to access the data source in the project classpath. Use the `setenv` file to specify these application-level settings (see Section 5.2, "Configuring the TopLink Workbench Environment").

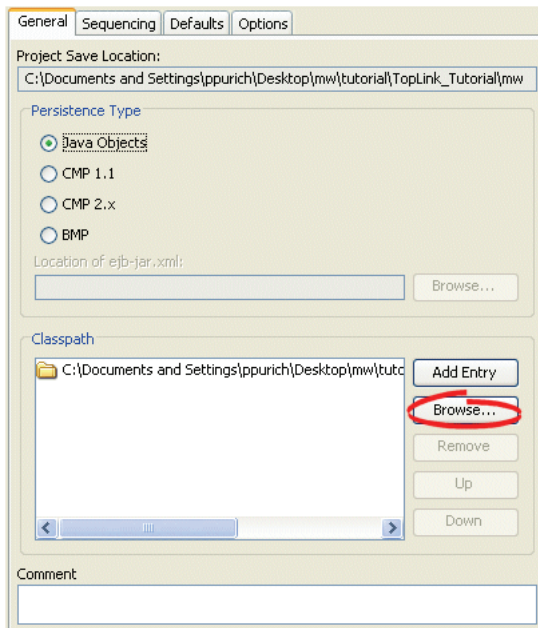
After you configure the project classpath, you can use TopLink Workbench to import classes into your project (see Section 5.7.3, "How to Import and Update Classes").

### 117.3.1 How to Configure Project Classpath Using TopLink Workbench

To specify the project classpath information, use this procedure:

1. Select the project object in the **Navigator**.
2. Click the **General** tab in the **Editor**. The General tab appears.

**Figure 117-2 General Tab, Classpath Options**



To add a new classpath entry, click **Add Entry** or **Browse** and select the directory, `.jar` file, or `.zip` file for this project. To create a relative classpath, select an entry and edit the path, as necessary. The path will be relative to the **Project Save Location**.

To remove a classpath entry, select the entry and click **Remove**.

To change the order of the entries, select the entry and click **Up** or **Down**.



## 117.4 Configuring Method or Direct Field Access at the Project Level

By default, when TopLink performs a persistence operation, it accesses the persistent attributes of an object directly: this is known as direct field access. Alternatively, you can configure TopLink to access persistent attributes using accessor methods of the object: this is known as method access.

Oracle recommends using field access for mappings. Not only is it more efficient, but using method access may cause issues if the method produces unexpected side-effects.

Table 117-5 summarizes which projects support mapped field access configuration.

**Table 117-5 Project Support for Mapped Field Access**

Descriptor	How to Use Oracle JDeveloper	How to Configure Method or Direct Field Access at the Project Level Using TopLink Workbench	How to Use Java
Relational Projects	✓	✓	
EIS Projects	✓	✓	
XML Projects	✓	✓	

This section describes configuring mapped field access at the project level: by default, this configuration applies to all descriptors and their mappings.

---

**Note:** If you change the access default, existing mappings retain their current access settings, but new mappings will be created with the new default.

---

You can also configure mapped field access at the mapping level to override this project-level configuration on a mapping-by-mapping basis. For more information, see [Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"](#).

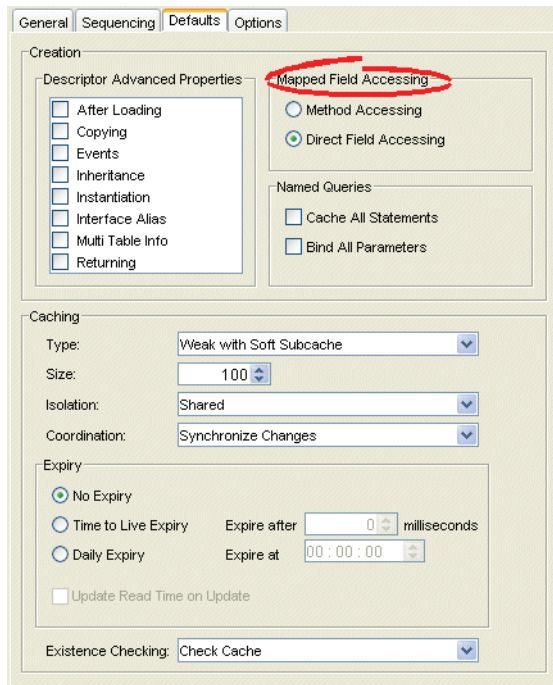
If you enable change tracking on a property (for example, you decorate method `getPhone` with `@ChangeTracking`) and you access the field (`phone`) directly, note that TopLink does not detect the change. For more information, see [Section 2.4.1.4, "Using Method and Direct Field Access"](#).

### 117.4.1 How to Configure Method or Direct Field Access at the Project Level Using TopLink Workbench

To specify the field access method information, use this procedure:

1. Select the project object in the **Navigator**.
2. Select the **Defaults** tab in the **Editor**. The Defaults tab appears.

**Figure 117-3 Defaults Tab, Field Accessing Options**



## 117.5 Configuring Persistence Type

You can configure your project persistence type when using Oracle JDeveloper or TopLink Workbench.

Using TopLink Workbench, you can specify the persistence type to use with the project. For example, your TopLink project may use plain Java objects, entity beans with container-managed persistence, or entity beans with bean-managed persistence (BMP).

Table 117-6 summarizes which projects support a persistence type configuration.

**Table 117-6 Project Support for Persistence Type**

Descriptor	How to Use Oracle JDeveloper	How to Configure Persistence Type Using TopLink Workbench	How to Use Java
Relational Projects	✓	✓	
EIS Projects	✓	✓	
XML Projects			

To create a TopLink descriptor for a persistent class, TopLink Workbench reads a compiled Java .class file to read its attributes and relationships. See Section 14.1.2, "Descriptors" for more information on TopLink descriptors.

For EJB projects, you can specify an ejb-jar.xml file from which TopLink will read and to which it will write the necessary persistence information. You use the ejb-jar.xml file to map the virtual fields of the entity beans with container-managed persistence (called container-managed fields, defined by <cmp-field> tag) or relationships (called container-managed relationship, defined by <cmr-field> tag) to a data source.



TopLink Workbench defines all descriptors for entity classes (as defined in the `ejb-jar.xml` file) as EJB descriptors. TopLink Workbench does not create (or remove) descriptors for the interfaces and primary key class for the entity when refreshing from the `ejb-jar.xml` file.

---

**Note:** TopLink Workbench creates class descriptors for entity classes not defined in the `ejb-jar.xml` file. You must manually change the descriptor type (see [Section 119.18, "Configuring a Descriptor with EJB CMP and BMP Information"](#)).

---

To update your project from the XML file, right-click an EJB descriptor and select **Update Descriptors from ejb-jar.xml**. You can also update the project by choosing **Selected > Update Descriptors from ebj-jar.xml** from the menu.

For more information on creating and using deployment files such as the `ejb-jar.xml` file, see the following:

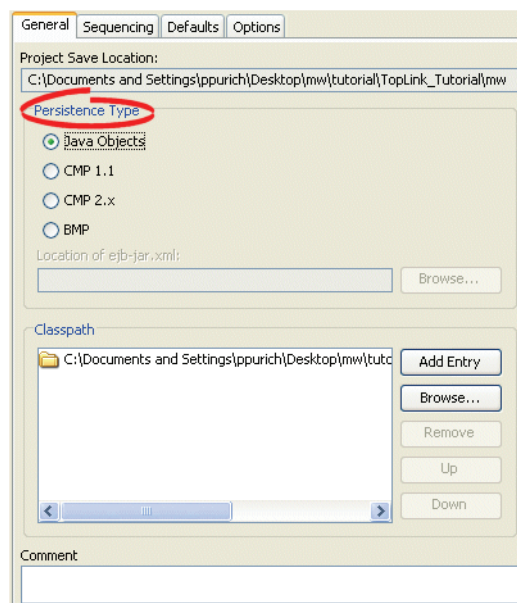
- [Section 9.1, "Introduction to the TopLink Deployment File Creation"](#)
- [Chapter 8, "Integrating TopLink with an Application Server"](#)
- [Chapter 10, "Packaging a TopLink Application"](#)
- [Chapter 11, "Deploying a TopLink Application"](#)

## 117.5.1 How to Configure Persistence Type Using TopLink Workbench

To specify the persistence information, use this procedure:

1. Select the project object in the **Navigator**.
2. Select the **General** tab in the **Editor**. The General tab appears.

**Figure 117-4 General Tab, Persistence Type Options**



Use this table to enter data in the following fields on the project's **General** tab to configure the persistence options:

Field	Description
Persistence Type	Specify the persistence type of the project: <b>Java Objects</b> , <b>CMP 1.1</b> , <b>CMP 2.x</b> , or <b>BMP</b> . For EJB projects, specify the location of the <code>ejb-jar.xml</code> file. Note: This field does not apply to XML projects.
Location of <code>ejb-jar.xml</code>	Specify the location of the <code>ejb-jar.xml</code> file for this project. <a href="#">Section 19.7, "Working with the ejb-xml.File"</a> for more information. Note: This field applies to EJB projects only.

## 117.6 Configuring Default Descriptor Advanced Properties

You can configure default descriptor advanced properties when using Oracle JDeveloper or TopLink Workbench.

By default, TopLink Workbench displays a subset of features for each descriptor type. You can modify this subset so that descriptors include additional advanced properties by default.

You can also select specific advanced properties for individual descriptors (see [Chapter 119, "Configuring a Descriptor"](#)).

[Table 117-7](#) summarizes which projects support default descriptor advanced property configuration.

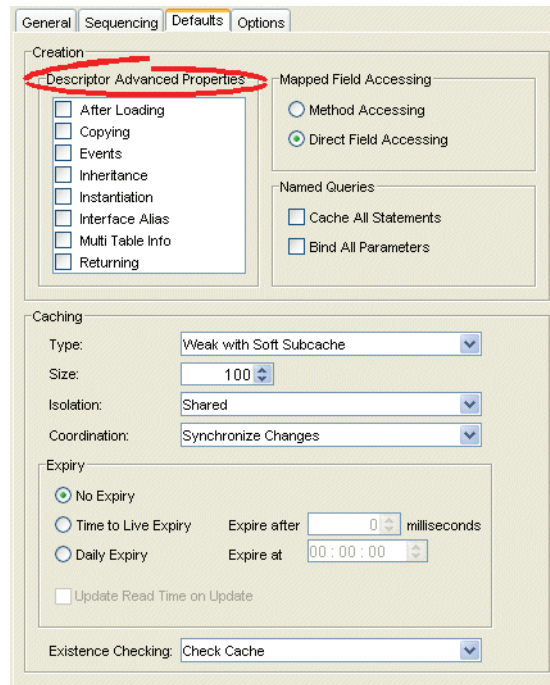
**Table 117-7 Project Support for Default Descriptor Advanced Properties**

Descriptor	How to Use Oracle JDeveloper	<a href="#">How to Configure Default Descriptor Advanced Properties Using TopLink Workbench</a>	How to Use Java
Relational Projects	✓	✓	
EIS Projects	✓	✓	
XML Projects	✓	✓	

### 117.6.1 How to Configure Default Descriptor Advanced Properties Using TopLink Workbench

To specify the default advanced properties for newly created descriptors in your project, use this procedure:

1. Select the project object in the **Navigator**.
2. Select the **Defaults** tab in the **Editor**. The Defaults tab appears.

**Figure 117-5 Defaults Tab, Descriptor Advanced Properties**

Select which **Descriptor Advanced Properties** to add to newly created descriptors. The list of advanced properties will vary, depending on the project type.

## 117.7 Configuring Existence Checking at the Project Level

When TopLink writes an object to the database, it runs an existence check to determine whether to perform an insert or an update operation.

By default, TopLink checks against the cache. Oracle recommends that you use this default existence check option for most applications. Checking the database for existence can cause a performance bottleneck in your application.

Table 117-8 summarizes which projects support existence checking configuration.

**Table 117-8 Project Support for Existence Checking**

Descriptor	How to Use Oracle JDeveloper	How to Configure Existence Checking at the Project Level Using TopLink Workbench	How to Use Java
Relational Projects	✓	✓	✓
EIS Projects	✓	✓	✓
XML Projects			

By default, this configuration applies to all descriptors in a project. You can also configure existence checking at the descriptor level to override this project-level configuration on a descriptor-by-descriptor basis. For more information, see [Section 119.17, "Configuring Cache Existence Checking at the Descriptor Level"](#).

For more information see the following:

- [Section 102.2.1, "Cache Type and Object Identity"](#)

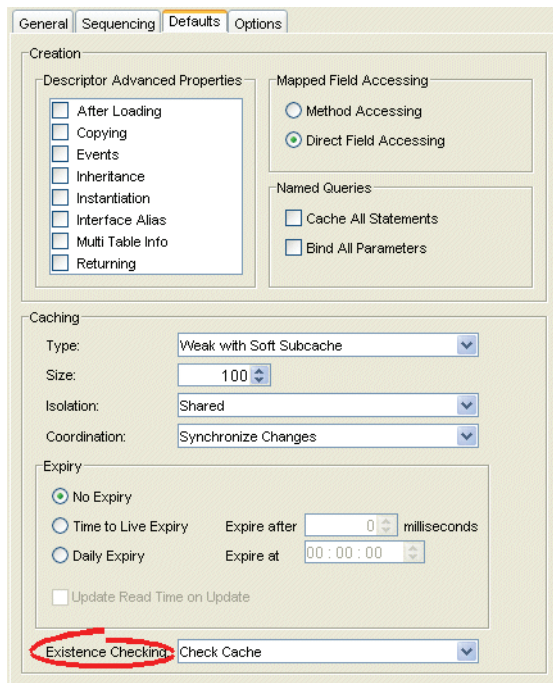
- [Section 108.16, "Queries and the Cache"](#)
- [Section 115.1.3, "How to Use Registration and Existence Checking"](#)

### 117.7.1 How to Configure Existence Checking at the Project Level Using TopLink Workbench

To specify the existence checking information, use this procedure:

1. Select the project object in the **Navigator**.
2. Select the **Defaults** tab in the **Editor**. The Defaults tab appears.

**Figure 117–6 Defaults Tab, Existence Checking Options**



Use this table to enter data in following fields to specify the existence checking options for newly created descriptors:

Field	Description
<b>Check Cache</b>	Check the session cache. If the object is not in the cache, assume that the object does not exist (do an insert). If the object is in the cache, assume that the object exists (do an update).  Oracle recommends using this option for most applications.
<b>Check Database</b>	If an object is not in the cache, query the database to determine if the object exists. If the object exists, do an update. Otherwise, do an insert.  Selecting this option may negatively impact performance. For more information, see <a href="#">Section 115.1.3.1, "Using Check Database"</a> .
<b>Assume Existence</b>	Always assume objects exist: always do an update (never do an insert).  For more information, see <a href="#">Section 115.1.3.2, "Using Assume Existence"</a> .

Field	Description
<b>Assume Nonexistence</b>	Always assume objects do not exist: always do an insert (never do an update).  For more information, see <a href="#">Section 115.1.3.3, "Using Assume Nonexistence"</a> .

## 117.8 Configuring Project Deployment XML Options

You can configure project deployment XML options when using TopLink Workbench.

Using TopLink Workbench, you can specify the default file names, class names, and directories, when exporting or generating deployment XML. Directories are relative to the project save location (see [Section 117.2, "Configuring Project Save Location"](#)), and will contain folders for each generated package.

[Table 117-9](#) summarizes which projects support deployment XML options.

**Table 117-9 Project Support for Project Deployment XML Options**

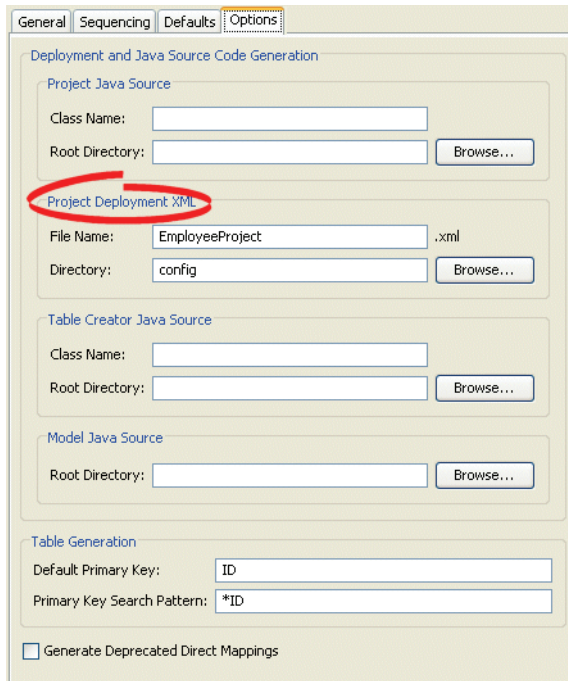
Descriptor	How to Use Oracle JDeveloper	How to Configure Project Deployment XML Options Using TopLink Workbench	How to Use Java
Relational Projects		✓	
EIS Projects		✓	
XML Projects		✓	

### 117.8.1 How to Configure Project Deployment XML Options Using TopLink Workbench

To specify the default export options, use this procedure:

1. Select the project object in the **Navigator**.
2. Select the **Options** tab in the **Editor**. The Options tab appears.

**Figure 117–7 Options Tab, Project Deployment XML Options**



Use this table to enter data in following fields to specify the default Project Deployment XML options:

Field	Description
File Name	File name (such as <code>project.xml</code> ) to use when generating project deployment XML.
Directory	Directory in which to save the generated deployment XML file.

## 117.9 Configuring Model Java Source Code Options

You can configure model java source code options when using Oracle JDeveloper or TopLink Workbench.

Using TopLink Workbench, you can specify the default file names, class names, and directories, when exporting or generating Java source code for your domain objects. Directories are relative to the project save location (see [Section 117.2, "Configuring Project Save Location"](#)), and will contain folders for each generated package.

[Table 117–10](#) summarizes which projects support model Java source code options.

**Table 117–10 Project Support for Model Java Source Options**

Descriptor	How to Use Oracle JDeveloper	How to Configure Model Java Source Code Options Using TopLink Workbench	How to Use Java
Relational Projects	✓	✓	
EIS Projects	✓	✓	
XML Projects	✓	✓	



## 117.9.1 How to Configure Model Java Source Code Options Using TopLink Workbench

To specify the default export options, use this procedure:

1. Select the project object in the **Navigator**.
2. Select the **Options** tab in the **Editor**. The Options tab appears.

**Figure 117–8 Options Tab, Model Java Source options**

The screenshot shows the 'Options' tab in the TopLink Workbench. The 'Model Java Source' section is highlighted with a red circle. The 'Project Java Source' section includes fields for 'Class Name' and 'Root Directory' with a 'Browse...' button. The 'Project Deployment XML' section includes fields for 'File Name' (set to 'EmployeeProject.xml') and 'Directory' (set to 'config') with a 'Browse...' button. The 'Table Creator Java Source' section includes fields for 'Class Name' and 'Root Directory' with a 'Browse...' button. The 'Table Generation' section includes fields for 'Default Primary Key' (set to 'ID') and 'Primary Key Search Pattern' (set to '\*ID'). There is also a checkbox for 'Generate Deprecated Direct Mappings' which is unchecked.

Specify the project root directory to which TopLink Workbench generates model Java source files. For more information, see [Section 118.3, "Generating Java Code for Descriptors"](#).

## 117.10 Configuring Cache Type and Size at the Project Level

The TopLink cache is an in-memory repository that stores recently read or written objects based on class and primary key values. TopLink uses the cache to achieve the following:

- improve performance by holding recently read or written objects and accessing them in-memory to minimize database access;
- manage locking and isolation level;
- manage object identity.

[Table 117–11](#) summarizes which projects support identity map configuration.

**Table 117–11 Project Support for Identity Map Configuration**

<b>Descriptor</b>	<b>How to Use Oracle JDeveloper</b>	<b>How to Configure Cache Type and Size at the Project Level Using TopLink Workbench</b>	<b>How to Configure Cache Type and Size at the Project Level Using Java</b>
Relational Projects	✓	✓	✓
EIS Projects	✓	✓	✓
XML Projects			

The cache options you configure at the project level apply globally to all descriptors. Use this section to define global cache options for a TopLink project.

You can override the project-level identity map configuration by defining identity map configuration at the descriptor level. For information on caching and defining identity map configuration for a specific descriptor, see [Section 119.12, "Configuring Cache Type and Size at the Descriptor Level"](#).

---

**Note:** When using TopLink Workbench, changing the project's default identity map does not affect descriptors that already exist in the project; only newly added descriptors are affected.

---

For detailed information on caching and object identity, and the recommended settings to maximize TopLink performance, see to [Section 102.2.1, "Cache Type and Object Identity"](#).

For more information about the cache, see [Chapter 102, "Introduction to Cache"](#).

### 117.10.1 How to Configure Cache Type and Size at the Project Level Using TopLink Workbench

To specify the cache identity map, use this procedure:

1. Select the project object in the **Navigator**.
2. Select the **Defaults** tab in the **Editor**. The Defaults tab appears.

**Figure 117–9 Defaults Tab, Cache Identity Map Options**

The screenshot shows the 'Defaults' tab in an IDE, specifically the 'Cache Identity Map Options' section. The 'Caching' section is highlighted with red circles around the 'Type' and 'Size' fields. The 'Type' dropdown is set to 'Weak with Soft Subcache' and the 'Size' spinner is set to '100'. Other options include 'Isolation: Shared', 'Coordination: Synchronize Changes', 'Expiry: No Expiry', and 'Existence Checking: Check Cache'.

Use this table to enter data in each of the following fields to specify the caching options:

Field	Description
Type	<p>Use the <b>Type</b> list to choose the identity map as follows:</p> <ul style="list-style-type: none"> <li> <b>Weak with Soft Subcache</b>            (<code>SoftCacheWeakIdentityMap</code>)—cache first <i>n</i> elements in soft space, anything after that in weak space (see <a href="#">Section 102.2.1.4</a>, "Soft Cache Weak Identity Map and Hard Cache Weak Identity Map").         </li> <li> <b>Weak with Hard Subcache</b>            (<code>HardCacheWeakIdentityMap</code>)—cache first <i>n</i> elements in soft space, anything after that in hard space (see <a href="#">Section 102.2.1.4</a>, "Soft Cache Weak Identity Map and Hard Cache Weak Identity Map").         </li> <li> <b>Weak</b> (<code>WeakIdentityMap</code>)—cache everything in weak space (see <a href="#">Section 102.2.1.2</a>, "Weak Identity Map").         </li> <li> <b>Soft</b> (<code>SoftIdentityMap</code>)—cache everything in soft space (see <a href="#">Section 102.2.1.3</a>, "Soft Identity Map").         </li> <li> <b>Full</b> (<code>FullIdentityMap</code>)—cache everything permanently (see <a href="#">Section 102.2.1.1</a>, "Full Identity Map").         </li> <li> <b>None</b> (<code>NoIdentityMap</code>)—cache nothing (see <a href="#">Section 102.2.1.5</a>, "No Identity Map").         </li> </ul> <p>For more information, see <a href="#">Section 102.2.1</a>, "Cache Type and Object Identity".</p> <p>Changing the project's default identity map does not affect descriptors that already exist in the project.</p>

Field	Description
Size	<p>Specify the size of the cache as follows:</p> <ul style="list-style-type: none"> <li>When using <b>Weak with Soft Subcache</b> or <b>Weak with Hard Subcache</b>, the size is the <i>maximum</i> number of objects stored in the identity map.</li> <li>When using <b>Full</b> or <b>Weak</b>, the size indicates the <i>starting size</i> of the identity map.</li> </ul>

### 117.10.2 How to Configure Cache Type and Size at the Project Level Using Java

Use one of the following `ClassDescriptor` methods to configure the descriptor to use the appropriate type of identity map:

- `useFullIdentityMap`
- `useWeakIdentityMap`
- `useSoftIdentityMap`
- `useSoftCacheWeakIdentityMap`
- `useHardCacheWeakIdentityMap`
- `useNoIdentityMap`

Use the `ClassDescriptor` method `setIdentityMapSize` to configure the size of the identity map.

## 117.11 Configuring Cache Isolation at the Project Level

If you plan to use isolated sessions (see [Section 102.2.7, "Cache Isolation"](#)), you must configure descriptors as isolated for any object that you want confined to an isolated session cache.

Configuring a descriptor to be isolated means that TopLink will not store the object in the shared session cache and the object will not be shared across client sessions. This means that each client will have their own object read directly from the database. Objects in an isolated client session cache can reference objects in their parent server session's shared session cache, but no objects in the shared session cache can reference objects in an isolated client session cache. Isolation is required when using Oracle Database Virtual Private Database (VPD) support or database user-based read security. Isolation can also be used if caching is not desired across client sessions.

[Table 117–11](#) summarizes which projects support cache isolation configuration.

**Table 117–12 Project Support for Cache Isolation Configuration**

Descriptor	How to Use Oracle JDeveloper	How to Configure Cache Isolation at the Project Level Using TopLink Workbench	How to Use Java
Relational Projects	✓	✓	✓
EIS Projects	✓	✓	✓
XML Projects			

The cache isolation options you configure at the project level apply globally to all descriptors. Use this section to define global options for a TopLink project.

You can override the project-level configuration by defining cache isolation options at the descriptor level. For information, see [Section 119.13, "Configuring Cache Isolation at the Descriptor Level"](#).

---

**Note:** When using TopLink Workbench, changing the project's default cache isolation option does not affect descriptors that already exist in the project; only newly added descriptors are affected.

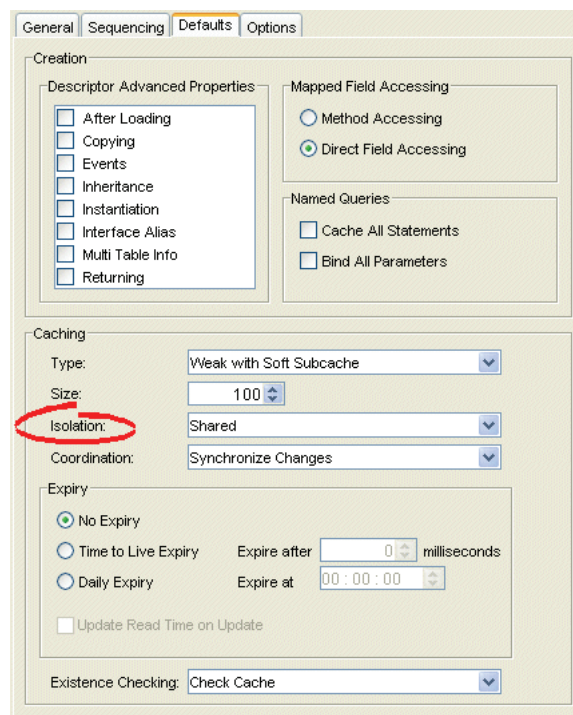
---

### 117.11.1 How to Configure Cache Isolation at the Project Level Using TopLink Workbench

To specify the cache isolation options, use this procedure:

1. Select the project object in the **Navigator**.
2. Select the **Defaults** tab in the **Editor**. The Defaults tab appears.

**Figure 117–10 Defaults Tab, Cache Isolation Options**



Use the **Isolation** list to choose one of the following:

- **Isolated**—if you want all objects confined to an isolated client session cache. For more information, see [Section 102.2.7, "Cache Isolation"](#).
- **Shared**—if you want all objects visible in the shared session cache (default).

### 117.12 Configuring Cache Coordination Change Propagation at the Project Level

If you plan to use a coordinated cache (see [Section 102.3, "Cache Coordination"](#)), you can configure how and under what conditions a coordinated cache propagates changes.

Table 117–11 summarizes which projects support cache coordination change propagation configuration.

**Table 117–13 Project Support for Cache Coordination Change Propagation Configuration**

Descriptor	How to Use Oracle JDeveloper	How to Configure Cache Coordination Change Propagation at the Project Level Using TopLink Workbench	How to Use Java
Relational Projects	✓	✓	✓
EIS Projects	✓	✓	✓
XML Projects			

The cache coordination change propagation options you configure at the project level apply globally to all descriptors. Use this section to define global options for a TopLink project.

You can override the project-level configuration by defining cache coordination change propagation options at the descriptor level. For information, see [Section 119.15, "Configuring Cache Coordination Change Propagation at the Descriptor Level"](#).

To complete your coordinated cache configuration, see [Chapter 103, "Configuring a Coordinated Cache"](#).

---



---

**Note:** When using TopLink Workbench, changing the project's default cache coordination change propagation option does not affect descriptors that already exist in the project; only newly added descriptors are affected.

---



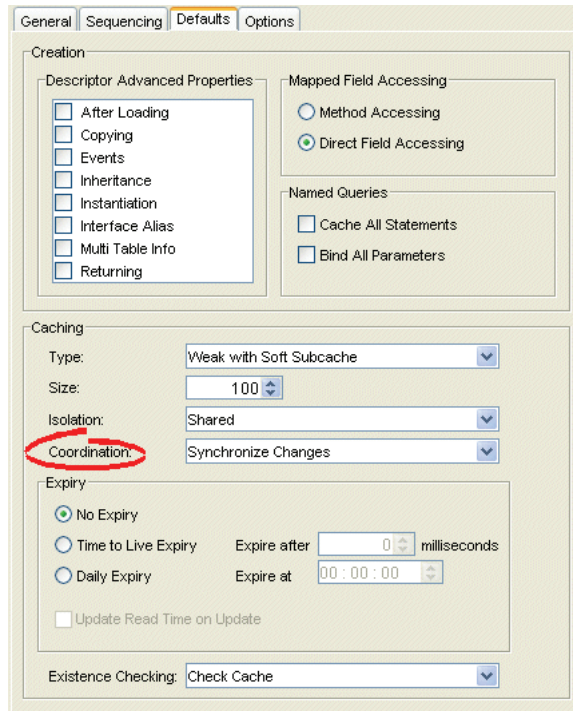
---

### 117.12.1 How to Configure Cache Coordination Change Propagation at the Project Level Using TopLink Workbench

To specify the coordinated cache change propagation options, use this procedure:

1. Select the project object in the **Navigator**.
2. Select the **Defaults** tab in the **Editor**. The Defaults tab appears.

**Figure 117–11 Defaults Tab, Coordination Options**



Use the following information to enter data in the Coordination field:

Coordination Option	Description	When to Use
<b>None</b>	For both existing and new instances, do not propagate a change notification.	Infrequently read or changed objects.
<b>Synchronize Changes</b>	For an existing instance, propagate a change notification that contains each changed attribute.  For a new instance, propagate an object creation (along with all the new instance’s attributes) only if the new instance is related to other existing objects that are also configured with this change propagation option.	Frequently read or changed objects that contain few attributes or in cases where only a few attributes are frequently changed.  Objects that have many or complex relationships.
<b>Synchronize Changes and New Objects</b>	For an existing instance, propagate a change notification that contains each changed attribute.  For a new instance, propagate an object creation (along with all the new instance’s attributes).	Frequently read or changed objects that contain few attributes or in cases where only a few attributes are frequently changed.  Objects that have few or simple relationships.
<b>Invalidate Changed Objects</b>	For an existing instance, propagate an object invalidation that marks the object as invalid in all other sessions. This tells other sessions that they must update their cache from the data source the next time this object is read.  For a new instance, no change notification is propagated.	Frequently read or changed objects that contain many attributes in cases where many of the attributes are frequently changed.

## 117.13 Configuring Cache Expiration at the Project Level

By default, objects remain in the cache until they are explicitly deleted (see [Section 114.7, "Deleting Objects"](#)) or garbage-collected when using a weak identity map (see [Section 117.10, "Configuring Cache Type and Size at the Project Level"](#)). Alternatively, you can configure an object with a `CacheInvalidationPolicy` that lets you specify, either automatically or manually, that an object is invalid: when any query attempts to read an invalid object, TopLink will go to the data source for the most up-to-date version of that object and update the cache with this information.

Using cache invalidation ensures that your application does not use stale data. It provides a better performing alternative to refreshing (see [Section 119.9, "Configuring Cache Refreshing"](#)).

[Table 117-14](#) summarizes which projects support cache invalidation configuration.

**Table 117-14 Project Support for Cache Invalidation Configuration**

Descriptor	How to Use Oracle JDeveloper	How to Configure Cache Expiration at the Project Level Using TopLink Workbench	How to Use Java
Relational Projects	✓	✓	✓
EIS Projects	✓	✓	✓
XML Projects			

The cache invalidation options you configure at the project level apply globally to all descriptors. Use this section to define global cache invalidation options for a TopLink project.

You can override the project-level cache invalidation configuration by defining cache invalidation at the descriptor (see [Section 119.16, "Configuring Cache Expiration at the Descriptor Level"](#)) or query level (see [Section 111.13.2, "How to Configure Cache Expiration at the Query Level"](#)).

You can customize how TopLink communicates the fact that an object has been declared invalid to improve efficiency if you are using a coordinated cache. For more information, see [Section 119.15, "Configuring Cache Coordination Change Propagation at the Descriptor Level"](#).

---



---

**Note:** When using TopLink Workbench, changing the project's default cache invalidation does not affect descriptors that already exist in the project; only newly added descriptors are affected.

---



---

For more information, see [Section 102.2.5, "Cache Invalidation"](#).

### 117.13.1 How to Configure Cache Expiration at the Project Level Using TopLink Workbench

To specify the cache expiration options for the project, use this procedure:

1. Select the project object in the **Navigator**.
2. Select the **Defaults** tab in the **Editor**. The Defaults tab appears.



**Figure 117–12 Defaults Tab, Cache Expiry Options**

Use this table to enter data in the following fields on this tab:

Field	Description
<b>No Expiry</b>	Specify that objects in the cache do not expire.
<b>Time to Live Expiry</b>	Specify that objects in the cache will expire after a specified amount of time. Use the <b>Expire After</b> field to indicate the time (in milliseconds) after which the objects will expire.
<b>Daily Expiry</b>	Specify that objects in the cache will expire at a specific time each day. Use the <b>Expire At</b> field to indicate the exact time to the second (using a 24-hour clock) at which the objects will expire.
<b>Update Read Time on Update</b>	Specify if the expiry time should be reset after updating an object.

---

**Note:** These options apply to all descriptors in a project. See [Section 119.16, "Configuring Cache Expiration at the Descriptor Level"](#) for information on configuring descriptor-specific options.

---

## 117.14 Configuring Project Comments

You can define a free-form textual comment for each project. You can use these comments however you wish: for example, to record important project implementation details such as the purpose or importance of a project.

In a Oracle JDeveloper or TopLink Workbench project, the comments are stored in the TopLink deployment XML file. There is no Java API for this feature.

[Table 117–15](#) summarizes which projects support this option.

**Table 117–15 Project Support for Project Comments**

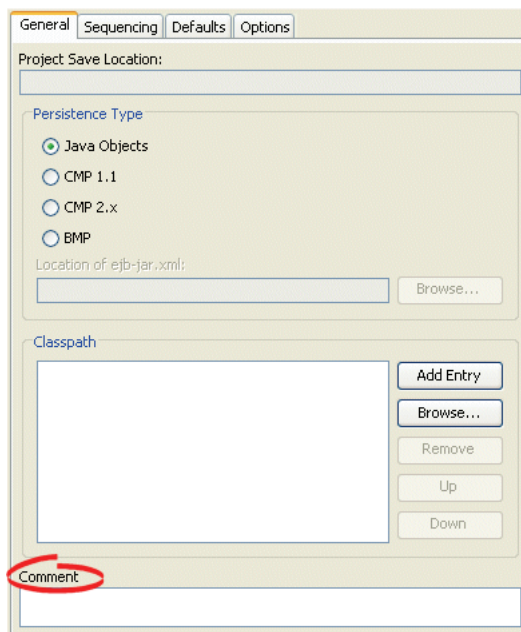
Project Type	How to Use Oracle JDeveloper	How to Configure Project Comments Using TopLink Workbench	How to Use Java
Relational Projects		✓	
EIS Projects		✓	
XML Projects		✓	

### 117.14.1 How to Configure Project Comments Using TopLink Workbench

To specify a comment for the project, use this procedure:

1. Select the project object in the **Navigator**.
2. Select the **General** tab in the **Editor**. The General tab appears.

**Figure 117–13 General Tab, Comments Options**



3. Enter descriptive text information in the **Comment** field.

# Part XXVII

---

## Creation and Configuration of Descriptors

This part describes the TopLink artifact used to describe persistent objects. It contains the following chapters.

- [Chapter 118, "Creating a Descriptor"](#)

This chapter contains procedures for creating TopLink descriptors.

- [Chapter 119, "Configuring a Descriptor"](#)

This chapter explains how to configure TopLink descriptor options common to two or more descriptor types.



---

---

## Creating a Descriptor

This chapter describes how to create TopLink descriptors.

This chapter includes the following sections:

- [Introduction to Descriptor Creation](#)
- [Validating Descriptors](#)
- [Generating Java Code for Descriptors](#)

### 118.1 Introduction to Descriptor Creation

For information on creating different types of descriptors, see the following:

- [Chapter 22, "Creating a Relational Descriptor"](#)
- [Chapter 75, "Creating an EIS Descriptor"](#)
- [Chapter 51, "Creating an XML Descriptor"](#)

After you create a descriptor, you must configure its various options (see [Chapter 119, "Configuring a Descriptor"](#)) and use it to define mappings.

For complete information on the various types of mapping that TopLink supports, see [Chapter 17, "Introduction to Mappings"](#) and [Chapter 120, "Creating a Mapping"](#).

For complete information on the various types of descriptor that TopLink supports, see [Section 16.1, "Descriptor Types"](#).

### 118.2 Validating Descriptors

You can validate descriptors in the following ways:

- Run the project in a test environment and watch for and interpret any exceptions that occur.
- Run the TopLink integrity checker. For more information, see [Section 87.2.8, "Integrity Checker"](#).
- Review the project status report. For more information, see [Section 116.2.3, "How to Generate the Project Status Report"](#).

### 118.3 Generating Java Code for Descriptors

Typically, you capture descriptor configuration in the `project.xml` file and the TopLink runtime reads this information, and then creates and configures all necessary descriptor objects.

Alternatively, for relational projects only, you can export a TopLink project as a Java class (`oracle.toplink.sessions.Project`) that contains all descriptor configuration in Java. This lets you use TopLink Workbench to quickly create and configure descriptors, and then, manually code features that TopLink Workbench does not support. This gives you the best of both TopLink Workbench and Java access to your descriptors. After configuring your Java project class, compile it and include it in your application's JAR file.

For more information, see [Section 19.6.1, "How to Export Project Java Source Using TopLink Workbench"](#).

---

---

## Configuring a Descriptor

This chapter describes how to configure TopLink project options common to two or more project types.

This chapter includes the following sections:

- [Configuring Common Descriptor Options](#)
- [Configuring Primary Keys](#)
- [Configuring Read-Only Descriptors](#)
- [Configuring Unit of Work Conforming at the Descriptor Level](#)
- [Configuring Descriptor Alias](#)
- [Configuring Descriptor Comments](#)
- [Configuring Named Queries at the Descriptor Level](#)
- [Configuring Query Timeout at the Descriptor Level](#)
- [Configuring Cache Refreshing](#)
- [Configuring Query Keys](#)
- [Configuring Interface Query Keys](#)
- [Configuring Cache Type and Size at the Descriptor Level](#)
- [Configuring Cache Isolation at the Descriptor Level](#)
- [Configuring Unit of Work Cache Isolation at the Descriptor Level](#)
- [Configuring Cache Coordination Change Propagation at the Descriptor Level](#)
- [Configuring Cache Expiration at the Descriptor Level](#)
- [Configuring Cache Existence Checking at the Descriptor Level](#)
- [Configuring a Descriptor with EJB CMP and BMP Information](#)
- [Configuring Reading Subclasses on Queries](#)
- [Configuring Inheritance for a Child \(Branch or Leaf\) Class Descriptor](#)
- [Configuring Inheritance for a Parent \(Root\) Descriptor](#)
- [Configuring Inheritance Expressions for a Parent \(Root\) Class Descriptor](#)
- [Configuring Inherited Attribute Mapping in a Subclass](#)
- [Configuring a Domain Object Method as an Event Handler](#)
- [Configuring a Descriptor Event Listener as an Event Handler](#)

- [Configuring Locking Policy](#)
- [Configuring Returning Policy](#)
- [Configuring Instantiation Policy](#)
- [Configuring Copy Policy](#)
- [Configuring Change Policy](#)
- [Configuring a History Policy](#)
- [Configuring Wrapper Policy](#)
- [Configuring Fetch Groups](#)
- [Configuring a Descriptor Customizer Class](#)
- [Configuring Amendment Methods](#)

Table 119–1 lists the types of TopLink descriptors that you can configure and provides a cross-reference to the type-specific chapter that lists the configurable options supported by that type.

**Table 119–1 Configuring TopLink Descriptor**

If you are creating...	See...
Relational Descriptors	Chapter 23, "Configuring a Relational Descriptor"
Object-Relational Data Type Descriptors	Chapter 26, "Configuring an Object-Relational Data Type Descriptor"
EIS Descriptor Concepts	Chapter 76, "Configuring an EIS Descriptor"
XML Descriptor Concepts	Chapter 52, "Configuring an XML Descriptor"

Table 119–2 lists the configurable options shared by two or more TopLink descriptor types.

For more information, see the following:

- [Section 118.1, "Introduction to Descriptor Creation"](#)
- [Chapter 16, "Introduction to Descriptors"](#)

## 119.1 Configuring Common Descriptor Options

Table 119–2 lists the configurable options shared by two or more TopLink descriptor types. In addition to the configurable options described here, you must also configure the options described for the specific [Descriptor Types](#), as shown in [Table 119–1](#).

**Table 119–2 Common Descriptor Options**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Primary keys (see <a href="#">Section 119.2, "Configuring Primary Keys"</a> )	✓	✓	✓
Read-only (see <a href="#">Section 119.3, "Configuring Read-Only Descriptors"</a> )	✓	✓	✓
Unit of work conforming (see <a href="#">Section 119.4, "Configuring Unit of Work Conforming at the Descriptor Level"</a> )	✓	✓	✓
Alias (see <a href="#">Section 119.5, "Configuring Descriptor Alias"</a> )	✓	✓	✓



**Table 119–2 (Cont.) Common Descriptor Options**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Comments (see Section 119.6, "Configuring Descriptor Comments")	✓	✓	
Classes (see Section 5.7.2, "How to Configure Classes")		✓	
Named queries (see Section 119.7, "Configuring Named Queries at the Descriptor Level")	✓	✓	✓
Query timeout (see Section 119.8, "Configuring Query Timeout at the Descriptor Level")	✓	✓	✓
Cache refreshing (see Section 119.9, "Configuring Cache Refreshing")	✓	✓	✓
Query keys (see Section 119.10, "Configuring Query Keys")	✓	✓	✓
Interface query keys (see Section 119.11, "Configuring Interface Query Keys")	✓	✓	✓
Cache type and size (see Section 119.12, "Configuring Cache Type and Size at the Descriptor Level")	✓	✓	✓
Cache isolation (see Section 119.13, "Configuring Cache Isolation at the Descriptor Level")	✓	✓	✓
Unit of work cache isolation (see Section 119.14, "Configuring Unit of Work Cache Isolation at the Descriptor Level")			✓
Cache coordination change propagation (see Section 119.15, "Configuring Cache Coordination Change Propagation at the Descriptor Level")	✓	✓	✓
Cache expiration (see Section 119.16, "Configuring Cache Expiration at the Descriptor Level")	✓	✓	✓
Cache existence checking (see Section 119.17, "Configuring Cache Existence Checking at the Descriptor Level")	✓	✓	✓
EJB information (see Section 119.18, "Configuring a Descriptor with EJB CMP and BMP Information")	✓	✓	✓
Reading subclasses on queries (see Section 119.19, "Configuring Reading Subclasses on Queries")	✓	✓	✓
Inheritance for a child class descriptor (see Section 119.20, "Configuring Inheritance for a Child (Branch or Leaf) Class Descriptor")	✓	✓	✓
Inheritance for a parent class descriptor (see Section 119.21, "Configuring Inheritance for a Parent (Root) Descriptor")	✓	✓	✓
Inheritance expressions for a parent class descriptor (see Section 119.22, "Configuring Inheritance Expressions for a Parent (Root) Class Descriptor")	✓	✓	✓
Inherited attribute mapping in a subclass (see Section 119.23, "Configuring Inherited Attribute Mapping in a Subclass")	✓	✓	✓
Domain object method as an event handler (see Section 119.24, "Configuring a Domain Object Method as an Event Handler")	✓	✓	✓
Descriptor event listener as an event handler (see Section 119.25, "Configuring a Descriptor Event Listener as an Event Handler")		✓	✓
Locking policy (see Section 119.26, "Configuring Locking Policy")	✓	✓	✓
Returning policy (see Section 119.27, "Configuring Returning Policy")	✓	✓	✓

**Table 119–2 (Cont.) Common Descriptor Options**

Option to Configure	Oracle JDeveloper	TopLink Workbench	Java
Instantiation policy (see Section 119.28, "Configuring Instantiation Policy")	✓	✓	✓
Copy policy (see Section 119.29, "Configuring Copy Policy")	✓	✓	✓
Change policy (see Section 119.30, "Configuring Change Policy")			✓
History policy (see Section 119.31, "Configuring a History Policy")			✓
Wrapper policy (see Section 119.32, "Configuring Wrapper Policy")			✓
Fetch groups (see Section 119.33, "Configuring Fetch Groups")			✓
Amendment methods (see Section 119.35, "Configuring Amendment Methods")		✓	
Mapping (see Section 121, "Configuring a Mapping")	✓	✓	✓

## 119.2 Configuring Primary Keys

A **primary key** is a unique identifier (made up of one or more persistent attributes) that distinguishes one instance of a class from all other instances of the same type. You use primary keys to define relationships and to define queries.

For the descriptors shown in Table 119–3, you must configure a primary key and you must ensure that your class contains one or more persistent fields suitable for this purpose.

Table 119–3 summarizes which descriptors support primary keys.

**Table 119–3 Descriptor Support for Primary Keys**

Descriptor	How to Use Oracle JDeveloper	How to Configure Primary Keys Using TopLink Workbench	How to Configure Primary Keys Using Java
Relational Descriptors <sup>1</sup>	✓	✓	✓
Object-Relational Data Type Descriptors			✓
EIS Descriptors <sup>2</sup>	✓	✓	✓
XML Descriptors			

<sup>1</sup> Relational class descriptors only (see Section 22.2.1.1, "Creating Relational Class Descriptors").

<sup>2</sup> EIS root descriptors only (see Section 75.2.1.1, "EIS Root Descriptors").

For a relational class (non-aggregate) descriptor, choose any unique database field or set of unique database fields from the descriptor's associated table (see Section 23.2, "Configuring Associated Tables").

For an EIS root descriptor (see Section 76.6, "Configuring an EIS Descriptor as a Root or Composite Type"), choose any unique attribute or text node or set of unique attributes or text nodes from the descriptor's schema context (see Section 76.2, "Configuring Schema Context for an EIS Descriptor").

## 119.2.1 How to Configure Primary Keys Using TopLink Workbench

To associate a descriptor with one or more primary keys, use this procedure:

1. Select a descriptor in the **Navigator**. Its properties appear in the Editor.
2. Click the **Descriptor Info** tab. The Descriptor Info tab appears.

**Figure 119–1** Descriptor Info Tab, Primary Key Options

The screenshot shows the 'Descriptor Info' tab with the following details:

- Associated Table: ADDRESS
- Primary Keys** (circled in red):
 

ADDRESS.ADDRESS_ID	Add...	Remove
--------------------	--------	--------
- Use Sequencing: 
  - Name:
  - Table: <none selected>
  - Field: <none selected>
- Read-Only:
- Conform Results in Unit of Work:
- Descriptor Alias: Address
- Comment:

Use this table to enter data in Primary Keys field on the descriptor's **Descriptor Info** tab to specify the primary key(s):

Field	Description
<b>Primary Keys</b>	<p>To specify the primary keys for the table, click <b>Add</b> in order to do the following:</p> <ul style="list-style-type: none"> <li>■ For a relational class descriptor, select a database field from the descriptor's associated table (see <a href="#">Section 23.2, "Configuring Associated Tables"</a>).</li> <li>■ For an EIS root descriptor, select an attribute or text node from the descriptor's schema context (see <a href="#">Section 76.2, "Configuring Schema Context for an EIS Descriptor"</a>). For more information on choosing an element or attribute, see <a href="#">Section 76.2.1.1, "Choosing a Schema Context"</a>.</li> </ul> <p>To remove a primary key, select the key and click <b>Remove</b>.</p>

## 119.2.2 How to Configure Primary Keys Using Java

You can use Java to configure primary keys for the following:

- [Relational Projects](#)
- [EIS Projects](#)

### 119.2.2.1 Relational Projects

Use `ClassDescriptor` method `addPrimaryKeyFieldName` to specify the primary key field of the descriptor's table. This should be called for each field that makes up the primary key of the table.

If the descriptor has more than one table, and all other tables have the same primary key, use the `ClassDescriptor` method `addPrimaryKeyFieldName` to specify the the primary key in the first table.

If the descriptor has more than one table, and each table has a different primary key, use `ClassDescriptor` method `addForeignKeyFieldNameForMultipleTable` to map the source foreign key field name to target primary key field name.

### 119.2.2.2 EIS Projects

Use `EISDescriptor` method `addPrimaryKeyFieldName` to specify the primary key field of the descriptor's class. Call this method for each field that makes up the primary key.

## 119.3 Configuring Read-Only Descriptors

You can configure a relational class or EIS root descriptor as read-only. This indicates that instances of the reference class will never be modified.

Read-only descriptors are usually used within a unit of work as a performance gain, because there is no need to register, clone, and merge the read-only classes. For more information, see [Chapter 113, "Introduction to TopLink Transactions"](#).

In a CMP project, you can declare an entity bean as read-only within the TopLink deployment XML file. For more information, see [Section 119.3.1, "How to Use Read-Only EJB CMP Entity Beans"](#).

[Table 119–4](#) summarizes which descriptors support read-only configuration.

**Table 119–4** Descriptor Support for Read Only

Descriptor	How to Use Oracle JDeveloper	How to Configure Read-Only Descriptors Using TopLink Workbench	How to Configure Read-Only Descriptors Using Java
Relational Descriptors <sup>1</sup>	✓	✓	✓
Object-Relational Data Type Descriptors			✓
EIS Descriptor <sup>2</sup>	✓	✓	✓
XML Descriptors			

<sup>1</sup> Relational class descriptors only (see [Section 22.2.1.1, "Creating Relational Class Descriptors"](#)).

<sup>2</sup> EIS root descriptors only (see [Section 75.2.1.1, "EIS Root Descriptors"](#))

---

**Note:** Relational aggregate and EIS composite descriptors get their read-only setting from their owner.

---

### 119.3.1 How to Use Read-Only EJB CMP Entity Beans

TopLink can declare an entity bean with container-managed persistence as read-only. This ensures that the entity bean cannot be modified and allows TopLink to optimize unit of work performance.

If an attempt is made to modify a read-only entity bean (create, update, or remove), TopLink immediately throws a `javax.ejb.EJBException`: TopLink does not wait until the transaction commits.

If an attempt is made to change a CMR field on a read-only entity bean, TopLink throws a `javax.ejb.EJBException`.

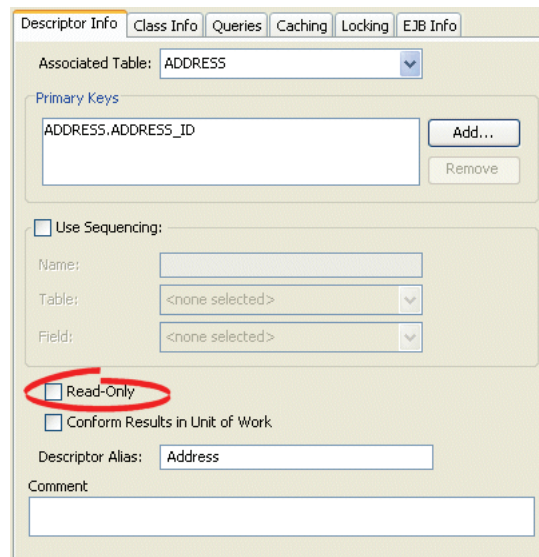
When TopLink is configured as the OC4J persistence manager, the TopLink read-only bean configuration replaces the OC4J `READ-ONLY CMP` concurrency mode.

### 119.3.2 How to Configure Read-Only Descriptors Using TopLink Workbench

To configure a descriptor as read-only use this procedure:

1. Select a descriptor in the **Navigator**. Its properties appear in the Editor.
2. Click the **Descriptor Info** tab. The Descriptor Info tab appears.

**Figure 119–2 Descriptor Info Tab, Read Only Option**



Specify whether this descriptor is read-only or not.

### 119.3.3 How to Configure Read-Only Descriptors Using Java

Use `ClassDescriptor` method `setReadOnly`.

## 119.4 Configuring Unit of Work Conforming at the Descriptor Level

Conforming is a query feature that lets you include new, changed, or deleted objects in queries within a unit of work prior to committing the transaction. This feature enables you to query against your relative logical or transaction view of a data source.

[Table 119–5](#) summarizes which descriptors support descriptor level unit of work conforming.

**Table 119–5 Descriptor Support for Unit of Work Conforming**

<b>Descriptor</b>	<b>How to Use Oracle JDeveloper</b>	<b>How to Configure Unit of Work Conforming at the Descriptor Level Using TopLink Workbench</b>	<b>How to Configure Unit of Work Conforming at the Descriptor Level Using Java</b>
Relational Descriptors <sup>1</sup>	✓	✓	✓
Object-Relational Data Type Descriptors			✓
EIS Descriptors <sup>2</sup>	✓	✓	✓
XML Descriptors			

<sup>1</sup> Relational class descriptors only (see [Section 22.2.1.1, "Creating Relational Class Descriptors"](#)).

<sup>2</sup> EIS root descriptors only (see [Section 75.2.1.1, "EIS Root Descriptors"](#))

When you configure a descriptor to conform results in a unit of work, when you execute a query in the unit of work, TopLink filters the data source result set to the changes currently made in the unit of work. TopLink adds new or changed objects that correspond to the query's selection criteria and removes changed objects that no longer correspond to the query's selection criteria.

---



---

**Note:** For EIS root descriptors, only deleted objects would be filtered, not new or changed objects.

---



---

Conforming can reduce performance. Before you enable a descriptor for conforming, be aware of its limitations (see [Section 115.4.1, "How to Use Conforming"](#)) and make sure that conforming is actually necessary.

For examples, see the following:

- [Section 115.4, "Using Conforming Queries and Descriptors"](#)
- [Section 115.4.4, "What You May Need to Know About Conforming Query Alternatives"](#)

### 119.4.1 How to Configure Unit of Work Conforming at the Descriptor Level Using TopLink Workbench

To conform a descriptor's results in a unit of work, use this procedure:

1. Select a descriptor in the **Navigator**. Its properties appear in the Editor.
2. Click the **Descriptor Info** tab. The Descriptor Info tab appears.

**Figure 119–3 Descriptor Info Tab, Conform Results in Unit of Work Option**

The screenshot shows the 'Descriptor Info' tab in Oracle JDeveloper. The 'Associated Table' is set to 'ADDRESS'. Under 'Primary Keys', 'ADDRESS.ADDRESS\_ID' is listed. There are 'Add...' and 'Remove' buttons. Below this, there are sections for 'Use Sequencing' (with Name, Table, and Field dropdowns) and 'Read-Only'. The 'Conform Results in Unit of Work' checkbox is checked and highlighted with a red circle. The 'Descriptor Alias' is 'Address' and there is a 'Comment' field at the bottom.

Enable or disable conforming: when enabled, this feature ensures that any queries for this descriptor will conform the data source result with the current changes in the unit of work. For more information, see [Section 115.4.1, "How to Use Conforming"](#).

## 119.4.2 How to Configure Unit of Work Conforming at the Descriptor Level Using Java

Use `ClassDescriptor` method  
`setShouldAlwaysConformResultsInUnitOfWork(true)`.

## 119.5 Configuring Descriptor Alias

In EJB CMP, use the descriptor alias to specify the value of the `ejb-jar.xml` attribute `abstract-schema-name`. This is the logical name that is referenced in EJB QL queries. You should configure a descriptor alias for each entity bean with container-managed persistence. The descriptor alias defaults to the class name without a path information.

[Table 119–6](#) summarizes which descriptors support descriptor alias configuration.

**Table 119–6 Descriptor Support for Descriptor Alias Configuration**

Descriptor	How to Use Oracle JDeveloper	How to Configure Descriptor Alias Using TopLink Workbench	How to Configure Descriptor Alias Using Java
Relational Descriptors	✓	✓	✓
Object-Relational Data Type Descriptors			
EIS Descriptors <sup>1</sup>	✓	✓	✓
XML Descriptors			

<sup>1</sup> EIS root descriptors only (see [Section 75.2.1.1, "EIS Root Descriptors"](#)).

---

**Note:** The alias is also used in JPA—it is the entity name. This is the logical name referenced in JP QL queries. It defaults to the class name without a path information.

For more information, see "Introduction to EclipseLink JPA" chapter of *EclipseLink Developer's Guide* at

[http://wiki.eclipse.org/Introduction\\_to\\_EclipseLink\\_JPA\\_%28ELUG%29](http://wiki.eclipse.org/Introduction_to_EclipseLink_JPA_%28ELUG%29).

---

For more information, see the following:

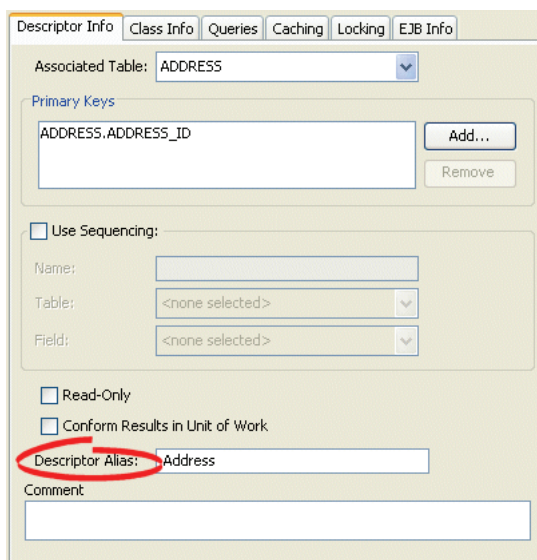
- [Section 9.1.3, "ejb-jar.xml File"](#)
- [Section 108.15, "EJB 2.n CMP Finders"](#)

## 119.5.1 How to Configure Descriptor Alias Using TopLink Workbench

To specify a descriptor alias, use this procedure:

1. In the **Navigator**, select a descriptor.
2. Click the **Descriptor Info** tab in the Property window.

**Figure 119–4** *Descriptor Info Tab, Descriptor Alias Field*



In the **Descriptor Alias** field, enter an alias for this descriptor. For more information, see [Section 119.5, "Configuring Descriptor Alias"](#).

## 119.5.2 How to Configure Descriptor Alias Using Java

Use `ClassDescriptor` method `setAlias` passing in the descriptor alias as a `String`.



## 119.6 Configuring Descriptor Comments

You can define a free-form textual comment for each descriptor. You can use these comments however you wish: for example, to record important project implementation details such as the purpose or importance of a descriptor.

Comments are stored in the TopLink Workbench project, in the TopLink deployment XML file. There is no Java API for this feature.

Table 119–7 summarizes which descriptors support descriptor comment configuration.

**Table 119–7** Descriptor Support for Descriptor Comment Configuration

Descriptor	How to Use Oracle JDeveloper	How to Configure Descriptor Comments Using TopLink Workbench	How to Use Java
Relational Descriptors		✓	
Object-Relational Data Type Descriptors			
EIS Descriptors		✓	
XML Descriptors		✓	

### 119.6.1 How to Configure Descriptor Comments Using TopLink Workbench

To create a comment for a descriptor, use this procedure:

1. In the **Navigator**, select a descriptor.
2. Click the **Descriptor Info** tab in the Property window.

**Figure 119–5** Descriptor Info Tab, Comment Field

The screenshot shows the 'Descriptor Info' tab in the TopLink Workbench property window. The 'Associated Table' is set to 'ADDRESS'. Under 'Primary Keys', 'ADDRESS.ADDRESS\_ID' is listed. There are checkboxes for 'Use Sequencing', 'Read-Only', and 'Conform Results in Unit of Work'. The 'Descriptor Alias' is 'Address'. The 'Comment' field is a text area at the bottom, which is circled in red in the image.

In the **Comment** field, enter a description of this descriptor.

## 119.7 Configuring Named Queries at the Descriptor Level

A named query is a TopLink query that you create and store, by name, in a descriptor's `DescriptorQueryManager` for later retrieval and execution. Named queries improve application performance because they are prepared once and they (and all their associated supporting objects) can be efficiently reused thereafter making them well suited for frequently executed operations.

If a named query is global to a `Class`, configure it at the descriptor level. Alternatively, you can configure a named query at the session level (see [Section 89.13](#), "Configuring Named Queries at the Session Level").

Use named queries to specify SQL, EJB QL, or TopLink Expression queries to access your data source.

For EJB CMP entity bean descriptors, you must define a named query for every finder defined for the corresponding entity bean.

---



---

**Note:** You can also use named queries in JPA (see "Using EclipseLink JPA Extensions for Stored Procedure Query" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#Using\\_EclipseLink\\_JPA\\_Extensions\\_for\\_Stored\\_Procedure\\_Query](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#Using_EclipseLink_JPA_Extensions_for_Stored_Procedure_Query)). Because the scope of JPA named queries is global to the session, ensure that each named query has a unique name.

---



---

Using Oracle JDeveloper or TopLink Workbench, you can configure named queries for a subset of query types and store them in a descriptor's `DescriptorQueryManager` (see [Section 119.7.1](#), "How to Configure Named Queries at the Descriptor Level Using TopLink Workbench").

Using Java, you can create named queries for all query types and store them in a descriptor's `DescriptorQueryManager` (see [Section 119.7.2](#), "How to Configure Named Queries at the Descriptor Level Using Java").

[Table 119-4](#) summarizes which descriptors support named query configuration.

**Table 119-8 Descriptor Support for Named Queries**

Descriptor	How to Use Oracle JDeveloper	How to Configure Named Queries at the Descriptor Level Using TopLink Workbench	How to Configure Named Queries at the Descriptor Level Using Java
Relational Descriptors <sup>1</sup>	✓	✓	✓
Object-Relational Data Type Descriptors			✓
EIS Descriptor <sup>2</sup>	✓	✓	✓
XML Descriptors			

<sup>1</sup> Relational class descriptors only (see [Section 22.2.1.1](#), "Creating Relational Class Descriptors").

<sup>2</sup> EIS root descriptors only (see [Section 75.2.1.1](#), "EIS Root Descriptors").

For CMP projects, the `ejb-jar.xml` file stores query lists. You can define the queries in the file, and then read them into Oracle JDeveloper or TopLink Workbench, or

define them on the **Queries** tab and write them to the file. See [Section 19.7, "Working with the ejb-xml.File"](#) for more information.

After you create a named query, you can execute it by name and class on the TopLink session (see [Section 109.3, "Using Named Queries"](#)).

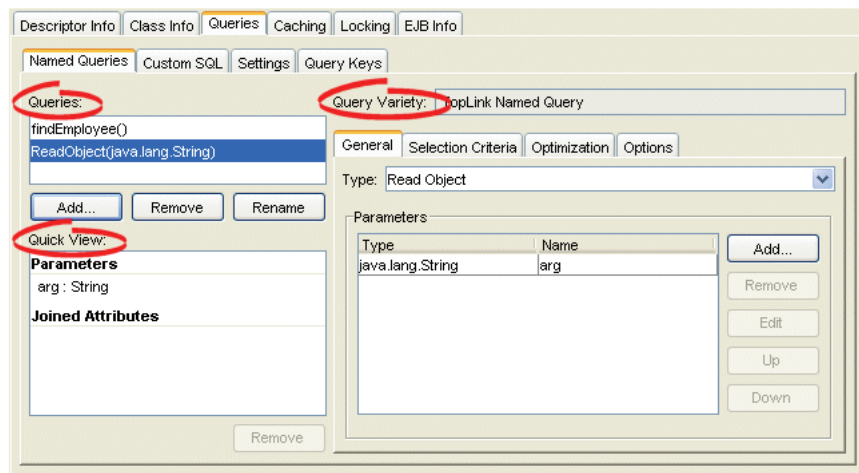
For more information about named queries, see [Section 108.8, "Named Queries"](#).

## 119.7.1 How to Configure Named Queries at the Descriptor Level Using TopLink Workbench

To create a named query, use this procedure

1. In the **Navigator**, select a descriptor. Its properties appear in the Editor.
2. Click the **Queries** tab in the Editor. The Queries tab appear with three additional tabs.
3. Click the **Named Queries** tab in the Queries tab. The Named Queries tab appears.

**Figure 119–6 Queries Tab–Named Queries Subtab**



Use the following information to complete each field on this tab:

Field	Description
<b>Queries</b>	Lists the existing queries for this descriptor. <ul style="list-style-type: none"> <li>■ To create a new query, click <b>Add</b> (see <a href="#">Section 119.7.1.1, "Adding Named Queries"</a>).</li> <li>■ To delete an existing query, select the query and click <b>Delete</b>. TopLink Workbench prompts for confirmation.</li> <li>■ To rename an existing query, select the query and click <b>Rename</b>. The Rename dialog box appears. Type a new name for the query and click <b>OK</b>.</li> </ul>
<b>Query Variety</b>	Displays the variety of the currently selected query (see <a href="#">Section 119.7.1.1, "Adding Named Queries"</a> ).
<b>Quick View</b>	Lists the parameters and joined attributes defined for the query. Clicking on a heading in the <b>Quick View</b> area selects the corresponding subtab. You can also remove parameters or attributes from the Quick View area by selecting the item and clicking <b>Remove</b> .

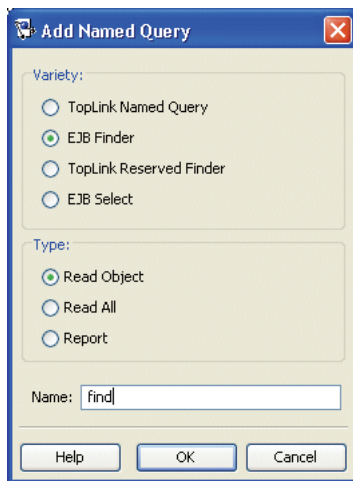
The Named Queries tab includes the following subtabs:

- **General**—See [Section 119.7.1.2, "Configuring Named Query Type and Parameters"](#).
- **Selection Criteria**—See [Section 119.7.1.3, "Configuring Named Query Selection Criteria"](#).
- **Order**—This tab appears for `ReadAllQuery` queries only. See [Section 119.7.1.4, "Configuring Read All Query Order"](#).
- **Optimization**—See [Section 119.7.1.5, "Configuring Named Query Optimization"](#).
- **Attributes**—This tab appears for `ReportQuery` queries only. See [Section 119.7.1.6, "Configuring Named Query Attributes"](#).
- **Group/Order**—This tab appears for `ReportQuery` queries only. [Section 119.7.1.7, "Configuring Named Query Group/Order Options"](#).
- **Calls**—This tab appears for EIS root descriptors only (for `ReadAllQuery` and `ReadObjectQuery` queries). See [Section 119.7.1.8, "Creating an EIS Interaction for a Named Query"](#).
- **Options**—See [Section 119.7.1.9, "Configuring Named Query Options"](#).

### 119.7.1.1 Adding Named Queries

Use this dialog box to create a new named query.

**Figure 119–7 Add Named Query Dialog Box**



Use the following information to complete the dialog box and create the named query:

Field	Description
<b>Variety</b>	For EJB CMP descriptors only, select the variety of query:
<b>TopLink Named Query</b>	Select to create a general purpose TopLink query of the type given by the <b>Type</b> area. You can execute this query by name on the TopLink session passing in the class and arguments (see <a href="#">Section 109.3, "Using Named Queries"</a> ).

Field	Description
<b>EJB Finder</b>	Select to create a TopLink query of the type given by the <b>Type</b> area for use as the implementation of the EJB CMP finder method of the name you enter. The TopLink runtime executes this query when you call the EJB CMP finder method of the given name.
<b>TopLink Reserved Finder</b>	Select to create a TopLink query of the type given by the <b>Type</b> area for use as the implementation of the TopLink reserved finder method name you select. The TopLink runtime executes this query when you call the EJB CMP finder method of the given name.  For more information, see <a href="#">Section 108.15.1, "Predefined Finders"</a> .
<b>EJB Select</b>	Select to create a TopLink query of the type given by the <b>Type</b> area for use as the implementation of the EJB CMP life cycle method <code>ejbSelect</code> . The TopLink runtime executes this query whenever the <code>ejbSelect</code> method is called.
<b>Type</b>	Select the type of query: <ul style="list-style-type: none"> <li>■ <b>ReadObject</b> (<code>ReadObjectQuery</code>)</li> <li>■ <b>ReadAll</b> (<code>ReadAllQuery</code>)</li> <li>■ <b>Report</b><sup>1</sup> (<code>ReportQuery</code>)</li> </ul> <p><b>Note:</b> If <b>Variety</b> is set to <b>TopLink Reserved Finder</b>, you cannot select a query <b>Type</b>.</p>
<b>Name</b>	The name of this query. <ul style="list-style-type: none"> <li>■ If <b>Variety</b> is set to <b>TopLink Named Query</b>, you can specify any name.</li> <li>■ If <b>Variety</b> is set to <b>EJB Finder</b>, the name must be prefixed by <b>find</b>.</li> <li>■ If <b>Variety</b> is set to <b>TopLink Reserved Finder</b>, select from the list of available names that TopLink reserves. For more information, see <a href="#">Section 108.15.1, "Predefined Finders"</a>.</li> <li>■ If <b>Variety</b> is set to <b>EJB Select</b>, the name must be <b>ejbSelect</b>.</li> </ul>

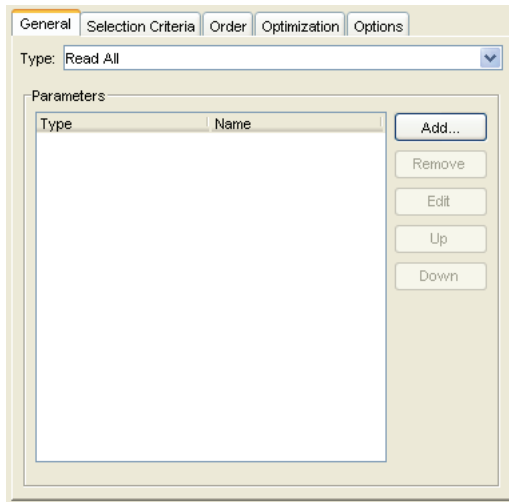
<sup>1</sup> Relational descriptors only.

Enter the necessary information and click **OK**. TopLink Workbench adds the query to the list of queries in the Named Query tab.

### 119.7.1.2 Configuring Named Query Type and Parameters

Use this tab to select the query type and add or remove parameters.

**Figure 119–8 Named Queries, General Tab**



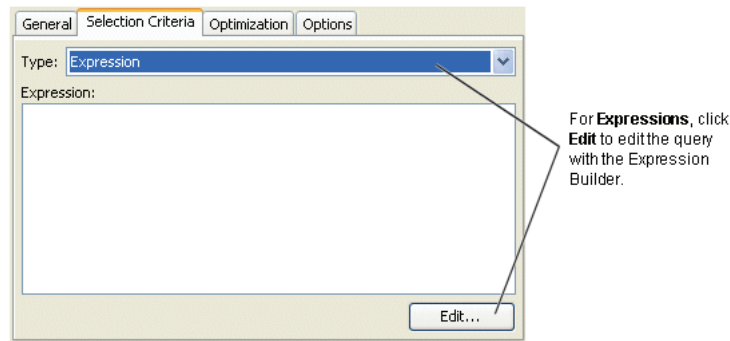
Use the following information to complete each field on this tab:

Field	Description
Type	<p>Select the type of query from the list. You can create any of the following query types:</p> <ul style="list-style-type: none"> <li>■ <b>ReadAllQuery</b></li> <li>■ <b>ReadObjectQuery</b></li> <li>■ <b>ReportQuery</b><sup>1</sup></li> </ul> <p>To create other types of query, you must use Java (see <a href="#">Section 119.7.2, "How to Configure Named Queries at the Descriptor Level Using Java"</a>).</p> <p>When you change the type of an existing query, TopLink Workbench preserves any configuration that is common between the old and new type and warns you if changing the type will result in the loss of configuration that is not shared by the new type.</p>
Parameters	<p>For queries that take parameters, specify the parameters:</p> <ul style="list-style-type: none"> <li>■ To add a new parameter, click <b>Add</b>. The Add Query Parameter dialog box appears. Click <b>Browse</b> to select the type, specify a name, and click <b>OK</b>.</li> <li>■ To delete an existing parameter, select the parameter and click <b>Remove</b>. TopLink Workbench prompts for confirmation.</li> <li>■ To modify an existing parameter, select the parameter and click <b>Edit</b>. The Edit Query Parameter dialog box appears. Modify the name and type of the parameter and click <b>OK</b>.</li> <li>■ To change the order of the parameters, select an existing parameter and click <b>Up</b> or <b>Down</b>.</li> </ul>
Type	Select the class of the parameter's type.
Name	Enter the name of the parameter.

<sup>1</sup> Relational descriptors only.

### 119.7.1.3 Configuring Named Query Selection Criteria

Use this tab to specify the format of the named query and enter the query string.

**Figure 119–9 Named Queries, Selection Criteria Tab**

Use the following information to complete each field on this tab:

Field	Description
Type	Specify if query uses a TopLink <b>Expression</b> , <b>SQL</b> , or <b>EJB QL</b> .
Expression or Query String	<p>If the <b>Type</b> is <b>SQL</b> or <b>EJB QL</b>, enter the query string (either <b>SQL</b> or <b>EJB QL</b>).</p> <p>TopLink Workbench does not validate the query string.</p> <p>See a note that follows this table for information on query syntax.</p>

---

**Note:** Use a combination of an escape character and a double-quote ( \ " ) instead of just a double-quote ( " ) when defining your query using **SQL** or **EJB QL**. For example:

```
SELECT OBJECT(employee) FROM Employee employee WHERE
employee.name = \"Bob\"
```

If you fail to do so, the generated Java code would look as follows:

```
query.setEJBQLString("SELECT OBJECT(employee) FROM Employee
employee WHERE employee.name = \"Bob\"");
```

The preceding code produces an error at compile time.

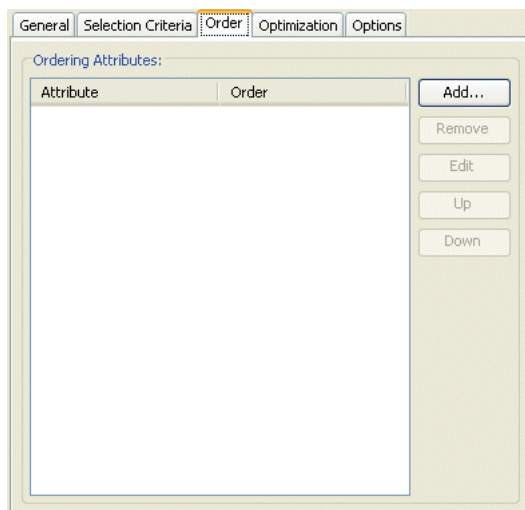
If you define your query using the correct syntax, the generated Java code will contain no errors and be similar to the following:

```
query.setEJBQLString("SELECT OBJECT(employee) FROM Employee
employee WHERE employee.name = \"Bob\"");
```

---

#### 119.7.1.4 Configuring Read All Query Order

Use this tab to specify how the results of a read all query should be ordered by attribute name.

**Figure 119–10 Named Queries, Order Tab**

Select one of the following actions:

- To add a new attribute by which to order query results, click **Add**. The Add Ordering Attribute dialog box appears. Select the mapped attribute to order by, specify ascending or descending order, and then click **OK**.
- To change the order of the order attributes, select an existing attribute and click **Up** or **Down**.
- To modify an existing order attribute's ordering options, select an existing attribute and click **Edit**.
- To remove an order attribute, select an existing attribute and click **Remove**.

#### 119.7.1.5 Configuring Named Query Optimization

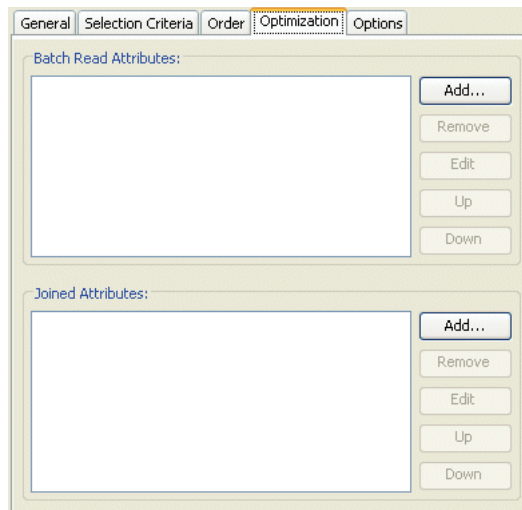
You can optimize a named query by configuring batch (`ReadAllQuery` only) or joining (`ReadAllQuery` and `ReadObjectQuery`) attributes.

For more information on using batch reading, see [Optimizing Queries, Section 12.12.9.2, "Reading Case 2: Batch Reading Objects"](#), and [Section 109.2.1.9, "Using Batch Reading"](#).

For more information on joining, see [Section 108.7.1.5, "Join Reading and Object-Level Read Queries"](#) and [Section 109.2.1.10, "Using Join Reading with ObjectLevelReadQuery"](#).

Use this tab to specify batch reading and joining attributes.



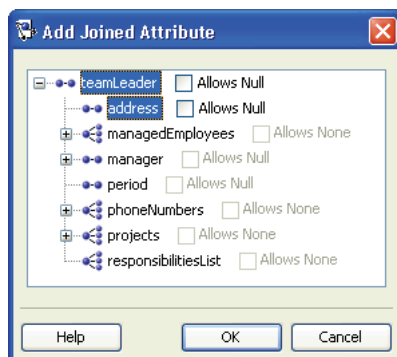
**Figure 119–11** *Named Queries, Optimization Tab*

Select one of the following actions for **Batch Read Attributes** (ReadAllQuery only):

- To add a new batch read attribute, click **Add**. The Add Batch Read Attribute dialog box appears. Select the mapped attribute and click **OK**.
- To change the order of the batch read attributes, select an existing attribute and click **Up** or **Down**.
- To modify an existing batch read attribute's options, select an existing attribute and click **Edit**.
- To remove a batch read attribute, select an existing attribute and click **Remove**.

Select one of the following actions for **Joined Attributes** (ReadAllQuery and ReadObjectQuery):

- To add a new joined attribute, click **Add**. The Add Joined Attribute dialog box appears.

**Figure 119–12** *Add Joined Attribute Dialog Box*

Select the mapped attribute. Optionally, enable or disable **Allows Null** or, for a Collection attribute, **Allows None**. Click **OK**.

- To change the order of the joined attributes, select an existing attribute and click **Up** or **Down**.
- To modify an existing joined attribute's options, select an existing attribute and click **Edit**.

- To remove a joined attribute, select an existing attribute and click **Remove**.

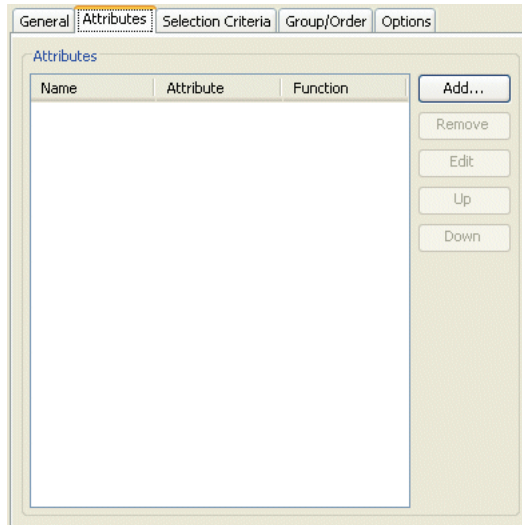
### 119.7.1.6 Configuring Named Query Attributes

For ReportQuery queries only, you can configure report query functions to apply to one or more attributes.

For more information about report queries, see [Section 108.7.5, "Report Query"](#).

Use this tab to configure report query attributes.

**Figure 119–13** *Named Queries, Attributes Tab*



Select one of the following actions for **Attributes** (ReportQuery only):

- To add a new report query attribute, click **Add**. The Add Joined Attribute dialog box appears. Continue with [Section 119.7.1.6.1, "Adding Report Query Attributes"](#).
- To change the order of the report query attribute attributes, select an existing attribute and click **Up** or **Down**.
- To modify an existing report query attribute's options, select an existing attribute and click **Edit**.
- To remove a report query attribute, select an existing attribute and click **Remove**.

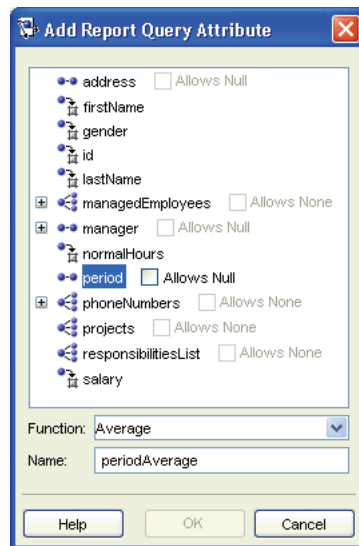
---

**Note:** You can only choose attributes that are configured with a direct mapping (converters included) or a user-defined query key.

---

#### 119.7.1.6.1 Adding Report Query Attributes

Use this dialog box to add a report query attribute.

**Figure 119–14 Add Report Query Attribute Dialog Box**

Select the attribute you want in this report query and use the following table to complete the dialog box and add the report query attribute:

Option	Description
<b>Allows None or Allows Null</b>	<p>Use the <b>Allows Null</b> and <b>Allows None</b> options to define an expression with an outer join.</p> <p>Check the <b>Allows Null</b> option to use the <code>ExpressionBuilder</code> method <code>getAllowingNull</code>.</p> <p>Check the <b>Allows None</b> option for <code>Collection</code> attributes to use the <code>ExpressionBuilder</code> method <code>anyOfAllowingNone</code>.</p> <p>For more information, see <a href="#">Section 110.2.7.2, "Using TopLink Expression API for Joins"</a>.</p>
<b>Function</b>	<p>Select from the list of report query functions that TopLink provides. This function will be applied to the specified attribute. You must select an attribute for all functions, except <b>Count</b>.</p> <p>Alternatively, you can enter the name of a custom function that you implement in your database. For more information, see <code>Expression</code> method <code>getFunction</code> in the <i>Oracle TopLink API Reference</i>.</p>
<b>Name</b>	<p>The name associated with the calculated value. By default, the name is <code>&lt;AttributeName&gt;&lt;FunctionName&gt;</code>.</p>

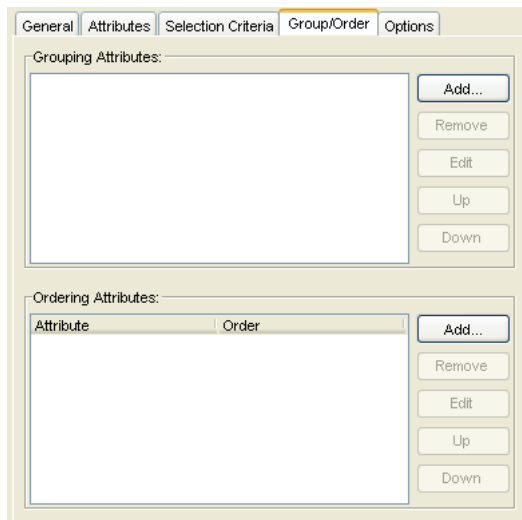
Enter the necessary information and click **OK**. TopLink Workbench adds the report query attribute to the list of attributes in the Attribute tab.

### 119.7.1.7 Configuring Named Query Group/Order Options

For `ReportQuery` queries only, you can configure grouping and ordering attributes.

For more information about report queries, see [Section 108.7.5, "Report Query"](#).

Use this tab to specify grouping and ordering attributes.

**Figure 119–15 Named Queries, Group/Order Tab**

Select one of the following actions for **Grouping Attributes** (ReportQuery only):

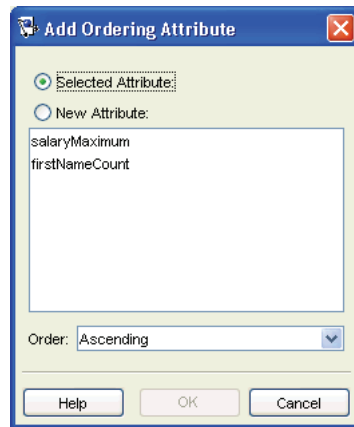
- To add a new grouping attribute, click **Add**. The Add Grouping Attribute dialog appears. Select the desired mapped attribute and click **OK**.
- To change the order of the grouping attributes, select an existing attribute and click **Up** or **Down**.
- To modify an existing grouping attribute's options, select an existing attribute and click **Edit**.
- To remove a grouping attribute, select an existing attribute and click **Remove**.

Select one of the four following actions for **Ordering Attributes** (ReportQuery only):

- To add a new ordering attribute, click **Add**. The Add Ordering Attribute dialog box appears. Continue with [Section 119.7.1.7.1, "Adding Ordering Attributes"](#).
- To change the order of the ordering attributes, select an existing attribute and click **Up** or **Down**.
- To modify an existing ordering attribute's options, select an existing attribute and click **Edit**.
- To remove an ordering attribute, select an existing attribute and click **Remove**.

#### 119.7.1.7.1 Adding Ordering Attributes

Use this dialog box to add a report query ordering attribute.

**Figure 119–16 Add Ordering Attribute Dialog Box**

Use the following information to complete the fields on the dialog box and add an ordering attribute:

Option	Description
<b>Selected Attribute</b>	Select this option to view a list of the report query attributes you added (see <a href="#">Section 119.7.1.6, "Configuring Named Query Attributes"</a> ). Select an attribute and choose its ordering option in the <b>Order</b> field.
<b>New Attribute</b>	Select this option to view a list of all class attributes. Select an attribute and choose its ordering option in the <b>Order</b> field.
<b>Order</b>	Select ascending or descending.

Enter the necessary information and click **OK**. TopLink Workbench adds the report query attribute to the list of attributes in the Attribute tab.

### 119.7.1.8 Creating an EIS Interaction for a Named Query

For an EIS root descriptor, you can define EIS interactions to invoke methods on an EIS.

You can use TopLink to define an interaction as a named query for read object and read all object queries, as described here. These queries are not called for basic persistence operations ([Section 76.5, "Configuring Custom EIS Interactions for Basic Persistence Operations"](#)); you can call these additional queries by name in your application for special purposes.

Use this tab to define an interaction as a named query for read object and read all object queries.

**Figure 119–17 Call Tab**

Use the following information to complete each field on the tab:

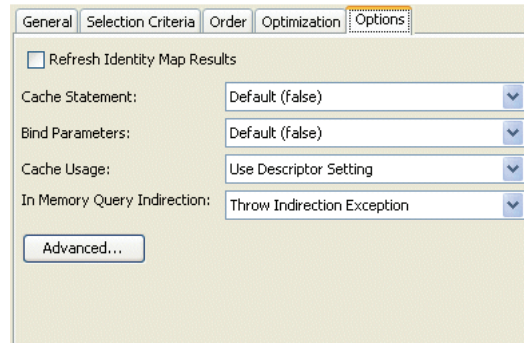
Field	Description
<b>Interaction Type</b>	Using TopLink Workbench, you can only use XML Interactions. You cannot change this field.
<b>Function Name</b>	Specify the name of the EIS function that this call type (Read Object or Read All) invokes on the EIS.
<b>Input Record Name</b>	Specify the name passed to the JCA adapter when creating the input record.
<b>Input Root Element</b>	Specify the root element name to use for the input DOM.
<b>Input Arguments</b>	Specify the query argument name to map to the interaction field or XPath nodes in the argument record.  For example, if you are using XML records, use this option to map input argument name to the XPath <code>name/first-name</code> .
<b>Output Arguments</b>	Specify the result record field or XPath nodes to map the correct nodes in the record used by the descriptor's mappings.  For example, if you are using XML records, use this option to map the output <code>fname</code> to <code>name/first-name</code> .  Output arguments are not required if the interaction returns an XML result that matches the descriptor's mappings.
<b>Input Result Path</b>	Use this option if the EIS interaction expects the interaction arguments to be nested in the XML record.  For example, specify <code>arguments</code> , if the arguments were to be nested under the root element <code>exec-find-order</code> , then under an <code>arguments</code> element.
<b>Output Result Path</b>	Use this option if the EIS interaction result record contains the XML data that maps to the objects in a nested structure.  For example, specify <code>order</code> , if the results were return under a root element <code>results</code> , then under an <code>order</code> element.

Field	Description
Properties	Specify any properties required by your EIS platform. For example, property name <code>operation</code> (from <code>AQPlatform.QUEUE_OPERATION</code> ) and property value <code>enqueue</code> (from <code>AQPlatform.ENQUEUE</code> ).

### 119.7.1.9 Configuring Named Query Options

Use this tab to configure additional options for the query.

**Figure 119–18** *Named Queries, Options Tab*



Use the following information to complete each field on the tab:

Field	Description
<b>Refresh Identity Map Results</b> <sup>2</sup>	Refreshes the attributes of the object(s) resulting from the query. If cascading is used, the private parts of the objects will also be refreshed.
<b>Cache Statement</b> <sup>1</sup>	Caches the prepared statements. This requires full parameter binding as well (see <b>Bind Parameters</b> ).
<b>Bind Parameters</b> <sup>1</sup>	By default, TopLink binds all of the query's parameters. Deselect this option to disable binding.
<b>Cache Usage</b> <sup>2</sup>	<p>Selects how TopLink should use the session cache when a query is executed:</p> <ul style="list-style-type: none"> <li>■ Use descriptor settings</li> <li>■ Do not check cache</li> <li>■ Check cache by exact primary key</li> <li>■ Check cache by primary key</li> <li>■ Check cache then database</li> <li>■ Check cache only</li> <li>■ Conform results in unit of work</li> </ul> <p>For more information, see the following:</p> <ul style="list-style-type: none"> <li>■ <a href="#">Section 108.16.2.1, "Configuring Cache Usage for In-Memory Queries"</a>.</li> <li>■ <a href="#">Section 119.4, "Configuring Unit of Work Conforming at the Descriptor Level"</a></li> </ul>

Field	Description
<b>In Memory Query Indirection</b> <sup>2</sup>	<p>Selects how TopLink should handle indirection (lazy loading) when an in-memory or conforming query is executed:</p> <ul style="list-style-type: none"> <li>■ Throw indirection exception—if this object uses indirection and indirection has not been triggered, TopLink will throw an exception.</li> <li>■ Trigger indirection—if this object uses indirection and indirection has not been triggered, TopLink will trigger indirection.</li> <li>■ Ignore exception return conformed—returns conforming if an untriggered value holder is encountered. That is, you expect results from the database to conform, and an untriggered value holder is taken to mean that the underlying attribute has not changed.</li> <li>■ Ignore exception return not conformed—returns not conforming if an untriggered value holder is encountered.</li> </ul> <p>For more information, see the following:</p> <ul style="list-style-type: none"> <li>■ <a href="#">Section 108.16.2.3, "Handling Exceptions Resulting from In-Memory Queries"</a>.</li> <li>■ <a href="#">Section 17.2.4, "Indirection (Lazy Loading)"</a>.</li> </ul>
<b>Return Choice</b> <sup>3</sup>	<p>Selects how TopLink should handle ReportQuery results:</p> <ul style="list-style-type: none"> <li>■ Result collection—return ReportQuery results as a Collection of ReportQueryResult objects.</li> <li>■ Single result—return only the first ReportQueryResult object (not wrapped in a Collection or Map). Use this option if you know that the ReportQuery returns only one row.</li> <li>■ Single value—return only a single value. Use this option if you know that the ReportQuery returns only one row that contains only one attribute.</li> <li>■ Single attribute—return only a single Collection of values. If the query returns multiple rows, but each row only has a single attribute, this option will return a Collection of values, instead of a Collection of ReportQueryResults.</li> </ul> <p>For more information, see <a href="#">Section 108.5.1, "Collection Query Results"</a>.</p>
<b>Retrieve Primary Keys</b> <sup>3</sup>	<p>Selects whether or not TopLink retrieves the primary key values within each result. You can use the primary keys to retrieve the real objects.</p> <ul style="list-style-type: none"> <li>■ None—do not retrieve primary keys</li> <li>■ All—retrieve primary keys for each object read;</li> <li>■ First—return only the first primary key value (in the case of a composite primary key). This can be used if you just want to know if something exists or not, but do not really care about the value.</li> </ul>

<sup>1</sup> For more information, see [Section 12.11.5, "How to Use Parameterized SQL \(Parameter Binding\) and Prepared Statement Caching for Optimization"](#).

<sup>2</sup> For ReadObjectQuery and ReadAllQuery queries only.

<sup>3</sup> For ReportQuery queries only.

Click **Advanced** to configure additional options. See [Section 119.7.1.10, "Configuring Named Query Advanced Options"](#).



**See Also**

[Configuring Named Queries at the Descriptor Level](#)

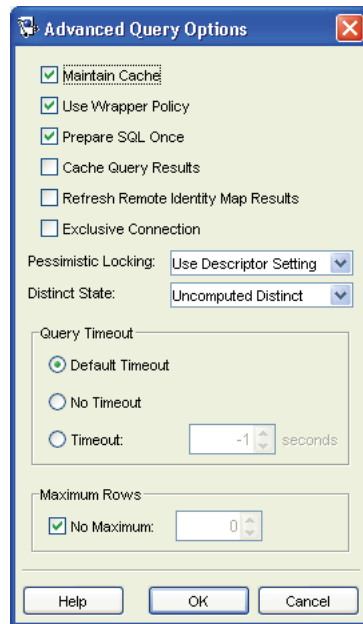
**119.7.1.10 Configuring Named Query Advanced Options**

To configure additional advanced query options, use this procedure.

1. From the Named Queries – Options tab, click **Advanced**. The Advanced Query Options dialog box appears.

From the Named Queries – Options tab, click **Advanced**. The Advanced Query Options dialog box appears.

**Figure 119–19 Advanced Query Options Dialog Box**



2. Complete each field in the Advanced Query Options dialog box.

Use the following information to enter data in each field and click **OK**:

Field	Description
<b>Maintain Cache</b>	Specify whether to use the cache for the query or to build objects directly from the database result. You should only use this option if you are executing a partial object query (see <a href="#">Section 108.7.1.3, "Partial Object Queries"</a> ), whose results cannot be cached.  For more information, see <a href="#">Section 108.16.4, "How to Disable the Identity Map Cache Update During a Read Query"</a> .
<b>Use Wrapper Policy</b>	Specify whether or not the named query will use the wrapper policy configured for this descriptor.  For more information, see <a href="#">Section 119.32, "Configuring Wrapper Policy"</a> .

Field	Description
<b>Prepare SQL Once</b>	Specify the <code>setShouldPrepare()</code> for the named query. By default, TopLink optimizes queries to generate their SQL only once. You may need to disable this option for certain types of queries that require dynamic SQL based on their arguments, such as the following: <ul style="list-style-type: none"> <li>Expressions that use <code>equal</code> where the argument value could be <code>null</code>. This may cause problems on databases that require <code>IS NULL</code>, instead of <code>= NULL</code>.</li> <li>Expressions that use <code>in</code> and use parameter binding. This will cause problems as the <code>in</code> values must be bound individually.</li> </ul>
<b>Cache Query Results</b>	Specify the <code>cacheQueryResults</code> method for the query. The query will only access the database the first time it is executed. Subsequent execution will return exactly the original result.  For more information, see <a href="#">Section 111.13.1, "How to Cache Results in a ReadQuery"</a> .
<b>Refresh Remote Identity Map Results</b>	Specify the <code>refreshRemoteIdentityMapResult</code> method for the query. TopLink can refresh the attributes of the object(s) resulting from the query. With cascading, TopLink will also refresh the private parts of the object(s).
<b>Exclusive Connection</b>	Specify whether or not the named query will use an exclusive connection. You can also configure exclusive connection acquisition at the session level (see <a href="#">Section 89.12, "Configuring Connection Policy"</a> ).
<b>Pessimistic Locking</b>	Specify the specific pessimistic locking policy for the query or use the locking policy from the descriptor.
<b>Distinct State</b>	Specify if TopLink prints the <code>DISTINCT</code> clause, if a distinct has been set. The <code>DISTINCT</code> clause allows you to remove duplicates from the result set.
<b>Query Timeout</b>	Specify if the query will time out (or abort) after a specified number of seconds.
<b>Maximum Rows</b>	Specify if the query will limit the results to a specified number of rows. Use this to option for queries that could return an excessive number of objects.

## 119.7.2 How to Configure Named Queries at the Descriptor Level Using Java

To configure named queries in Java, use a descriptor amendment method (see [Section 119.35, "Configuring Amendment Methods"](#)). [Example 119-1](#) illustrates an amendment method that creates a named query and adds it to the `DescriptorQueryManager`.

### **Example 119-1** Creating a Named Query with an Amendment Method

```
public class EmployeeAmendmentMethodClass {
    ....
    // Create named query with Employee as its reference class
    public static void createEmployeeQuery(ClassDescriptor descriptor) {
        ReadObjectQuery query = new ReadObjectQuery(Employee.class);
        ExpressionBuilder emp = query.getExpressionBuilder();
        Expression firstNameExpression =
            emp.get("firstName").equal(emp.getParameter("firstName"));
        query.setSelectionCriteria(firstNameExpression);
        query.addArgument("firstName");

        descriptor.getQueryManager().addQuery(
```

```

        "employeeReadByFirstName", query);
    }
}

```

## 119.8 Configuring Query Timeout at the Descriptor Level

You can specify how the TopLink runtime handles the duration of queries on a descriptor's reference class. Specifying a query timeout at the descriptor level applies to all queries on the descriptor's reference class. A query timeout ensures that your application does not block forever over a hung or lengthy query that does not return in a timely fashion.

Table 119-4 summarizes which descriptors support query timeout configuration.

**Table 119-9** Descriptor Support for Cache Refresh

Descriptor	How to Use Oracle JDeveloper	How to Configure Query Timeout at the Descriptor Level TopLink Workbench	How to Configure Query Timeout at the Descriptor Level Java
Relational Descriptors <sup>1</sup>	✓	✓	✓
Object-Relational Data Type Descriptors			✓
EIS Descriptor			
XML Descriptors			

<sup>1</sup> Relational class descriptors only (see Section 22.2.1.1, "Creating Relational Class Descriptors").

You can also configure a timeout on a per-query basis. For more information, see the following:

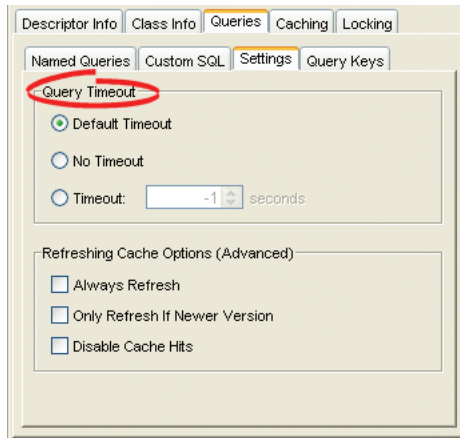
- Section 119.7.1.10, "Configuring Named Query Advanced Options"
- Section 109.2.1.8, "Configuring Query Timeout at the Query Level"

### 119.8.1 How to Configure Query Timeout at the Descriptor Level TopLink Workbench

To configure how TopLink handles the duration of queries to this descriptor, use this procedure:

1. Select a descriptor in the **Navigator**. Its properties appear in the Editor.
2. Click the **Queries** tab. The Queries tab appears.
3. Click the **Settings** tab. The Settings tab appears.

**Figure 119–20 Descriptor Queries Settings Tab, Query Timeout Options**



Use the following table to enter data in the fields on the descriptor’s **Settings** tab to specify how TopLink handles query duration:

Field	Description
<b>Default Timeout</b>	TopLink throws a <code>DatabaseException</code> if a query on this descriptor does not return within the timeout period you configure on the parent descriptor. If there is no parent descriptor, the query timeout defaults to <b>No Timeout</b> .
<b>No Timeout</b>	TopLink blocks until a query on this descriptor returns.
<b>Timeout</b>	Enter the timeout period in seconds. TopLink throws a <code>DatabaseException</code> if a query on this descriptor does not return within this time.

### 119.8.2 How to Configure Query Timeout at the Descriptor Level Java

Use `DescriptorQueryManager` method `setQueryTimeout` passing in the timeout value as a number of milliseconds.

## 119.9 Configuring Cache Refreshing

By default, TopLink caches objects read from a data source (see [Chapter 102, "Introduction to Cache"](#)). Subsequent queries for these objects will access the cache and thus improve performance by reducing data source access and avoiding the cost of rebuilding object's and their relationships. Even if a query, such as a read-all query, accesses the data source, if the objects corresponding to the records returned are in the cache, TopLink will use the cache objects.

This can lead to stale data in the application. Although using an appropriate locking policy (see [Section 119.26, "Configuring Locking Policy"](#)) is the only way to ensure that stale or conflicting data does not get committed to the data source, sometimes certain data in the application changes so frequently that it is desirable to always refresh the data, instead of only refreshing the data when a conflict is detected.

You can specify how the TopLink runtime handles cache refreshing for all queries on a descriptor’s reference class.

[Table 119–4](#) summarizes which descriptors support query cache refresh configuration.

**Table 119–10 Descriptor Support for Query Cache Refresh**

Descriptor	How to Use Oracle JDeveloper	How to Configure Cache Refreshing Using TopLink Workbench	How to Configure Cache Refreshing Using Java
Relational Descriptors <sup>1</sup>	✓	✓	✓
Object-Relational Data Type Descriptors			✓
EIS Descriptor <sup>2</sup>	✓	✓	✓
XML Descriptors			

<sup>1</sup> Relational class descriptors only (see [Section 22.2.1.1](#), "Creating Relational Class Descriptors").

<sup>2</sup> EIS root descriptors only (see [Section 75.2.1.1](#), "EIS Root Descriptors").

Configuring descriptor-level cache refresh may affect performance. As an alternative, consider configuring the following:

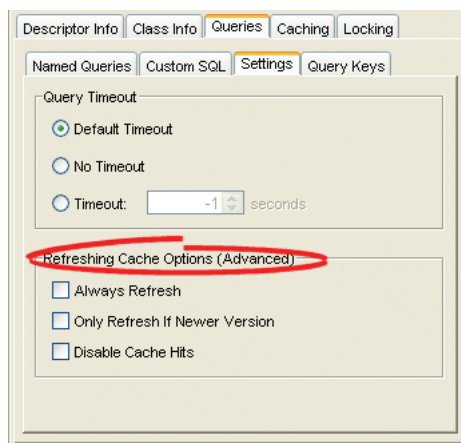
- cache refresh on a query-by-query basis (see [Section 108.16.5](#), "How to Refresh the Cache")
- cache expiration (see [Section 119.16](#), "Configuring Cache Expiration at the Descriptor Level")
- isolated caching (see [Section 102.2.7](#), "Cache Isolation")

For more information, see [Section 12.10](#), "Optimizing Cache".

### 119.9.1 How to Configure Cache Refreshing Using TopLink Workbench

To configure how TopLink refreshes the cache for queries to this descriptor, use this procedure:

1. Select a descriptor in the **Navigator**. Its properties appear in the Editor.
2. Click the **Queries** tab. The Queries tab appears.
3. Click the **Settings** tab. The Settings tab appears.

**Figure 119–21 Descriptor Queries Settings Tab, Cache Refreshing Options**

Use the following table to enter data in the fields on the descriptor's **Settings** tab to specify how TopLink will refresh the cache for queries:

Field	Description
<b>Always Refresh</b>	<p>Refreshes the cache on all queries.</p> <p>Avoids stale data by ensuring that any query that accesses the data source will refresh the resulting objects in the cache. This has no effect on queries that get a cache hit and never access the data source, such as read-object primary key queries or in-memory queries.</p> <p>Configuring descriptor level cache refresh may affect performance. As an alternative, consider configuring:</p> <ul style="list-style-type: none"> <li>▪ cache refresh on a query-by-query basis (see <a href="#">Section 108.16.5, "How to Refresh the Cache"</a>)</li> <li>▪ cache expiration (see <a href="#">Section 119.16, "Configuring Cache Expiration at the Descriptor Level"</a>)</li> <li>▪ isolated caching (see <a href="#">Section 102.2.7, "Cache Isolation"</a>)</li> </ul>
<b>Only Refresh If Newer Version</b>	<p>Refreshes the cache only if the object in the database is newer than the object in the cache (as determined by the <b>Optimistic Locking</b> field). See <a href="#">Section 119.26, "Configuring Locking Policy"</a> for more information.</p> <p>Improves performance by avoiding unnecessary refreshing of an object if its version matches the data source version. This option does not cause refreshing on its own: you must use it in combination with <b>Always Refresh</b>, query refreshing (see <a href="#">Section 108.16.5, "How to Refresh the Cache"</a>), or cache expiration (see <a href="#">Section 119.16, "Configuring Cache Expiration at the Descriptor Level"</a>).</p>
<b>Disable Cache Hits</b>	<p>When selected, TopLink bypasses the cache and goes to the database for read object queries based on primary key. Using this option in conjunction with <b>Always Refresh</b> ensures that TopLink always goes to the database.</p> <p>This option ensures that all queries including read-object primary key queries will always access the data source. This option does not cause refreshing on its own: you must use it in combination with <b>Always Refresh</b>.</p> <p>This option can cause a serious performance issue: avoid whenever possible.</p>

---

**Caution:** Use the **Always Refresh** and **Disable Cache Hits** properties with caution as they may lead to poor performance. As an alternative, consider configuring cache refresh on a query-by-query basis (see [Section 108.16.5, "How to Refresh the Cache"](#)) or configuring cache expiration (see [Section 119.16, "Configuring Cache Expiration at the Descriptor Level"](#)). For more information about cache performance, see [Section 12.10, "Optimizing Cache"](#).

---

## 119.9.2 How to Configure Cache Refreshing Using Java

Configure cache refresh options using the following `ClassDescriptor` methods:

- `setShouldAlwaysRefreshCache`
- `setShouldAlwaysRefreshCacheOnRemote`
- `setShouldDisableCacheHits`
- `setShouldDisableCacheHitsOnRemote`

- `setShouldOnlyRefreshCacheIfNewerVersion`

Use these methods in a descriptor amendment method (see [Section 119.35](#), "Configuring Amendment Methods"), as [Example 119–2](#) illustrates.

**Example 119–2 Configuring Remote Refreshing**

```
public void addToDescriptor(ClassDescriptor descriptor) {
    descriptor.setShouldRefreshCacheOnRemote(true);
    descriptor.setShouldDisableCacheHitsOnRemote(true);
}
```

## 119.10 Configuring Query Keys

A **query key** is a schema-independent alias for a database field name. For example, consider a class `Employee` with attribute `firstName` mapped directly to a database field `F_NAME` in database table `EMPLOYEE`. Without a query key, when you create a query or expression that involves `Employee` attribute `firstName`, you must use the database management system-specific field name `F_NAME`. This makes it more difficult to build a query and ties the query to the schema. With a query key, you can refer to this field using a schema-independent alias, such as `firstName`.

[Table 119–11](#) summarizes which descriptors support query keys.

**Table 119–11 Descriptor Support for Query Keys**

Descriptor	How to Use Oracle JDeveloper	How to Configure Query Keys Using TopLink Workbench	How to Configure Query Keys Using Java
Relational Descriptors	✓	✓	✓
Object-Relational Data Type Descriptors			✓
EIS Descriptors			
XML Descriptors			

Using query keys offers the following advantages:

- Enhances code readability in TopLink expressions and simplifies expression development. You can compose expressions entirely within the context of your object model.
- Increases portability by making code independent of the database schema. If you rename a field in your schema, you can redefine the query key without changing any code that uses it.
- Query keys used with interface descriptors allow the implementor descriptor's tables to have different field names.

Query keys are automatically generated for all mapped attributes. The name of the query key is the name of the class attribute specified in your object model.

For information on how to use query keys in queries and expressions, see [Section 108.2.7](#), "Query Keys".

When query keys are generated and how you can add or modify query keys depends on the type of mapping or descriptor involved:

- [Direct Mappings](#)

- [Relationship Mappings](#)
- [Interface Descriptors](#)

### Direct Mappings

TopLink Workbench automatically generates query keys for all direct mappings at the time you create the mapping.

TopLink Workbench provides support for adding or modifying query keys for simple unmapped attributes that could be mapped by a direct mapping: for example, the `version` field used for optimistic locking or the `type` field used for inheritance. You cannot modify or remove automatically generated query keys.

### Relationship Mappings

TopLink automatically generates query keys for all relationship mappings at run time.

For example, if you have a class `Customer` with attribute `orders` mapped in a one-to-many relationship to class `PurchaseOrders`, then the TopLink runtime will generate a query key named `orders` for this `Customer` attribute.

Neither Oracle JDeveloper nor TopLink Workbench currently support adding or modifying the query keys for relationship mappings. If you must add or modify such a query key, you must do so in Java code, using a descriptor amendment method.

### Interface Descriptors

Interface descriptors (see [Section 22.2.1.3, "Creating Relational Interface Descriptors"](#)) define only the query keys that are shared among their implementors. In the descriptor for an interface, only the name of the query key is specified.

TopLink Workbench provides support for choosing the implementors of an interface that share at least one common automatically generated query key (see [Section 119.11, "Configuring Interface Query Keys"](#)).

## 119.10.1 How to Configure Query Keys Using TopLink Workbench

To add query keys to simple unmapped fields and to view the query keys automatically generated for directly mapped attributes, use this procedure:

1. Select a descriptor in the **Navigator**. Its properties appear in the Editor.
2. Click the **Query Keys** tab in the Editor.

**Figure 119–22** *Queries, Query Keys Tab*



To add a new query key, click **Add**.



To delete an existing query key, select the query key and click **Remove**.

To rename an existing query key, select the query key and click **Rename**.

Use the **Field** list to select the field in the table associated with the query key.

## 119.10.2 How to Configure Query Keys Using Java

To manually create a relationship query key, implement a descriptor amendment method (see [Section 119.35, "Configuring Amendment Methods"](#)) that uses one of the following `ClassDescriptor` methods to register the query keys:

- `addQueryKey`—specify a query key using an instance of `QueryKey` such as `DirectQueryKey`, `DirectCollectionQueryKey`, `ManyToManyQueryKey`, `OneToManyQueryKey`, or `OneToOneQueryKey`.
- `addDirectQueryKey`—add a query key that maps directly to the given database field.
- `addAbstractQueryKey`—add an abstract query key to an interface descriptor. Any implementors of that interface must define the query key defined by this abstract query key.

[Example 119-3](#), [Example 119-4](#), and [Example 119-5](#) illustrate how to define a query key in Java code.

### **Example 119-3 Defining a Query Key**

```
// Add a query key for the foreign key field using the direct method
descriptor.addDirectQueryKey("managerId", "MANAGER_ID");

// The same query key can also be added through the addQueryKey method
DirectQueryKey directQueryKey = new DirectQueryKey();
directQueryKey.setName("managerId");
directQueryKey.setFieldName("MANAGER_ID");
descriptor.addQueryKey(directQueryKey);

// Add a one-to-one query key for the large project of which the
// employee is a leader (this assumes only one project)
OneToOneQueryKey projectQueryKey = new OneToOneQueryKey();
projectQueryKey.setName("managedLargeProject");
projectQueryKey.setReferenceClass(LargeProject.class);
ExpressionBuilder builder = new ExpressionBuilder();
projectQueryKey.setJoinCriteria(builder.getField(
    "PROJECT.LEADER_ID").equal(builder.getParameter("EMPLOYEE.EMP_ID")));
descriptor.addQueryKey(projectQueryKey);
```

### **Example 119-4 Defining a One-to-Many Query Key**

```
// Add a one-to-many query key for the projects where the employee
// manages multiple projects
OneToManyQueryKey projectsQueryKey = new OneToManyQueryKey();
projectsQueryKey.setName("managedProjects");
projectsQueryKey.setReferenceClass(Project.class);
ExpressionBuilder builder = new ExpressionBuilder();
projectsQueryKey.setJoinCriteria(builder.getField(
    "PROJECT.LEADER_ID").equal(builder.getParameter("EMPLOYEE.EMP_ID")));
descriptor.addQueryKey(projectsQueryKey);
```

### **Example 119-5 Defining a Many-to-Many Query Key**

```
// Add a many-to-many query key to an employee project that uses a join table
```

```

ManyToManyQueryKey projectsQueryKey = new ManyToManyQueryKey();
projectsQueryKey.setName("projects");
projectsQueryKey.setReferenceClass(Project.class);
ExpressionBuilder builder = new ExpressionBuilder();
projectsQueryKey.setJoinCriteria(
    builder.getTable("EMP_PROJ").getField("EMP_ID").equal(
        builder.getParameter("EMPLOYEE.EMP_ID").and(
            builder.getTable("EMP_PROJ").getField("PROJ_ID").equal(
                builder.getField("PROJECT.PROJ_ID")));
descriptor.addQueryKey(projectsQueryKey);

```

**Example 119–6** illustrates how to implement a `Descriptor` amendment method to define a one-to-one query key. In this example, the object model for the `Address` class does not include a reference to its owner, an `Employee` object. You can amend the `Address` class descriptor to add a query key named `owner` to make up for this deficiency. At run time, you can compose expressions that select `Address` objects based on this `owner` query key.

**Example 119–6 Defining a One-to-One Query Key with an Amendment Method**

```

// Static amendment method in Address class, addresses do not know
// their owners in the object model, however you can still
// query on their owner if a user-defined query key is defined
public static void addToDescriptor(Descriptor descriptor) {
    OneToOneQueryKey ownerQueryKey = new OneToOneQueryKey();
    ownerQueryKey.setName("owner");
    ownerQueryKey.setReferenceClass(Employee.class);
    ExpressionBuilder builder = new ExpressionBuilder();
    ownerQueryKey.setJoinCriteria(
        builder.getField("EMPLOYEE.ADDRESS_ID").equal(
            builder.getParameter("ADDRESS.ADDRESS_ID")));
    descriptor.addQueryKey(ownerQueryKey);
}

```

## 119.11 Configuring Interface Query Keys

A query key is a schema independent alias for a database field name. For more information about query keys, see [Section 119.10, "Configuring Query Keys"](#).

Interface descriptors (see [Section 22.2.1.3, "Creating Relational Interface Descriptors"](#)) are defined only with query keys that are shared among their implementors. In the descriptor for an interface, only the name of the query key is specified.

In each implementor descriptor, the key must be defined with the appropriate field from one of the implementor descriptor’s tables.

This allows queries and relationship mappings to be defined on the interface using the query key names.

Interface query keys are supported in relational database projects only.

[Table 119–11](#) summarizes which descriptors support interface query keys.

**Table 119–12 Descriptor Support for Interface Query Keys**

Descriptor	How to Use Oracle JDeveloper	How to Configure Interface Query Keys Using TopLink Workbench	How to Configure Interface Query Keys Using Java
Relational Descriptors	✓	✓	✓

**Table 119–12 (Cont.) Descriptor Support for Interface Query Keys**

Descriptor	How to Use Oracle JDeveloper	How to Configure Interface Query Keys Using TopLink Workbench	How to Configure Interface Query Keys Using Java
Object-Relational Data Type Descriptors			✓
EIS Descriptors			
XML Descriptors			

Consider an `Employee` that contains a contact of type `Contact`. The `Contact` class is an interface with two implementors: `Phone` and `Email`. The `Phone` class has attributes `id` and `number`. The `Email` class has attributes `id` and `address`.

Figure 119–23 illustrates the generated keys:

**Figure 119–23 Automatically Generated Query Keys for Phone and Email**

Both classes have an attribute, `id`, that is directly mapped to fields that have different names. However, a query key is generated for this attribute. For the `Contact` interface descriptor, you must indicate that the `id` query key must be defined for each of the implementors.

If either of the implementor classes did not have the `id` query key defined, Oracle JDeveloper and TopLink Workbench would flag that descriptor as deficient.

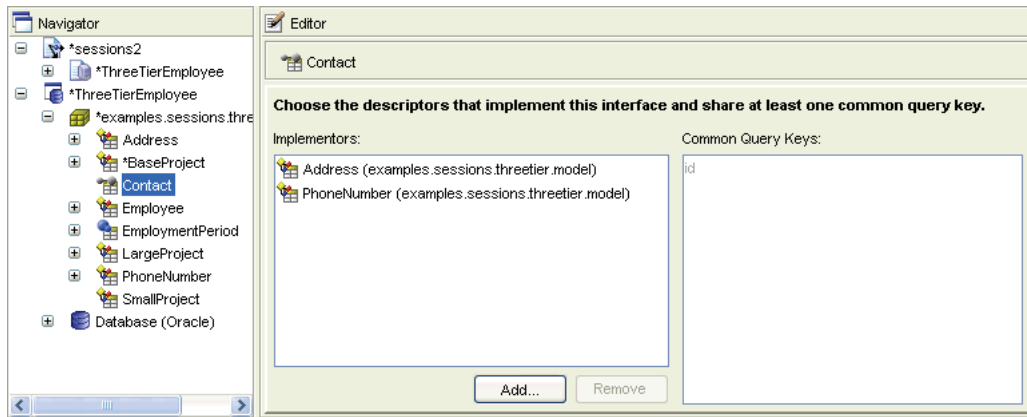
Now that a descriptor with a commonly shared query key has been defined for `Contact`, you can use it as the reference class for a variable one-to-one mapping (see Section 111.8, "Using Queries on Variable One-to-One Mappings").

For example, you can now create a variable one-to-one mapping for the `contact` attribute of `Employee`. When you edit the foreign key field information for the mapping, you must match the `Employee` descriptor's tables to query keys from the `Contact` interface descriptor.

### 119.11.1 How to Configure Interface Query Keys Using TopLink Workbench

To choose the implementors of an interface that share at least one common automatically generated query key, use this procedure.

1. Select an interface descriptor in the **Navigator**. Its properties appear in the Editor.

**Figure 119–24 Interface Descriptor Editor Window**

To choose an implementor of the selected interface that shares at least one common query key, click **Add**.

To remove an implementor of the selected interface, select the implementor and click **Remove**.

## 119.11.2 How to Configure Interface Query Keys Using Java

[Example 119–7](#) shows how to define the `Contact` interface and `Email` and `Phone` implementors in Java.

### **Example 119–7 Defining Interface Query Keys**

```
Descriptor contactInterfaceDescriptor = new Descriptor();
contactInterfaceDescriptor.setJavaInterface(Contact.class);
contactInterfaceDescriptor.addAbstractQueryKey("id");
```

```
Descriptor emailClassDescriptor = new Descriptor();
emailClassDescriptor.setJavaClass(Email.class);
emailClassDescriptor.addDirectQueryKey("id", "E_ID");
emailClassDescriptor.getInterfacePolicy().addParentInterface(Contact.class);
emailClassDescriptor.setTableName("INT_EML");
emailClassDescriptor.setPrimaryKeyFieldName("E_ID");
emailClassDescriptor.setSequenceNumberName("SEQ");
emailClassDescriptor.setSequenceNumberFieldName("E_ID");
emailClassDescriptor.addDirectMapping("emailID", "E_ID");
emailClassDescriptor.addDirectMapping("address", "ADDR");
```

```
Descriptor phoneClassDescriptor = new Descriptor();
phoneClassDescriptor.setJavaClass(Phone.class);
phoneClassDescriptor.getInterfacePolicy().addParentInterface(Contact.class);
phoneClassDescriptor.addDirectQueryKey("id", "P_ID");
phoneClassDescriptor.setTableName("INT_PHN");
phoneClassDescriptor.setPrimaryKeyFieldName("P_ID");
phoneClassDescriptor.setSequenceNumberName("SEQ");
phoneClassDescriptor.setSequenceNumberFieldName("P_ID");
phoneClassDescriptor.addDirectMapping("phoneID", "P_ID");
phoneClassDescriptor.addDirectMapping("number", "P_NUM");
```

## 119.12 Configuring Cache Type and Size at the Descriptor Level

The `TopLink` cache is an in-memory repository that stores recently read or written objects based on class and primary key values. `TopLink` uses the cache to do the following:

- improve performance by holding recently read or written objects and accessing them in-memory to minimize database access;
- manage locking and isolation level;
- manage object identity.

Table 119–13 summarizes which descriptors support identity map configuration.

**Table 119–13 Descriptor Support for Identity Map**

Descriptor	How to Use Oracle JDeveloper	How to Configure Cache Type and Size at the Descriptor Level Using TopLink Workbench	How to Configure Cache Type and Size at the Descriptor Level Using Java
Relational Descriptors <sup>1</sup>	✓	✓	✓
Object-Relational Data Type Descriptors			✓
EIS Descriptors <sup>2</sup>	✓	✓	✓
XML Descriptors			

<sup>1</sup> Relational class descriptors only (see Section 22.2.1.1, "Creating Relational Class Descriptors").

<sup>2</sup> EIS root descriptors only (see Section 75.2.1.1, "EIS Root Descriptors").

This configuration overrides the default identity map configuration defined at the project level (see Section 117.10, "Configuring Cache Type and Size at the Project Level").

For detailed information on caching and object identity, and the recommended settings to maximize TopLink performance, see to Section 102.2.1, "Cache Type and Object Identity".

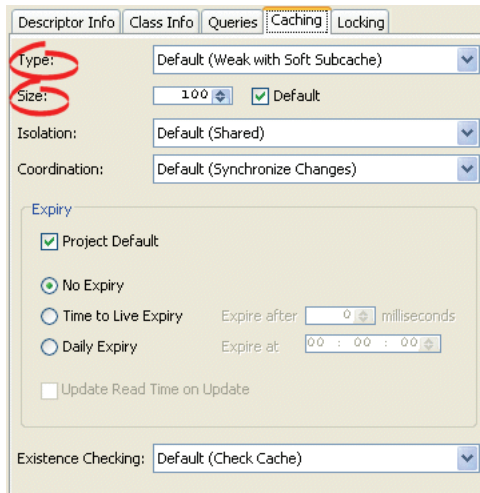
For more information about the cache, see Chapter 102, "Introduction to Cache".

### 119.12.1 How to Configure Cache Type and Size at the Descriptor Level Using TopLink Workbench

To specify the identity map information for a descriptor, use this procedure:

1. Select the descriptor in the **Navigator**.
2. Select the **Caching** tab in the **Editor**. The Caching tab appears.

**Figure 119–25 Caching Tab, Identity Map Options**



Use the following table to enter data in following fields on the **Caching** tab:

Field	Description
Type <sup>1</sup>	<p>Use the <b>Type</b> list to choose the identity map as follows:</p> <ul style="list-style-type: none"> <li> <b>Weak with Soft Subcache</b>                      (SoftCacheWeakIdentityMap)—cache first <i>n</i> elements in soft space, anything after that in weak space (see <a href="#">Section 102.2.1.4, "Soft Cache Weak Identity Map and Hard Cache Weak Identity Map"</a>)                 </li> <li> <b>Weak with Hard Subcache</b>                      (HardCacheWeakIdentityMap)—cache first <i>n</i> elements in hard space, anything after that in weak space (see <a href="#">Section 102.2.1.4, "Soft Cache Weak Identity Map and Hard Cache Weak Identity Map"</a>)                 </li> <li> <b>Weak</b> (WeakIdentityMap)—cache everything in weak space (see <a href="#">Section 102.2.1.2, "Weak Identity Map"</a>)                 </li> <li> <b>Soft</b> (SoftIdentityMap)—cache everything in soft space (see <a href="#">Section 102.2.1.3, "Soft Identity Map"</a>)                 </li> <li> <b>Full</b> (FullIdentityMap)—cache everything permanently (see <a href="#">Section 102.2.1.1, "Full Identity Map"</a>)                 </li> <li> <b>None</b> (NoIdentityMap)—cache nothing (see <a href="#">Section 102.2.1.5, "No Identity Map"</a>)                 </li> </ul> <p>For more information, see <a href="#">Section 102.2.1, "Cache Type and Object Identity"</a>.</p> <p>Changing the project's default identity map does not affect descriptors that already exist in the project.</p>
Size <sup>1</sup>	<p>Specify the size of the cache as follows:</p> <ul style="list-style-type: none"> <li>When using <b>Weak with Soft Subcache</b> or <b>Weak with Hard Subcache</b>, the size is the size of the subcache.</li> <li>When using <b>Full</b> or <b>Weak</b>, the size indicates the <i>starting size</i> of the identity map.</li> </ul>
Default	<p>When you enter a cache size, the <b>Default</b> check box is cleared. To reset the size to the default for the selected cache type, check the <b>Default</b> check box.</p>

<sup>1</sup> If a descriptor is a child in an inheritance hierarchy, TopLink makes this field read only and displays the options from the parent root descriptor. For more information, see [Section 16.2.3.3, "Inheritance"](#).

## 119.12.2 How to Configure Cache Type and Size at the Descriptor Level Using Java

Use one of the following `ClassDescriptor` methods to configure the descriptor to use the appropriate type of identity map:

- `useFullIdentityMap`
- `useWeakIdentityMap`
- `useSoftIdentityMap`
- `useSoftCacheWeakIdentityMap`
- `useHardCacheWeakIdentityMap`
- `useNoIdentityMap`

Use the `ClassDescriptor` method `setIdentityMapSize` to configure the size of the identity map.

## 119.13 Configuring Cache Isolation at the Descriptor Level

If you plan to use isolated sessions (see [Section 102.2.7, "Cache Isolation"](#)), you must configure descriptors as isolated for any object that you want confined to an isolated session cache.

Configuring a descriptor to be isolated means that `TopLink` will not store the object in the shared session cache and the object will not be shared across client sessions. Each client will have their own object read directly from the database. Objects in an isolated client session cache can reference objects in their parent server session's shared session cache, but no objects in the shared session cache can reference objects in an isolated client session cache. Isolation is required when using Oracle Database Virtual Private Database (VPD) support or database user-based read security. Isolation can also be used if caching is not desired across client sessions.

[Table 119–13](#) summarizes which descriptors support cache isolation configuration.

**Table 119–14** *Descriptor Support for Cache Isolation Map*

Descriptor	How to Use Oracle JDeveloper	How to Configure Cache Isolation at the Descriptor Level Using TopLink Workbench	How to Configure Cache Isolation at the Descriptor Level Using Java
Relational Descriptors <sup>1</sup>	✓	✓	✓
Object-Relational Data Type Descriptors			✓
EIS Descriptors <sup>2</sup>	✓	✓	✓
XML Descriptors			

<sup>1</sup> Relational class descriptors only (see [Section 22.2.1.1, "Creating Relational Class Descriptors"](#)).

<sup>2</sup> EIS root descriptors only (see [Section 75.2.1.1, "EIS Root Descriptors"](#)).

This configuration overrides the default cache isolation configuration defined at the project level (see [Section 117.11, "Configuring Cache Isolation at the Project Level"](#)).

---

**Note:** If you configure a descriptor as isolated, it cannot participate in a coordinated cache (see [Section 119.15, "Configuring Cache Coordination Change Propagation at the Descriptor Level"](#)).

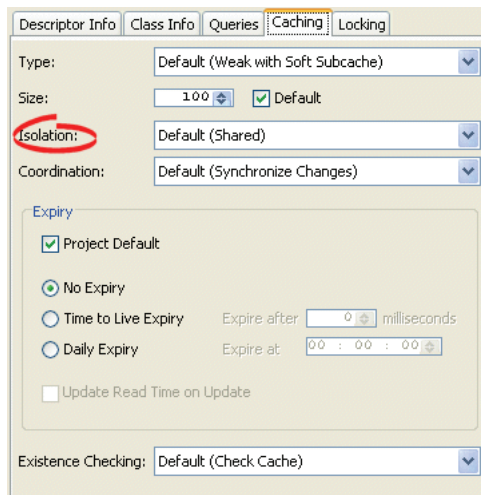
---

### 119.13.1 How to Configure Cache Isolation at the Descriptor Level Using TopLink Workbench

To specify the cache isolation options, use this procedure:

1. Select the descriptor in the **Navigator**.
2. Select the **Caching** tab in the **Editor**. The Caching tab appears.

**Figure 119–26 Caching Tab, Isolation Options**



Use the **Isolation** list to choose one of the following:

- **Isolated**—if you want all objects confined to an isolated client session cache. For more information, see [Section 102.2.7, "Cache Isolation"](#).
- **Shared**—if you want all objects visible in the shared session cache (default).

### 119.13.2 How to Configure Cache Isolation at the Descriptor Level Using Java

To specify that a class is isolated, use a descriptor amendment method (see [Section 119.35, "Configuring Amendment Methods"](#)) to call `ClassDescriptor` method `setIsIsolated`, passing in a boolean of `true`.

## 119.14 Configuring Unit of Work Cache Isolation at the Descriptor Level

Use this policy to determine how a unit of work uses a session cache for a specific class. [Table 119–15](#) lists the unit of work cache isolation options.



**Table 119–15 Unit of Work Cache Isolation Options**

Option	Description
Using the Session Cache After the Transaction	<p>USE_SESSION_CACHE_AFTER_TRANSACTION</p> <p>Objects built from new data accessed after a unit of work early transaction are stored in the session cache.</p> <p>This options is the most efficient as it allows the cache to be used after an early transaction.</p>
Isolating New Data After the Transaction	<p>ISOLATE_NEW_DATA_AFTER_TRANSACTION (default)</p> <p>Objects built from new data accessed after a unit of work early transaction are only stored in the unit of work.</p> <p>This still allows previously cached objects to be accessed in the unit of work after an early transaction, while ensuring that uncommitted data will never be put in the session cache by storing any object built from new data only in the unit of work</p>
Isolating the Cache after the Transaction	<p>ISOLATE_CACHE_AFTER_TRANSACTION</p> <p>After a unit of work early transaction the session cache is no longer used for this class. Objects are directly built from the database data and only stored in the unit of work, <i>even if previously cached</i>.</p> <p>Note: This option may affect performance because you are bypassing the session cache after an early transaction.</p>
Always Isolating the Cache	<p>ISOLATE_CACHE_ALWAYS</p> <p>The session cache will <i>never</i> be used for the class. Objects are directly built from the database data and only stored in the unit of work. New objects and changes will never be merged into the session cache.</p> <p>Note: This option my affect performance because you are bypassing the session cache. However if this class is isolated or pessimistic locked and always accessed in a transaction, this can avoid having to build two copies of the object.</p>

Most of these options apply only to a unit of work in an early transaction, such as the following:

- A unit of work that was flushed (`write changes`).
- Issued a modify query.
- Acquired a pessimistic lock.

### 119.14.1 How to Configure Unit of Work Cache Isolation at the Descriptor Level Using Java

To specify that a class is isolated, use a descriptor amendment method (see [Section 119.35, "Configuring Amendment Methods"](#)) to call `ClassDescriptor` method `setUnitOfWorkCacheIsolationLevel`.

## 119.15 Configuring Cache Coordination Change Propagation at the Descriptor Level

If you plan to use a coordinated cache (see [Section 102.3, "Cache Coordination"](#)), you can configure how, and under what conditions, a coordinated cache propagates changes for a given descriptor.

[Table 119–13](#) summarizes which descriptors support cache isolation configuration.

**Table 119–16 Descriptor Support for Cache Coordination Change Propagation Configuration**

Descriptor	How to Use Oracle JDeveloper	How to Configure Cache Coordination Change Propagation at the Descriptor Level Using TopLink Workbench	How to Configure Cache Coordination Change Propagation at the Descriptor Level Using Java
Relational Descriptors <sup>1</sup>	✓	✓	✓
Object-Relational Data Type Descriptors			✓
EIS Descriptors <sup>2</sup>	✓	✓	✓
XML Descriptors			

<sup>1</sup> Relational class descriptors only (see Section 22.2.1.1, "Creating Relational Class Descriptors").

<sup>2</sup> EIS root descriptors only (see Section 75.2.1.1, "EIS Root Descriptors").

This configuration overrides the default cache coordination change propagation configuration defined at the project level (see Section 117.12, "Configuring Cache Coordination Change Propagation at the Project Level").

To complete your coordinated cache configuration, see Chapter 103, "Configuring a Coordinated Cache".

---

**Note:** If you configure a descriptor as isolated (see Section 119.13, "Configuring Cache Isolation at the Descriptor Level"), it cannot participate in a coordinated cache.

---

### 119.15.1 How to Configure Cache Coordination Change Propagation at the Descriptor Level Using TopLink Workbench

To specify the coordinated cache change propagation options, use this procedure:

1. Select the descriptor in the **Navigator**.
2. Select the **Caching** tab in the **Editor**. The Caching tab appears.

**Figure 119–27 Caching Tab, Coordination Options**

The screenshot shows the 'Caching' tab of a configuration window. The 'Coordination' dropdown menu is highlighted with a red circle. The 'Expiry' section has 'Project Default' checked, and 'Update Read Time on Update' is unchecked.

Use the following information to enter data in the **Coordination** field:

Coordination Option	Description	When to Use
<b>None</b>	For both existing and new instances, do not propagate a change notification.	Infrequently read or changed objects.
<b>Synchronize Changes</b>	For an existing instance, propagate a change notification that contains each changed attribute. For a new instance, propagate an object creation (along with all the new instance's attributes) only if the new instance is related to other existing objects that are also configured with this change propagation option.	Frequently read or changed objects that contain few attributes or in cases where only a few attributes are frequently changed. Objects that have many or complex relationships.
<b>Synchronize Changes and New Objects</b>	For an existing instance, propagate a change notification that contains each changed attribute. For a new instance, propagate an object creation (along with all the new instance's attributes).	Frequently read or changed objects that contain few attributes or in cases where only a few attributes are frequently changed. Objects that have few or simple relationships.
<b>Invalidate Changed Objects</b>	For an existing instance, propagate an object invalidation that marks the object as invalid in all other sessions. This tells other sessions that they must update their cache from the data source the next time this object is read. For a new instance, no change notification is propagated.	Frequently read or changed objects that contain many attributes in cases where many of the attributes are frequently changed.

### 119.15.2 How to Configure Cache Coordination Change Propagation at the Descriptor Level Using Java

Use a descriptor amendment method (see [Section 119.35, "Configuring Amendment Methods"](#)) to invoke `ClassDescriptor` method `setCacheSynchronizationType` passing in one of the following parameters:

- `ClassDescriptor.DO_NOT_SEND_CHANGES`
- `ClassDescriptor.SEND_OBJECT_CHANGES`
- `ClassDescriptor.SEND_NEW_OBJECTS_WITH_CHANGES`

- `ClassDescriptor.INVALIDATE_CHANGED_OBJECTS`

## 119.16 Configuring Cache Expiration at the Descriptor Level

By default, objects remain in the cache until they are explicitly deleted (see [Section 114.7, "Deleting Objects"](#)) or garbage-collected when using a weak identity map (see [Section 117.11, "Configuring Cache Isolation at the Project Level"](#)). Alternatively, you can configure an object with a `CacheInvalidationPolicy` that allows you to specify, either automatically or manually, that an object is invalid: when any query attempts to read an invalid object, TopLink will go to the data source for the most up-to-date version of that object and update the cache with this information.

Using cache invalidation ensures that your application does not use stale data. It provides a better performing alternative to always refreshing (see [Section 119.9, "Configuring Cache Refreshing"](#)).

Table 119–17 summarizes which descriptors support a cache invalidation policy.

**Table 119–17 Descriptor Support for Cache Invalidation Policy**

Descriptor	How to Use Oracle JDeveloper	How to Configure Cache Expiration at the Descriptor Level Using TopLink Workbench	How to Configure Cache Expiration at the Descriptor Level Using Java
Relational Descriptors <sup>1</sup>	✓	✓	✓
Object-Relational Data Type Descriptors			✓
EIS Descriptors <sup>2</sup>	✓	✓	✓
XML Descriptors			

<sup>1</sup> Relational class descriptors only (see [Section 22.2.1.1, "Creating Relational Class Descriptors"](#)).

<sup>2</sup> EIS root descriptors only (see [Section 75.2.1.1, "EIS Root Descriptors"](#)).

You can override the project-level cache invalidation configuration (see [Section 117.13, "Configuring Cache Expiration at the Project Level"](#)) by defining cache invalidation at the descriptor or query level (see [Section 111.13.2, "How to Configure Cache Expiration at the Query Level"](#)).

You can customize how TopLink communicates the fact that an object has been declared invalid to improve efficiency, if you are using a coordinated cache. For more information, see [Section 119.15, "Configuring Cache Coordination Change Propagation at the Descriptor Level"](#).

For more information, see [Section 102.2.5, "Cache Invalidation"](#).

### 119.16.1 How to Configure Cache Expiration at the Descriptor Level Using TopLink Workbench

To specify the cache invalidation information for a descriptor, use this procedure:

1. Select the descriptor in the **Navigator**.
2. Select the **Caching** tab in the **Editor**. The **Caching** tab appears.

**Figure 119–28 Caching Tab, Expiration Options**

Use this table to enter data in the following fields on the **Caching** tab to specify the cache invalidation policy for the descriptors.

Field	Description
<b>Project Default</b>	Use the project's cache expiration options for this descriptor. See <a href="#">Section 117.13, "Configuring Cache Expiration at the Project Level"</a> for more information.
<b>No Expiry</b>	Specify that objects in the cache do not expire.
<b>Time to Live Expiry</b>	Specify that objects in the cache will expire after a specified amount of time. Use the <b>Expire After</b> field to indicate the time (in milliseconds) after which the objects will expire.
<b>Daily Expiry</b>	Specify that objects in the cache will expire at a specific time each day. Use the <b>Expire At</b> field to indicate the exact time to the second (using a 24-hour clock) at which the objects will expire.
<b>Update Read Time on Update</b>	Specify if TopLink should reset the cached object's expiry time after the TopLink successfully updates the object.

---

**Note:** These options apply per descriptor. See [Section 117.13, "Configuring Cache Expiration at the Project Level"](#) for information on configuring project-level options.

---

### 119.16.2 How to Configure Cache Expiration at the Descriptor Level Using Java

Use `ClassDescriptor` method `setCacheInvalidationPolicy` to set an appropriate instance of `CacheInvalidationPolicy`.

## 119.17 Configuring Cache Existence Checking at the Descriptor Level

When TopLink writes an object to the database, TopLink runs an existence check to determine whether to perform an insert or an update.

By default, TopLink checks against the cache. Oracle recommends that you use this default existence check option for most applications. Checking the database for existence can cause a performance bottleneck in your application.

Table 119–18 summarizes which descriptors support existence checking.

**Table 119–18 Descriptor Support for Existence Checking**

Descriptor	How to Use Oracle JDeveloper	How to Configure Cache Existence Checking at the Descriptor Level Using TopLink Workbench	How to Configure Cache Existence Checking at the Descriptor Level Using Java
Relational Descriptors <sup>1</sup>	✓	✓	✓
Object-Relational Data Type Descriptors			✓
EIS Descriptors <sup>2</sup>	✓	✓	✓
XML Descriptors			

<sup>1</sup> Relational class descriptors only (see Section 22.2.1.1, "Creating Relational Class Descriptors").

<sup>2</sup> EIS root descriptors only (see Section 75.2.1.1, "EIS Root Descriptors").

You can configure existence checking at the descriptor level to override the project level configuration (see Section 117.7, "Configuring Existence Checking at the Project Level").

For more information see the following:

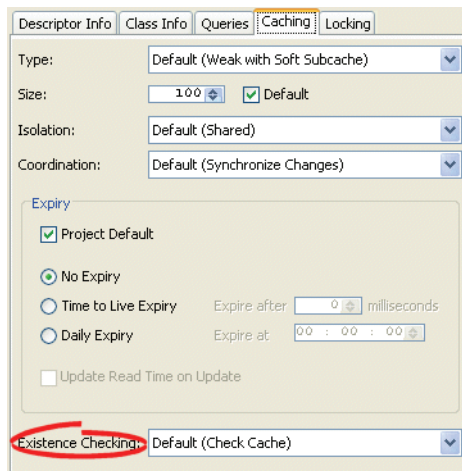
- Section 102.2.1, "Cache Type and Object Identity"
- Section 108.16, "Queries and the Cache"
- Section 115.1.3, "How to Use Registration and Existence Checking"

### 119.17.1 How to Configure Cache Existence Checking at the Descriptor Level Using TopLink Workbench

To specify the existence checking information for a descriptor, use this procedure:

1. Select the descriptor in the **Navigator**.
2. Select the **Caching** tab in the **Editor**. The Caching tab appears.

**Figure 119–29 Caching Tab, Existence Checking Options**



Use this table to enter data in the following fields of the tab to specify the existence checking options for newly created descriptors:

Field	Description
Check Cache	Check the session cache. If the object is not in the cache, assume that the object does not exist (do an insert). If the object is in the cache, assume that the object exists (do an update). Oracle recommends using this option for most applications.
Check Cache then Database	If an object is not in the cache, query the database to determine if the object exists. If the object exists, do an update. Otherwise, do an insert. Selecting this option may negatively impact performance. For more information, see <a href="#">Section 115.1.3.1, "Using Check Database"</a> .
Assume Existence	Always assume objects exist: always do an update (never do an insert). For more information, see <a href="#">Section 115.1.3.2, "Using Assume Existence"</a> .
Assume Non-Existence	Always assume objects do not exist: always do an insert (never do an update). For more information, see <a href="#">Section 115.1.3.3, "Using Assume Nonexistence"</a> .

## 119.17.2 How to Configure Cache Existence Checking at the Descriptor Level Using Java

To configure existence checking at the descriptor level using Java, use `ClassDescriptor` method `getQueryManager` to acquire the `DescriptorQueryManager` from the descriptor and then use one of the following `DescriptorQueryManager` methods (see [Example 119–8](#)):

- `checkCacheForDoesExist`—check the session cache. If the object is not in the cache, assume that the object does not exist (do an insert). If the object is in the cache, assume that the object exists (do an update). Oracle recommends using this option for most applications.
- `checkDatabaseForDoesExist`—if an object is not in the cache, query the database to determine if the object exists. If the object exists, do an update. Otherwise, do an insert. Selecting this option may negatively impact performance. For more information, see [Section 115.1.3.1, "Using Check Database"](#).
- `assumeExistenceForDoesExist`—always assume objects exist: always do an update (never do an insert). For more information, see [Section 115.1.3.2, "Using Assume Existence"](#).
- `assumeNonExistenceForDoesExist`—always assume objects do not exist: always do an insert (never do an update). For more information, see [Section 115.1.3.3, "Using Assume Nonexistence"](#).

### **Example 119–8 Configuring Existence Checking Using Java**

```
descriptor.getQueryManager().checkCacehForDoesExist();
```

## 119.18 Configuring a Descriptor with EJB CMP and BMP Information

If your project uses EJB CMP or BMP (see [Section 117.5, "Configuring Persistence Type"](#)), you can use descriptors to describe the characteristics of entity beans with container-managed or bean-managed persistence.

[Table 119–19](#) summarizes which descriptors support EJB information.

**Table 119–19 Descriptor Support for EJB Information**

Descriptor	How to Use Oracle JDeveloper	How to Configure a Descriptor with EJB CMP and BMP Information Using TopLink Workbench	How to Configure a Descriptor with EJB CMP and BMP Information Using Java
Relational Descriptors <sup>1</sup>	✓	✓	✓
Object-Relational Data Type Descriptors			
EIS Descriptors <sup>2</sup>	✓	✓	✓
XML Descriptors			

<sup>1</sup> Relational class descriptors only (see [Section 22.2.1.1, "Creating Relational Class Descriptors"](#)).

<sup>2</sup> EIS root descriptors only (see [Section 75.2.1.1, "EIS Root Descriptors"](#)).

When mapping enterprise beans, you create a descriptor for the bean class; you do not create a descriptor for the local interface, remote interface, home class, or primary key class.

When using TopLink Workbench, you must define the project with the correct entity bean type (such as container-managed or bean-managed persistence) and import the `ejb-jar.xml` file for the beans into the TopLink Workbench project.

For CMP projects, the `ejb-jar.xml` file defines the bean's attributes to be mapped. A descriptor for an entity bean container-managed persistence contains a CMP policy used to configure CMP-specific options.

For more information, see [Section 16.2.3, "Descriptors and CMP and BMP"](#).

### 119.18.1 How to Configure a Descriptor with EJB CMP and BMP Information Using TopLink Workbench

To configure a descriptor with EJB information, use this procedure:

1. In the **Navigator**, select a relational descriptor.
2. Click **EJB Descriptor** on the mapping toolbar.



An EJB Info tab is added to the descriptor.

To remove the EJB information for the selected descriptor, click **EJB Descriptor** again.

The EJB Info tab is removed from the descriptor.

3. Click the **EJB Info** tab in the **Editor**



**Figure 119–30 EJB Info Tab**

Use the following information to enter data in each field on the tab:

Field	Description
<b>EJB Name</b>	Enter the bean's base name. This is specified in the <b>&lt;ejb-name&gt;</b> element of the <code>ejb-jar.xml</code> file and is for display only.
<b>Primary Key Class</b>	Enter the primary key. This is specified in the <b>&lt;prim-key-class&gt;</b> element of the <code>ejb-jar.xml</code> file and is for display only.
<b>Unknown Primary Key Class</b>	Check this option if you choose not to specify the primary key class or the primary key fields for an entity bean with container managed persistence.  For example, select this field if the entity bean does not have a natural primary key or you want the deployer to select the primary key fields at deployment time. For more information, see <a href="#">Section 8.9.2, "How to Configure EJB CMP Unknown Primary Key Class Support"</a> .
<b>Local Interface</b>	Enter the local interface. This is specified in the <b>&lt;local&gt;</b> element of the <code>ejb-jar.xml</code> file and is for display only.
<b>Local Home Interface</b>	Enter the local home interface. This is specified in the <b>&lt;local-home&gt;</b> element of the <code>ejb-jar.xml</code> file and is for display only.
<b>Remote Interface</b>	Enter the remote interface. This is specified in the <b>&lt;remote&gt;</b> element of the <code>ejb-jar.xml</code> file and is for display only.
<b>Remote Home Interface</b>	Enter the remote interface. This is specified in the <b>&lt;home&gt;</b> element of the <code>ejb-jar.xml</code> file and is for display only.
<b>Change Deferral</b>	Use these options to specify how TopLink updates the database for this EJB descriptor.
<b>Defer All Changes</b>	Specify not to send changes to the database until the JTA transaction is committed. This is the default TopLink behavior. This is the most efficient option that results in the least amount of data source interaction.

Field	Description
<b>Defer Updates Only</b>	<p>Specify to send changes to the database immediately after any insert or delete operation, but do not send changes to the data source for update operations until the JTA transaction is committed.</p> <p>Select this option for backwards compatibility with some EJB containers, such as OC4J. For more information, see <a href="#">Section 16.2.3.1, "Nondeferred Changes"</a>.</p> <p>Use this option with caution as it will require the data source transaction and locks to be held longer and may cause referential integrity issues.</p>
<b>Defer None</b>	<p>Specify to send all changes to the database immediately. This is the least efficient option that generates the greatest amount of data source interaction.</p> <p>Select this option for backwards compatibility with some EJB containers, such as OC4J. For more information, see <a href="#">Section 16.2.3.1, "Nondeferred Changes"</a>.</p>
<b>Insert New Objects After</b>	<p>Specify to send new object insert changes to the database after bean life cycle method <code>ejbCreate</code> (default) or <code>ejbPostCreate</code>. This is only relevant when not deferring changes (see <a href="#">Section 119.30, "Configuring Change Policy"</a>).</p> <p>If non-null foreign key constraints cannot be satisfied when the insert is performed after <code>ejbCreate</code>, you may consider configuring <code>TopLink</code> CMP to do the insert after <code>ejbPostCreate</code>, if supported by your container. For more information, see <a href="#">Section 16.2.3.2, "Creating a New Entity Bean and <code>ejbCreate</code> / <code>ejbPostCreate</code> Methods"</a>.</p>

## 119.18.2 How to Configure a Descriptor with EJB CMP and BMP Information Using Java

Using Java code, you can use descriptors to describe the characteristics of entity beans with container-managed (see [Section 119.18.2.1, "Configuring CMP Information"](#)) or bean-managed (see [Section 119.18.2.2, "Configuring BMP Information"](#)) persistence.

### 119.18.2.1 Configuring CMP Information

To configure CMP-specific information on a descriptor, define a `CMPPolicy`:

```
descriptor.setCMPPolicy(new CMPPolicy());
```

You can use the following `CMPPolicy` API to configure optional behavior of enterprise beans:

---

**Note:** Most of these options are provided for compatibility with other CMP implementations. Use caution when using them as they will affect application performance.

---

- `setDeferModificationsUntilCommit`—By default `TopLink` defers all changes to the database until the transaction is committed. Use this method to configure `TopLink` to update the database after each EJB operation for the specified deferral level:
  - `CMPPolicy.NONE`—default behavior
  - `CMPPolicy.UPDATE_MODIFICATIONS`—update the database after each EJB operation for update modifications only

- `CMPPolicy.ALL_MODIFICATIONS`—update the database after each EJB operation for all modifications
- `setNonDeferredCreateTime`—when using nondeferred writes (see `setDeferModificationsUntilCommit`), use this method to configure `TopLink` to insert a new enterprise bean before (`CMPPolicy.AFTER_EJB_CREATE`) or after (`CMPPolicy.AFTER_EJB_POST_CREATE`) the `ejbPostCreate` method.
- `setForceUpdate`—use this method to make `TopLink` write all enterprise beans that have been accessed to the database regardless of whether they changed or not.
- `setUpdateAllFields`—use this method to configure `TopLink` to force all the fields of the bean to be updated instead of only the changed fields.
- `setPessimisticLockingPolicy`—use this method to configure EJB-level pessimistic locking.

### 119.18.2.2 Configuring BMP Information

BMP descriptors must be configured with a `BMPWrapperPolicy`. Neither Oracle JDeveloper nor `TopLink Workbench` currently support defining the `BMPWrapperPolicy`, so you must define this through Java code.

For more information, see [Section 119.32, "Configuring Wrapper Policy"](#).

## 119.19 Configuring Reading Subclasses on Queries

If you are mapping an inheritance hierarchy, by default, queries on root or branch classes return instances of the root class and their subclasses.

Alternatively, you can configure a root or branch class descriptor to do the following:

- not include subclasses when the root or branch class is queried;
- outer-join subclasses when the root or branch class is queried.

You can also specify a database view to optimize the reading of subclasses. The view can be used to optimize queries for root or branch classes that have subclasses spanning multiple tables. The view must apply an outer-join or union all of the subclass tables.

Do not configure this option for leaf classes.

[Table 119–20](#) summarizes which descriptors support inherited attribute mapping configuration.

**Table 119–20 Descriptor Support for Inherited Attribute Mapping Configuration**

Descriptor	How to Use Oracle JDeveloper	How to Configure Reading Subclasses on Queries Using TopLink Workbench	How to Configure Reading Subclasses on Queries Using Java
Relational Descriptors	✓	✓	✓
Object-Relational Data Type Descriptors			✓
EIS Descriptors			
XML Descriptors			

For more information, see [Section 16.2.2, "Descriptors and Inheritance"](#).

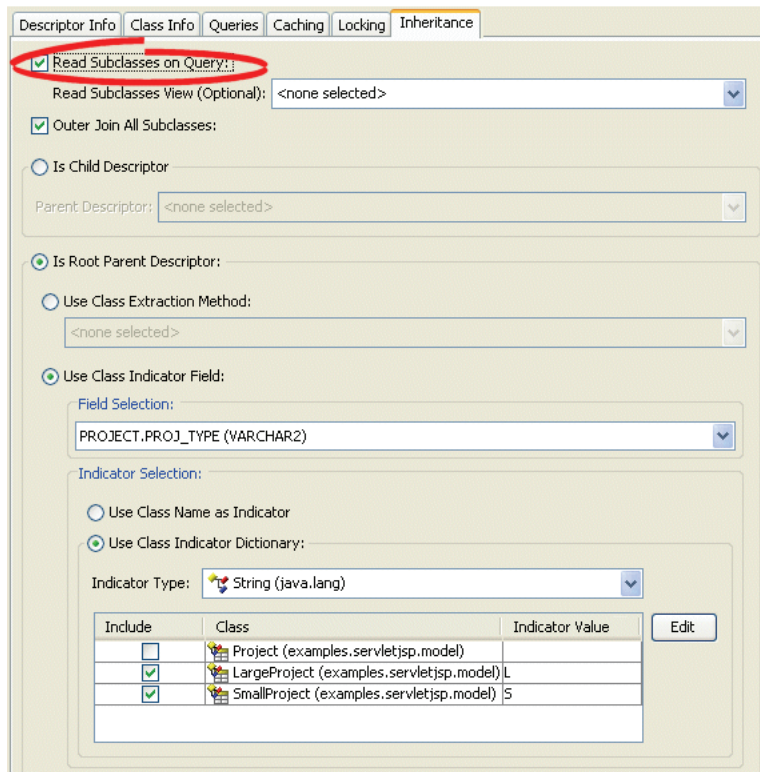
### 119.19.1 How to Configure Reading Subclasses on Queries Using TopLink Workbench

To configure reading classes on subqueries, use this procedure:

1. In the **Navigator**, select a root or branch descriptor.
 

If the **Inheritance** advanced property is not visible for the descriptor, right-click the descriptor and choose **Select Advanced Properties > Inheritance** from context menu or from the **Selected** menu.
2. Click the **Inheritance** tab.

**Figure 119–31 Inheritance Tab, Read Subclasses on Query Option**



Use the following information to enter data in Read Subclasses on Query and Read Subclasses View fields of the tab:

Field	Description
<b>Read Subclasses on Query</b>	Select this option to configure the root class descriptor to instantiate a subclass when the root class is queried.
<b>Read Subclasses View</b>	Optionally select a database view to use for reading subclasses.
<b>Outer Join All Subclasses</b>	Optionally use the <code>outerJoinAllSubclasses</code> option to optimize the query.

### 119.19.2 How to Configure Reading Subclasses on Queries Using Java

Create a descriptor amendment method ([Section 119.35, "Configuring Amendment Methods"](#)) to customize the root or branch class descriptor's `InheritancePolicy`.

[Example 119–9](#) shows an amendment method for the `Person` class. In this example, you use the `InheritancePolicy` method `dontReadSubclassesOnQueries` to configure the descriptor so that subclasses are not read on queries.

**Example 119–9 Configuring Reading Subclasses on Queries**

```
...
public static void addToPersonDescriptor(Descriptor descriptor) {
    descriptor.getInheritancePolicy().dontReadSubclassesOnQueries();
}
...
```

[Example 119–10](#) shows an amendment method for the `Person` class. In this example, you use the `InheritancePolicy` method `setReadAllSubclassesViewName` to optimize multiple table inheritance queries.

**Example 119–10 Configuring Reading Subclasses on Queries Using a View Name**

```
...
public static void addToPersonDescriptor(Descriptor descriptor) {
    descriptor.getInheritancePolicy().setReadAllSubclassesViewName(myView);
}
...
```

[Example 119–11](#) shows an amendment method for the `Person` class. In this example, you use the `InheritancePolicy` method `setShouldOuterJoinSubclasses` to configure the descriptor so that subclasses are outer-joined on queries.

**Example 119–11 Configuring Outer-Joining Subclasses on Queries**

```
...
public static void addToPersonDescriptor(Descriptor descriptor) {
    descriptor.getInheritancePolicy().setShouldOuterJoinSubclasses(true);
}
...
```

## 119.20 Configuring Inheritance for a Child (Branch or Leaf) Class Descriptor

Inheritance describes how a derived (child) class inherits the characteristics of its superclass (parent). When you designate a class as a child, you must also specify the descriptor that represents the child's parent in your inheritance hierarchy.

[Table 119–39](#) summarizes which descriptors support child inheritance configuration.

**Table 119–21 Descriptor Support for Child Inheritance Configuration**

Descriptor	How to Use Oracle JDeveloper	How to Configure Inheritance for a Child (Branch or Leaf) Class Descriptor Using TopLink Workbench	How to Configure Inheritance for a Child (Branch or Leaf) Class Descriptor Using Java
Relational Descriptors	✓	✓	✓
Object-Relational Data Type Descriptors			✓
EIS Descriptors	✓	✓	✓
XML Descriptors	✓	✓	✓

For more information about inheritance, see [Section 16.2.2, "Descriptors and Inheritance"](#).

For more information about configuring inheritance for a parent (root) class descriptor, see [Section 119.21, "Configuring Inheritance for a Parent \(Root\) Descriptor"](#).

### 119.20.1 How to Configure Inheritance for a Child (Branch or Leaf) Class Descriptor Using TopLink Workbench

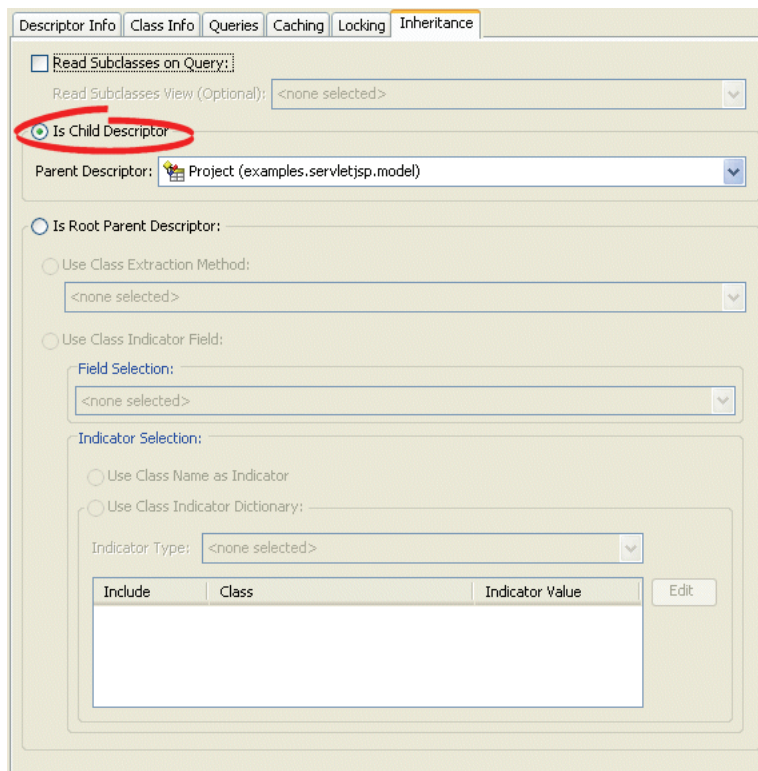
To create a child (branch or leaf class) for an inheritance, use this procedure.

1. In the **Navigator**, select the descriptor you wish to specify as a child.
2. Choose the **Inheritance** tab in the **Property** window.

If the **Inheritance** tab is not visible, right-click the descriptor and choose **Select Advanced Properties > Inheritance**.

3. Select the **Is Child Descriptor** option to specify this descriptor is a child class. The **Parent Descriptor** list is now enabled and the class indicator information is disabled.

**Figure 119–32 Inheritance Tab, Child Descriptor Option**



Use the following information to enter data in each child descriptor field on the tab:

Field	Description
<b>Is Child Descriptor</b>	Specify that this descriptor is a child class to be used in a branch or leaf.
<b>Parent Descriptor</b>	Use the list to select the parent of this descriptor. See <a href="#">Section 16.2.2, "Descriptors and Inheritance"</a> for more information.

## 119.20.2 How to Configure Inheritance for a Child (Branch or Leaf) Class Descriptor Using Java

Using Java, you can configure an inheritance child descriptor using `InheritancePolicy` method `setParentClass`, as [Example 119–12](#) shows.

### **Example 119–12** Configuring an Inheritance Child Descriptor

```
descriptor.getInheritancePolicy().setParentClass(ChildClass.class);
```

## 119.21 Configuring Inheritance for a Parent (Root) Descriptor

Inheritance describes how a derived (child) class inherits the characteristics of its superclass (parent). When you designate a class as a parent, you can configure how `TopLink` handles the class's inheritance hierarchy.

[Table 119–24](#) summarizes which descriptors support parent inheritance configuration.

**Table 119–22** Descriptor Support for Parent Inheritance Configuration

Descriptor	How to Use Oracle JDeveloper	How to Configure Inheritance for a Parent (Root) Descriptor Using TopLink Workbench	How to Configure Inheritance for a Parent (Root) Descriptor Using Java
Relational Descriptors	✓	✓	✓
Object-Relational Data Type Descriptors			✓
EIS Descriptors	✓	✓	✓
XML Descriptors	✓	✓	✓

For more information about configuring inheritance for a child (branch or leaf) class descriptor, see [Section 119.20](#), "Configuring Inheritance for a Child (Branch or Leaf) Class Descriptor".

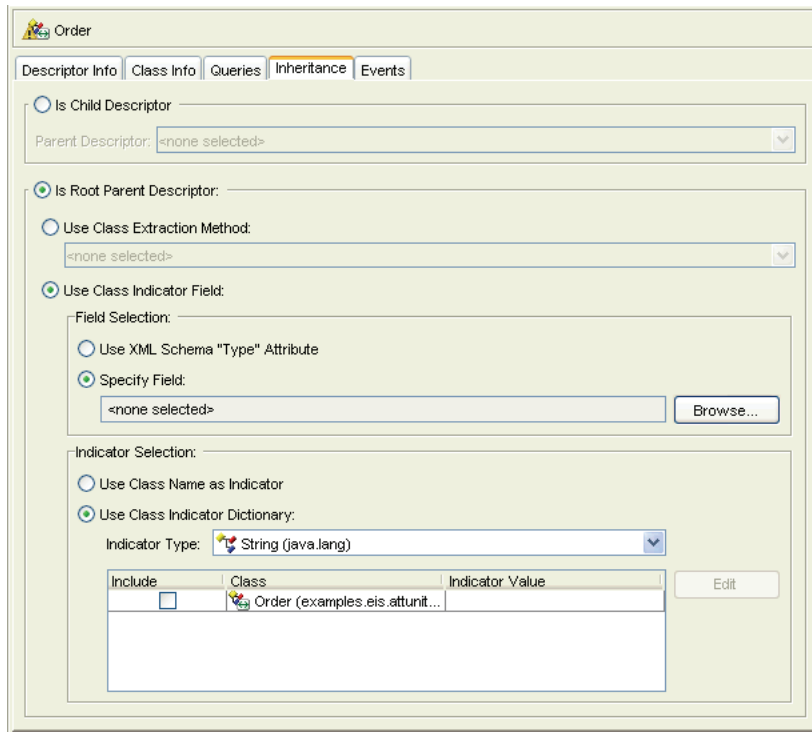
For more information, see [Section 16.2.2](#), "Descriptors and Inheritance".

### 119.21.1 How to Configure Inheritance for a Parent (Root) Descriptor Using TopLink Workbench

To create a root class for an inheritance, use this procedure.

1. In the **Navigator**, select the descriptor you wish to specify as the root.
2. Choose the **Inheritance** tab in the **Property** window.  
If the **Inheritance** tab is not visible, right-click the descriptor and choose **Select Advanced Properties > Inheritance**.
3. Select the **Is Root Parent Descriptor** option to specify this descriptor is a root class.

**Figure 119–33 Inheritance Tab, Configuring Inheritance for a Root Descriptor**



Use this table to complete the following root descriptor field on the Inheritance tab:

Field	Description
<b>Is Root Parent Descriptor</b>	Select this option to specify this descriptor as the root (parent) of the inheritance hierarchy.
<b>Use Class Extraction Method</b>	Choose this option to specify a class indicator using a class extraction method, and select your static class extraction method from the list.  For more information, see <a href="#">Section 16.3.1.2, "Using Class Extraction Methods"</a> .
<b>Use Class Indicator Field</b>	Choose this option to specify a class indicator using a class indicator field.  For more information, see <a href="#">Section 16.3.1.1, "Using Class Indicator Fields"</a> .
<b>Field Selection</b>	Choose the field to use as the class indicator field.
<b>Use XML Schema "Type" Attribute<sup>1</sup></b>	Select this option to use the type attribute specified in the XML schema for this descriptor's reference class.
<b>Specify Field</b>	For a relational descriptor, select the field of the database table associated with this descriptor (see <a href="#">Section 23.2, "Configuring Associated Tables"</a> ).  For an EIS root descriptor (using XML records) or an XML descriptor, click <b>Browse</b> to select an element attribute or text node.



Field	Description
<b>Indicator Selection</b>	Choose between using a class name as the class indicator field value or specifying specific class indicator field values for each (nonabstract) child class.
<b>Use Class Name as Indicator</b>	Choose this option to use class names as the class indicator field value.
<b>Use Class Indicator Dictionary</b>	Choose this option to specify specific class indicator field values for each (nonabstract) child class.  When you choose this option, you must specify the data type of the class indicator field and the specific class indicator field values for each (nonabstract) child class.
<b>Indicator Type</b>	Select the data type from the list to specify the data type of the class indicator field.  To specify the specific class indicator field values for each (nonabstract) child class, click <b>Edit</b> and enter the appropriate value for each child class.

<sup>1</sup> EIS root (see [Section 75.2.1.1, "EIS Root Descriptors"](#)) or XML descriptors (see [Section 50.1, "XML Descriptor Concepts"](#)) only.

## 119.21.2 How to Configure Inheritance for a Parent (Root) Descriptor Using Java

Create a descriptor amendment method ([Section 119.35, "Configuring Amendment Methods"](#)) to customize the root class descriptor's inheritance policy using `InheritancePolicy` methods `setParentClass`, `setClassIndicatorFieldName`, `addClassIndicator`, `useClassNameAsIndicator` and `setClassExtractionMethodName`, as required.

[Example 119–15](#) shows amendment methods for the `Person` and `Student` classes where `Student` extends `Person` in a relational project. In this example, a class indicator field is used (see [Section 16.3.1.1, "Using Class Indicator Fields"](#)).

### **Example 119–13 Configuring Inheritance for a Relational Root Class**

```
...
public static void addToPersonDescriptor(Descriptor descriptor) {
    descriptor.getInheritancePolicy().setClassIndicatorFieldName("CLIENT_TYPE");
    descriptor.getInheritancePolicy().addClassIndicator(Student.class, indicator);
}

public static void addToStudentDescriptor(Descriptor descriptor) {
    descriptor.getInheritancePolicy().setParentClass(Person.class);
    ...
}
...
```

If you are using a class-extraction method (see [Section 16.3.1.2, "Using Class Extraction Methods"](#)), you may also need to use `InheritancePolicy` methods `setOnlyInstancesExpression` and `setWithAllSubclassesExpression` (see [Section 119.22, "Configuring Inheritance Expressions for a Parent \(Root\) Class Descriptor"](#)).

[Example 119–15](#) shows amendment methods for the `Person` and `Student` classes where `Student` extends `Person` in an EIS project using XML records. In this example, a class indicator field is used (see [Section 16.3.1.1, "Using Class Indicator Fields"](#)).

**Example 119–14 Configuring Inheritance for an EIS Root Class**

```

...
public static void addToPersonDescriptor(Descriptor descriptor) {
    descriptor.getInheritancePolicy().setClassIndicatorField(
        new XMLField("@CLIENT_TYPE"));
    descriptor.getInheritancePolicy().addClassIndicator(Student.class, indicator);
}

public static void addToStudentDescriptor(Descriptor descriptor) {
    descriptor.getInheritancePolicy().setParentClass(Person.class);
    descriptor.getInheritancePolicy().setClassIndicatorField(
        new XMLField("@CLIENT_TYPE")
    );
}
...

```

## 119.22 Configuring Inheritance Expressions for a Parent (Root) Class Descriptor

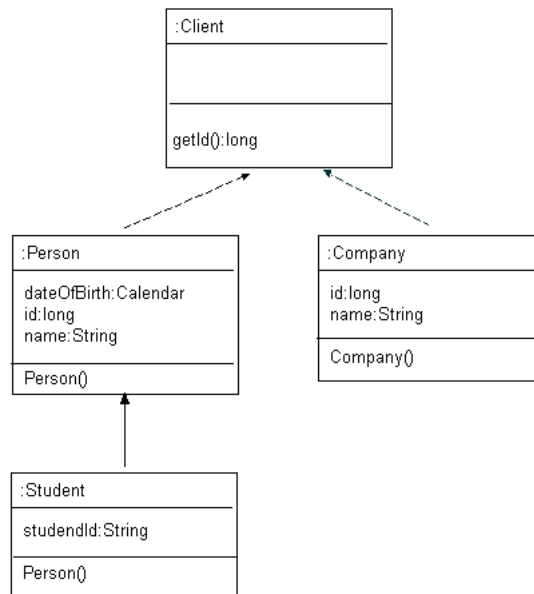
If your class uses inheritance (see [Section 16.3, "Descriptors and Inheritance"](#)) with a class extraction method (see [Section 16.3.1.2, "Using Class Extraction Methods"](#)) you must provide `TopLink` with expressions to correctly filter sibling instances for all classes that share a common table.

[Table 119–24](#) summarizes which descriptors support inheritance expression configuration.

**Table 119–23 Descriptor Support for Inheritance Expression Configuration**

Descriptor	How to Use Oracle JDeveloper	How to Use TopLink Workbench	How to Configure Inheritance Expressions for a Parent (Root) Class Descriptor Using Java
Relational Descriptors			✓
Object-Relational Data Type Descriptors			✓
EIS Descriptors			
XML Descriptors			

[Figure 119–34](#) shows a typical inheritance hierarchy. In this example, instances of both `Person` and `Student` are stored in the same `PERSON` table, as [Figure 119–35](#) shows: an instance of `Person` has a null value for `STUDENT_NUMBER`. Instances of `Company` are stored in a separate `COMPANY` table.

**Figure 119–34 Example Inheritance Hierarchy****Figure 119–35 PERSON Table**

DOB	ID	NAME	STUDENT_NUMBER
1964-08-24	123	Joan Smith	
1983-02-01	456	Jane Doe	4821

Queries on inheritance classes that share a common table, such as `Person` and `Student`, must filter out their sibling instances. `TopLink` performs this filtering using the `Expression` instances returned by the descriptor's `InheritancePolicy` methods `getOnlyInstancesExpression` and `getWithAllSubclassesExpression`.

Queries on a class that has its own table for its specific data, such as `Company`, and does not share this table with any sibling classes, do not require these expressions.

If you use a class indicator type field (see [Section 16.3.1.1, "Using Class Indicator Fields"](#)), `TopLink` automatically generates the required expressions.

If you use a class extraction method (see [Section 16.3.1.2, "Using Class Extraction Methods"](#)), you must provide `TopLink` with an expressions to correctly filter sibling instances for all classes that share a common table.

For concrete classes, you must define an `only-` instances expression.

For branch classes, you must define a `with-all-subclasses` expression.

When `TopLink` queries for a leaf class, it uses the `only-` instances expression to filter out any sibling classes.

When `TopLink` queries for a root or branch class whose subclasses do not define their own tables, it uses the `with-all-subclasses` expression. This is also the case when a subclass view is used (see [Section 119.19, "Configuring Reading Subclasses on Queries"](#)).

When querying for a root or branch class that has subclasses that span multiple tables, a query is performed for each concrete class in the inheritance hierarchy using the `only- instances` expression to filter sibling classes.

When a class extraction method is used the `only-instances` expression is used to determine if a class is concrete. If a class does not require an `only instances` expression, do not enable reading subclasses on queries (see [Section 119.19, "Configuring Reading Subclasses on Queries"](#)), otherwise `TopLink` will assume that the class has no instances and it will skip that class on queries.

For more information about inheritance expressions, see [Section 16.3.1.2.1, "Specifying Expressions for Only-Instances and With-All-Subclasses"](#).

## 119.22.1 How to Configure Inheritance Expressions for a Parent (Root) Class Descriptor Using Java

Create a descriptor amendment method ([Section 119.35, "Configuring Amendment Methods"](#)) to customize the root class descriptor's `InheritancePolicy` using `InheritancePolicy` methods `setOnlyInstancesExpression` and `setWithAllSubclassesExpression`, as required.

[Example 119–15](#) shows amendment methods for the `Person` and `Student` descriptors based on the class hierarchy shown in [Figure 119–34](#) and the database table shown in [Figure 119–35](#).

### **Example 119–15 Configuring Only-Instances Expressions**

```
...
// Only-instances expression for Person
public static void addToPersonDescriptor(Descriptor descriptor) {
    ExpressionBuilder builder = new ExpressionBuilder();
    descriptor.getInheritancePolicy().setOnlyInstancesExpression(
        builder.getField("STUDENT_NUMBER").isNull());
}

// Only-instances expression for Student
public static void addToStudentDescriptor(Descriptor descriptor) {
    ExpressionBuilder builder = new ExpressionBuilder();
    descriptor.getInheritancePolicy().setOnlyInstancesExpression(
        builder.getField("STUDENT_NUMBER").notNull());
}
...
```

[Example 119–16](#) shows amendment methods for the `Bicycle` and `NonFueledVehicle` descriptors based on the class hierarchy shown in [Figure 16–1](#) if the vehicle hierarchy stored all of the classes in a single vehicle table, and there was not a class indicator, but a class extraction method instead.

### **Example 119–16 Configuring Only-Instances and With-All-Subclasses Expressions**

```
// Bicycle ammendment
public static void addToBicycleDescriptor(Descriptor descriptor) {
    ExpressionBuilder builder = new ExpressionBuilder();
    descriptor.getInheritancePolicy().setOnlyInstancesExpression(
        builder.getField("BICYCLE_DESCR").notNull());
}

// NonFueledVehicle ammendment
public static void addToNonFueledVehicleDescriptor(Descriptor descriptor) {
    ExpressionBuilder builder = new ExpressionBuilder();
```

```

descriptor.getInheritancePolicy().setWithAllSubclassesExpression(
    builder.getField("FUEL_TYPE").isNull());
}

```

## 119.23 Configuring Inherited Attribute Mapping in a Subclass

If you are defining the descriptor for a class that inherits attributes from another class, then you can create mappings for those attributes. If you remap an attribute that was already mapped in the superclass, then the new mapping applies to the subclass only. Any other siblings that inherit the attribute are unaffected.

If you leave inherited attributes unmapped, TopLink uses the mapping (if any) from the superclass if the superclass's descriptor has been designated as the parent descriptor.

Table 119–24 summarizes which descriptors support inherited attribute mapping configuration.

**Table 119–24 Descriptor Support for Inherited Attribute Mapping Configuration**

Descriptor	How to Use Oracle JDeveloper	How to Configure Inherited Attribute Mapping in a Subclass Using TopLink Workbench	How to Configure Inherited Attribute Mapping in a Subclass Using Java
Relational Descriptors	✓	✓	✓
Object-Relational Data Type Descriptors			✓
EIS Descriptors	✓	✓	✓
XML Descriptors	✓	✓	✓

For more information, see [Section 16.2.2, "Descriptors and Inheritance"](#).

### 119.23.1 How to Configure Inherited Attribute Mapping in a Subclass Using TopLink Workbench

To map inherited attributes, use this procedure:

1. In the **Navigator**, right-click a descriptor and choose **Map Inherited Attributes > selected class** from the context menu or choose **Selected > Map Inherited Attributes** from the menu.

The mappings list now includes all the attributes from the superclass of this class.

2. Map any desired attributes. See [Chapter 120, "Creating a Mapping"](#) for more information.

### 119.23.2 How to Configure Inherited Attribute Mapping in a Subclass Using Java

Using Java, attributes inherited by a subclass from a superclass will be visible and you can always create a mapping to these inherited attributes.

## 119.24 Configuring a Domain Object Method as an Event Handler

You can associate a domain object method with any of the descriptor events shown in [Table 119–26](#). You can register any domain object method that complies with the following criteria:

- Is public.
- Returns void.
- Takes a single parameter of type `DescriptorEvent`

[Table 119–25](#) summarizes which descriptors support domain object method event handler configuration.

**Table 119–25** *Descriptor Support for Domain Object Method Event Handler Configuration*

Descriptor	How to Use Oracle JDeveloper	How to Configure a Domain Object Method as an Event Handler Using TopLink Workbench	How to Configure a Domain Object Method as an Event Handler Using Java
Relational Descriptors	✓	✓	✓
Object-Relational Data Type Descriptors			✓
EIS Descriptors	✓	✓	✓
XML Descriptors			

For example, you can add a method `handlePostDelete` (that is public, returns void, and takes a single parameter of type `DescriptorEvent`) to your `Employee` object to handle `PostDeleteEvent` descriptor events. After you register that method with the `DescriptorEventManager` owned by the `Employee` object's descriptor as the handler for `PostDeleteEvent` descriptor events, whenever the Oracle TopLink runtime performs a post-delete operation on an instance of the `Employee` object, the runtime dispatches a `PostDeleteEvent` to the `handlePostDelete` method on the instance of the `Employee` object associated with that `PostDeleteEvent`.

The **Descriptor Event ID** column in [Table 119–26](#) lists the `DescriptorEventManager` field name used to identify a particular event. The `DescriptorEvent` method `getEventCode` returns this value. For example:

```
if (descriptorEvent.getEventCode() == DescriptorEventManager.PreUpdateEvent) {
    // descriptorEvent represents a pre-update event
}
```

**Table 119–26** *Descriptor Events*

Category	Descriptor Event ID	Description
Delete	<code>PreDeleteEvent</code>	Occurs before an object is deleted from the data source.
	<code>AboutToDeleteEvent</code>	Occurs when an object is deleted from the data source.
	<code>PostDeleteEvent</code>	Occurs after an object is deleted from the data source.
Insert	<code>PreInsertEvent</code>	Occurs before an object is inserted in the data source.
	<code>AboutToInsertEvent</code>	Occurs when an object is inserted in the data source.

**Table 119–26 (Cont.) Descriptor Events**

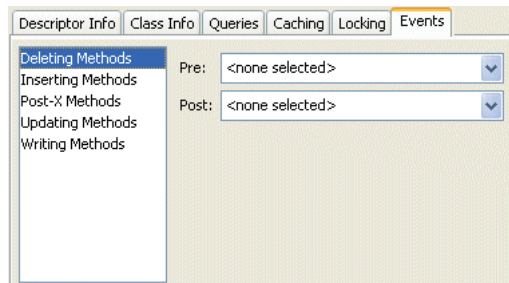
Category	Descriptor Event ID	Description
	PostInsertEvent	Occurs after an object is inserted into the data source.
Post-X	PostBuildEvent	Occurs after an object is built from the data source.
	PostCloneEvent	Occurs after an object has been cloned into a unit of work.
	PostMergeEvent	Occurs after an object has been merged from a unit of work.
	PostRefreshEvent	Occurs after an object is refreshed from the data source.
Update	PreUpdateEvent	Occurs before an object is updated in the data source. This may be called in a unit of work even if the object has no changes and does not require updating.
	AboutToUpdateEvent	Occurs when an object is updated in the data source. This method is called only if the object has changes in the unit of work.
	PostUpdateEvent	Occurs after an object is updated in the data source.
Write	PreWriteEvent	Occurs before an object is inserted or updated in the data source. This occurs before PreInsertEvent and PreUpdateEvent.
	PostWriteEvent	Occurs after an object is inserted or updated in the data source. This occurs after PostInsertEvent and PostUpdateEvent.

Alternatively, you can configure a descriptor event listener as an event handler (see [Section 119.25, "Configuring a Descriptor Event Listener as an Event Handler"](#)).

### 119.24.1 How to Configure a Domain Object Method as an Event Handler Using TopLink Workbench

To select event methods, use this procedure:

1. Select a descriptor in the **Navigator**. Its properties appear in the Editor.  
If the **Events** advanced property is not visible for the descriptor, then right-click the descriptor and choose **Select Advanced Properties > Events** from context menu or from the **Selected** menu.
2. Click the **Event** tab in the **Editor**.

**Figure 119–36 Events Tab**

3. Select the appropriate method category from the list on the left.

Use this table to enter data in the following fields to select the appropriate domain object method:

Category	Option	Description
Deleting Methods	Pre	Select the domain object method that is invoked on an instance of its reference class <i>before</i> the instance is deleted from the data source.
	Post	Select the domain object method that is invoked on an instance of its reference class <i>after</i> the instance is deleted from the data source.
Inserting Methods	Pre	Select the domain object method that is invoked on an instance of its reference class <i>before</i> the instance is inserted in the data source.
	About To	Select the domain object method that is invoked on an instance of its reference class when the instance is inserted in the data source.
	Post	Select the domain object method that is invoked on an instance of its reference class <i>after</i> the instance is inserted into the data source.
Post-X Methods	Build	Select the domain object method that is invoked on an instance of its reference class <i>after</i> the instance is <i>built</i> from the data source.
	Clone	Select the domain object method that is invoked on an instance of its reference class <i>after</i> the instance is <i>cloned</i> into a unit of work.
	Merge	Select the domain object method that is invoked on an instance of its reference class <i>after</i> the instance is <i>merged</i> from a unit of work.
	Refresh	Select the domain object method that is invoked on an instance of its reference class <i>after</i> the instance is <i>refreshed</i> from the data source.
Updating Methods	Pre	Select the domain object method that is invoked on an instance of its reference class <i>before</i> the instance is updated in the data source. This may be called in a unit of work even if the object has no changes and does not require updating.
	About to	Select the domain object method that is invoked on an instance of its reference class when the instance is updated in the data source. This method is called <i>only</i> if the object has changes in the unit of work.
	Post	Select the domain object method that is invoked on an instance of its reference class <i>after</i> the instance is updated in the data source.
Writing Methods	Pre	Select the domain object method that is invoked on an instance of its reference class <i>before</i> the instance is inserted or updated in the data source. <b>Note:</b> This occurs before Pre-Insert and Pre-Update event methods are invoked.



Category	Option	Description
	Post	Select the domain object method that is invoked on an instance of its reference class <i>after</i> the instance is inserted or updated in the data source.  <b>Note:</b> This occurs after Post-Insert or Post-Update event methods are invoked.

## 119.24.2 How to Configure a Domain Object Method as an Event Handler Using Java

[Example 119–17](#) shows a domain object class with method `handlePostDelete` defined to handle `PostDeleteEvent` descriptor events. [Example 119–18](#) shows how to register this method as the `PostDeleteEvent` event handler. Whenever the TopLink runtime performs a post-delete operation on an instance of `Employee`, the runtime will dispatch a `PostDeleteEvent` to the `DescriptorEventManager` owned by the `Employee` object's descriptor. The `DescriptorEventManager` will then invoke the `handlePostDelete` method on the instance of `Employee` associated with that `PostDeleteEvent`.

### **Example 119–17 Domain Object Method as a Descriptor Event Handler**

```
public class Employee {
    // domain object methods
    ...
    public void handlePostDelete(DescriptorEvent event) {
        // handler implementation
    }
}
```

### **Example 119–18 Registering a Domain Object Method as a Descriptor Event Handler**

```
employeeDescriptor.getEventManager().setPostDeleteSelector("handlePostDelete");
```

## 119.25 Configuring a Descriptor Event Listener as an Event Handler

You can create your own `DescriptorEventListener` and register it with a `DescriptorEventManager` in a descriptor amendment method. You can also configure a `DescriptorEventListener` to be notified of events through the Java event model.

You can register any object that implements the `DescriptorEventListener` interface with the `DescriptorEventManager` owned by a domain object's descriptor to handle any descriptor event type (see [Table 119–28](#)). To quickly implement this interface, you can extend abstract class `DescriptorEventAdapter` and override only the methods for the events you are interested in.

[Table 119–27](#) summarizes which descriptors support descriptor event listener configuration.

**Table 119–27 Descriptor Support for Descriptor Event Listener Configuration**

Descriptor	How to Use Oracle JDeveloper	How to Configure a Descriptor Event Listener as an Event Handler Using TopLink Workbench	How to Configure a Descriptor Event Listener as an Event Handler Using Java
Relational Descriptors	✓	✓	✓

**Table 119–27 (Cont.) Descriptor Support for Descriptor Event Listener Configuration**

Descriptor	How to Use Oracle JDeveloper	How to Configure a Descriptor Event Listener as an Event Handler Using TopLink Workbench	How to Configure a Descriptor Event Listener as an Event Handler Using Java
Object-Relational Data Type Descriptors			✓
EIS Descriptors	✓	✓	✓
XML Descriptors			

For example, you create a `DescriptorEventListener` to handle `PostBuildEvent` descriptor events for `Employee` objects. After you register this `DescriptorEventListener` with the `DescriptorEventManager` owned by the `Employee` object's descriptor, whenever the TopLink runtime performs a post-build operation on an instance of `Employee`, the runtime dispatches a `PostBuildEvent` to the event listener's `postBuild` method.

Table 119–28 lists the `DescriptorEventListener` methods associated with each descriptor event. The **Descriptor Event Listener Method** column lists the `DescriptorEventListener` methods associated with each `DescriptorEvent`.

**Table 119–28 Descriptor Events**

Category	Descriptor Event Listener Method	Description
Delete	<code>preDelete</code>	Occurs before an object is deleted from the data source.
	<code>aboutToDelete</code>	Occurs when an object is deleted from the data source.
	<code>postDelete</code>	Occurs after an object is deleted from the data source.
Insert	<code>preInsert</code>	Occurs before an object is inserted in the data source.
	<code>aboutToInsert</code>	Occurs when an object is inserted in the data source.
	<code>postInsert</code>	Occurs after an object is inserted into the data source.
Post-X	<code>postBuild</code>	Occurs after an object is built from the data source.
	<code>postClone</code>	Occurs after an object has been cloned into a unit of work.
	<code>postMerge</code>	Occurs after an object has been merged from a unit of work.
	<code>postRefresh</code>	Occurs after an object is refreshed from the data source.
Update	<code>preUpdate</code>	Occurs before an object is updated in the data source. This may be called in a unit of work even if the object has no changes and does not require updating.
	<code>aboutToUpdate</code>	Occurs when an object is updated in the data source. This method is called only if the object has changes in the unit of work.
	<code>postUpdate</code>	Occurs after an object is updated in the data source.
Write	<code>preWrite</code>	Occurs before an object is inserted or updated in the data source. This occurs before <code>PreInsertEvent</code> and <code>PreUpdateEvent</code> .
	<code>postWrite</code>	Occurs after an object is inserted or updated in the data source. This occurs after <code>PostInsertEvent</code> and <code>PostUpdateEvent</code> .

Alternatively, you can configure a domain object method as an event handler (see [Section 119.24, "Configuring a Domain Object Method as an Event Handler"](#)).

### 119.25.1 How to Configure a Descriptor Event Listener as an Event Handler Using TopLink Workbench

For more information, see [Section 119.24.1, "How to Configure a Domain Object Method as an Event Handler Using TopLink Workbench"](#).

### 119.25.2 How to Configure a Descriptor Event Listener as an Event Handler Using Java

[Example 119–19](#) shows a `DescriptorEventListener` that handles `PostBuildEvent` descriptor events. [Example 119–20](#) shows how to register this `DescriptorEventListener` with the `Employee` object's descriptor. Whenever the TopLink runtime performs a post-build operation on an instance of `Employee`, the runtime will dispatch a post build event to the corresponding `DescriptorEventListener` method on each registered event listener (in this case, it calls the `postBuild` method).

#### **Example 119–19** *DescriptorEventListener*

```
public class MyDescriptorEventListener extends DescriptorEventAdapter {

    public void postBuild(DescriptorEvent event) {
        // handler implementation
    }
}
```

#### **Example 119–20** *Registering a DescriptorEventListener with the DescriptorEventManager*

```
descriptor.getEventManager().addListener(new MyDescriptorEventListener());
```

## 119.26 Configuring Locking Policy

You can configure a descriptor with a locking policy that prevents one user writing over another user's work.

[Table 119–29](#) summarizes which descriptors support locking policies.

**Table 119–29** *Descriptor Support for Locking Policy*

Descriptor	Optimistic Version Locking Policies	Optimistic Field Locking Policies	Pessimistic Locking Policy	How to Use Oracle JDeveloper	How to Configure Locking Policy Using TopLink Workbench	How to Configure Locking Policy Using Java
Relational Descriptors <sup>1</sup>	✓	✓	✓	✓	✓	✓
Object-Relational Data Type Descriptors	✓	✓	✓			✓
EIS Descriptors <sup>2</sup>	✓		✓	✓	✓	✓
XML Descriptors						

<sup>1</sup> Relational class descriptors only (see [Section 22.2.1.1, "Creating Relational Class Descriptors"](#)).

<sup>2</sup> EIS root descriptors only (see [Section 75.2.1.1, "EIS Root Descriptors"](#)).

Oracle recommends that you use a locking policy. You should use a locking policy in any multiuser environment to prevent one user writing over another user's changes. Although locking can be particularly important if multiple servers or multiple applications access the same data, even in a single server application, the same locking issue still exists. In a multiple-server environment, locking is still relevant even if your application uses cache refreshing or cache coordination.

If you are building a three-tier application, in order to correctly lock an object, you must obtain the lock before the object is sent to client to be edited. The type of locking you choose has an influence on how you can achieve this (see [Section 16.4.6, "Locking in a Three-Tier Application"](#)).

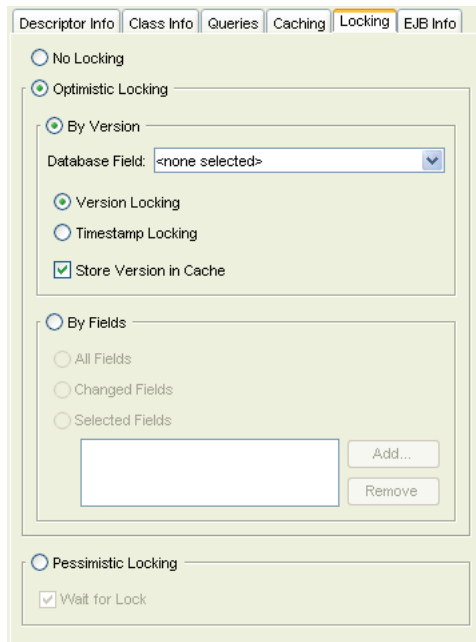
### 119.26.1 How to Configure Locking Policy Using TopLink Workbench

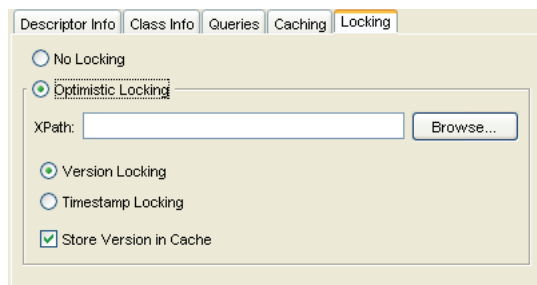
To specify a descriptor's locking policy, use this procedure:

1. In the **Navigator**, select a relational or EIS root descriptor.
 

If the **Locking** advanced property is not visible for the descriptor, right-click the descriptor and choose **Select Advanced Properties > Locking** from the context menu or from the **Selected** menu.
2. Click the **Locking** tab.

**Figure 119–37 Locking Tab for a Descriptor**



**Figure 119–38 Locking Tab for an EIS Root Descriptor**

Use this table to enter data in the following fields on the tab of the appropriate type:

Field	Description
<b>Optimistic Locking</b>	Specify that the descriptor uses optimistic locking.
<b>By Version</b>	Specify to use optimistic locking, based on versioning.
<b>Database Field</b>	Select the database field that contains the version value used for optimistic locking. This field appears for relational descriptors only.
<b>XPath</b>	Click <b>Browse</b> to define the path to the element or attribute that stores the version value. This field appears for EIS root descriptors only. Ensure that the attribute's type corresponds to the type of locking policy you choose ( <code>numeric</code> for <b>Version Locking</b> and <code>timestamp</code> for <b>Timestamp Locking</b> ).
<b>Version Locking</b>	Specify that the descriptor uses numeric version locking. The version field (defined by the <b>Database Field</b> , for relational descriptors, or the <b>XPath</b> , for EIS root descriptors) must be a <code>numeric</code> type
<b>Timestamp Locking</b>	Specify that the descriptor uses time stamp version locking, based on time stamp. The version field (defined by the <b>Database Field</b> , for relational descriptors, or the <b>XPath</b> , for EIS root descriptors) must be a <code>timestamp</code> type.
<b>Store Version in Cache</b>	Specify whether or not you want to store the version information in the cache.  If you choose not to define a mapping for the version field, then you must enable this option to configure the descriptor to store the version value in the Oracle TopLink cache.  If you choose to define a mapping for the version field, then you must disable this option in order to store the version value in the object.  For more information, see <a href="#">Section 16.4.6.1, "Optimistic Locking in a Three-Tier Application"</a> .
<b>By Fields<sup>1</sup></b>	Specify to use optimistic locking, based on database fields. These fields appear for relational descriptors only.
<b>All Fields</b>	Select <i>all</i> fields for optimistic locking.
<b>Changed Fields</b>	Select <i>only the changed</i> fields for optimistic locking.

Field	Description
<b>Selected Fields</b>	Click <b>Add</b> to select <i>specific</i> database fields for optimistic locking.
<b>Pessimistic Locking</b>	Specify to use pessimistic locking for this descriptor. This applies only to descriptors that have had EJB CMP information configured for them (see <a href="#">Section 119.18</a> , "Configuring a Descriptor with EJB CMP and BMP Information").
<b>Wait for Lock</b>	Specify whether or not TopLink should wait for a data source lock. When not selected, the thread of execution will immediately throw a <code>DatabaseException</code> if it cannot acquire a read lock on the object.  When selected, the thread of execution will wait indefinitely until the read lock is released, at which time, it will attempt to acquire it. Use this option with care as it can lead to application deadlocks.

<sup>1</sup> You cannot use field locking with the `AttributeChangeTrackingPolicy` (see [Section 113.2.3.3](#), "Attribute Change Tracking Policy").

## 119.26.2 How to Configure Locking Policy Using Java

This section describes the following:

- [Configuring an Optimistic Locking Policy](#)
- [Configuring Optimistic Locking Policy Cascading](#)
- [Configuring a Pessimistic Locking Policy](#)

### 119.26.2.1 Configuring an Optimistic Locking Policy

Use the `ClassDescriptor` method `setOptimisticLockingPolicy` to set an instance of the appropriate optimistic field locking policy:

- `AllFieldsLockingPolicy`
- `ChangedFieldsLockingPolicy`
- `SelectedFieldsLockingPolicy`
- `VersionLockingPolicy`
- `TimestampLockingPolicy`

Use the `ClassDescriptor` method `getOptimisticLockingPolicy` to get the selected locking policy type and configure it.

### 119.26.2.2 Configuring Optimistic Locking Policy Cascading

If you are using a `VersionLockingPolicy`, you can enable cascading to configure TopLink to automatically force a version field update on a parent object when its privately owned child object's version field changes. Use `VersionLockingPolicy` method `setIsCascaded` passing in a boolean of `true` to enable cascading, or `false` to disable cascading.

For more information, see [Section 16.4.2](#), "Optimistic Version Locking Policies and Cascading".

### 119.26.2.3 Configuring a Pessimistic Locking Policy

You can configure a descriptor with a `PessimisticLockingPolicy` only when using a `CMPPolicy`. That is, you only can configure a `PessimisticLockingPolicy` for descriptors that support EJB CMP information (see [Section 119.18](#), "Configuring a Descriptor with EJB CMP and BMP Information") in a CMP project.

Instantiate a `CMPPolicy` and use `CMPPolicy` method `setPessimisticLockingPolicy` to set an instance of `PessimisticLockingPolicy`. Then use the `ClassDescriptor` method `setCMPPolicy` to set the `CMPPolicy`.

## 119.27 Configuring Returning Policy

Using a `ReturningPolicy`, you can obtain field values from the data source when inserting or updating an object. `TopLink` uses the values that the data source returns to update the object attributes that map to these fields. You can specify which fields to return for inserts and updates. For insert fields, you can also specify whether or not to include the field value in the insert operation.

A `ReturningPolicy` is useful when the data source provides default or initial field values through defaults, triggers, or stored procedures. You can also use a `ReturningPolicy` to allow the data source to assign a sequence or primary key value.

Any object attribute that you do not configure in a descriptor's `ReturningPolicy` receives the default behavior: in the context of a unit of work, if the attribute has changed, its value is written to the database. If the SQL statement invokes a trigger or stored procedure that modifies the database field, the database generated value is not reflected by the object.

Use caution when deciding on whether or not to use a `ReturningPolicy`, as doing so may effect insert or update performance and is not compatible with batch writing (see [Section 12.11.3](#), "How to Use Batch Writing for Optimization").

By default, you can use a `ReturningPolicy` with Oracle Database, in which case, `TopLink` uses the Oracle `RETURNING` clause (see [Section 119.27.1](#), "How to Configure Returning Policy Using TopLink Workbench").

You can use a `ReturningPolicy` with a non-Oracle database if you configure your descriptor's insert or update query to use a stored procedure that returns the desired returned values as output parameters (see [Section 119.27.2](#), "How to Configure Returning Policy Using Java").

[Table 119–39](#) summarizes which descriptors support returning policy configuration.

**Table 119–30 Descriptor Support for Fetch Group Configuration**

Descriptor	How to Use Oracle JDeveloper	How to Configure Returning Policy Using TopLink Workbench	How to Configure Returning Policy Using Java
Relational Descriptors	✓	✓	✓
Object-Relational Data Type Descriptors			✓
EIS Descriptors <sup>1</sup>	✓	✓	✓
XML Descriptors			

<sup>1</sup> EIS root descriptors only (see [Section 75.2.1.1, "EIS Root Descriptors"](#)).

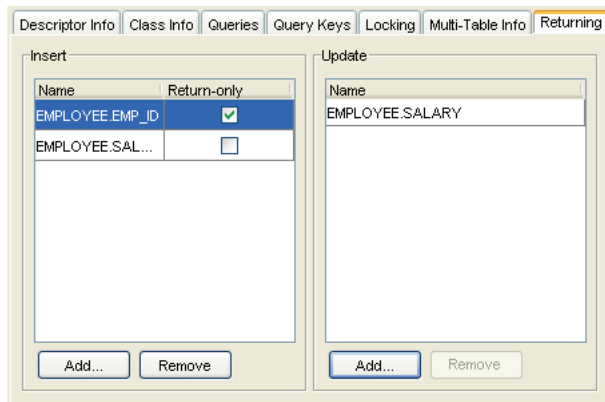
### 119.27.1 How to Configure Returning Policy Using TopLink Workbench

To specify the return policy for a descriptor, use this procedure:

1. Select a descriptor in the **Navigator**. Its properties appear in the Editor.
 

If the **Returning** advanced property is not visible for the descriptor, right-click the descriptor and choose **Select Advanced Properties > Returning** from the context menu or from the **Selected** menu.
2. Click the **Returning** tab in the **Editor**.

**Figure 119–39 Returning Tab**



Use the following information to enter data in each field on the tab:

Field	Description
<b>Insert</b>	These options apply to insert operations:
<b>Name</b>	Click <b>Add</b> to add a database field to this ReturningPolicy for insert operations.
<b>Return-only</b>	When selected, TopLink only returns a value for this field; it will not include the field in the insert. When not selected, TopLink returns a value for this field and includes the value in the insert.
<b>Update</b>	These options apply to update operations:
<b>Name</b>	Click <b>Add</b> to add a database field to this ReturningPolicy for update operations

To remove a database field from the descriptor’s ReturningPolicy, select the field in the **Insert** or **Update** window and click **Remove**.

---

**Note:** If you are using TopLink Workbench, you cannot configure a returning policy for an attribute mapped with a transformation mapping (see [Section 27.13, "Transformation Mapping"](#)).

---



## 119.27.2 How to Configure Returning Policy Using Java

You use a `ReturningPolicy` to configure how TopLink handles returning with the attributes of an object on a field-by-field basis. [Table 119–31](#) describes the `ReturnPolicy` methods you use to tell TopLink how to handle a particular database field. Each method takes a `String` or a `DatabaseField` type parameter as field name.

**Table 119–31** *Return Policy Methods*

Method	Applies to SQL Statements of Type...	Writes Current Value of Field to Database?	Returns Database-Generated Result?
<code>addFieldForInsert</code>	INSERT	Yes	Yes
<code>addFieldForInsertReturnOnly</code>	INSERT	No	Yes
<code>addFieldForUpdate</code>	UPDATE	Yes	Yes

You configure a descriptor with a `ReturningPolicy` using `ClassDescriptor` method `setReturningPolicy`.

## 119.28 Configuring Instantiation Policy

The TopLink runtime instantiates new instances of a class according to the instantiation policy you configure on the class's descriptor.

[Table 119–32](#) summarizes which descriptors support an instantiation policy.

**Table 119–32** *Descriptor Support for Instantiation Policy*

Descriptor	How to Use Oracle JDeveloper	How to Configure Instantiation Policy Using TopLink Workbench	How to Configure Instantiation Policy Using Java
Relational Descriptors	✓	✓	✓
Object-Relational Data Type Descriptors			✓
EIS Descriptors	✓	✓	✓
XML Descriptors	✓	✓	✓

You can specify one of the following types of instantiation policy:

- **Default:** TopLink creates a new instance of a class by calling the class's default constructor.
- **Method:** TopLink creates a new instance of a class by calling a public static method that you define on the class descriptor.
- **Factory:** TopLink creates a new instance of a class by calling the appropriate methods on a separate class that you implement according to the Factory design pattern.

### 119.28.1 How to Configure Instantiation Policy Using TopLink Workbench

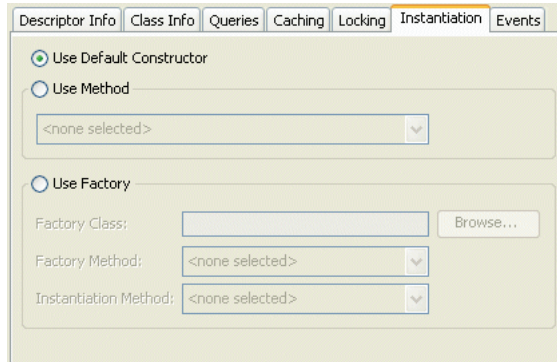
To set the instantiation policy for a descriptor, use this procedure:

1. In the **Navigator**, select a descriptor.

If the Instantiation advanced property is not visible for the descriptor, right-click the descriptor and choose **Select Advanced Properties > Instantiation** from the context menu or from the **Selected** menu.

2. Click the **Instantiation** tab.

**Figure 119–40 Instantiation Tab**



Use the following information to enter data in each field on the tab:

Field	Description
<b>Use Default Constructor</b>	Specify if the default constructor of the class instantiates a new instance.
<b>Use Method</b>	Specify a method to execute to create objects from the database.
<b>Method</b>	Select the name of a method to be executed to create objects from the database. The method must be a public, static method on the descriptor's class and must return a new instance of the object.
<b>Use Factory</b>	Specify an object factory method.
<b>Factory Class</b>	Select the class of the factory object that creates the new instances.
<b>Factory Method</b>	Select the method to be used to obtain a factory object. Choose <code>&lt;nothing&gt;</code> to use the default constructor.
<b>Instantiation Method</b>	Select the method to be called on the factory object to obtain a new instance that will be populated with data from the data source.

### 119.28.2 How to Configure Instantiation Policy Using Java

Use one of the following `ClassDescriptor` methods to set the appropriate type of instantiation policy:

- `useDefaultConstructorInstantiationPolicy`
- `useMethodInstantiationPolicy`
- `useFactoryInstantiationPolicy`

## 119.29 Configuring Copy Policy

The `TopLink` unit of work feature must be able to produce an exact copy (clone) persistent objects. [Table 119–33](#) summarizes which descriptors support a copy policy.

**Table 119–33** Configuring Descriptors with a Copy Policy

Descriptor	How to Use Oracle JDeveloper	How to Configure Copy Policy Using TopLink Workbench	How to Configure Copy Policy Using Java
Relational Descriptors	✓	✓	✓
Object-Relational Data Type Descriptors			✓
EIS Descriptors	✓	✓	✓
XML Descriptors	✓	✓	✓

TopLink supports the following two ways of copying objects:

- **Instantiation policy:** By default, TopLink creates a new copy of an object by using the currently configured instantiation policy (see [Section 119.28, "Configuring Instantiation Policy"](#)).
- **Method:** TopLink creates a new copy of an object by calling a method on the object that you specify. For example, you can specify the object's `clone` method (or any other appropriate method on the object). Note that the `clone` method should return a shallow clone of the object.

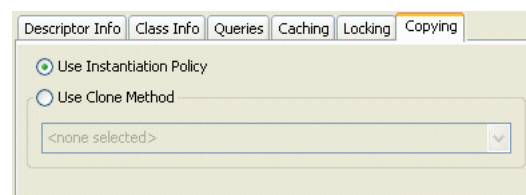
### 119.29.1 How to Configure Copy Policy Using TopLink Workbench

To specify the copy policy for a descriptor, use this procedure:

1. Select a descriptor in the **Navigator**. Its properties appear in the Editor.

If the **Copying** advanced property is not visible for the descriptor, right-click the descriptor and choose **Select Advanced Properties > Copying** from the context menu or from the **Selected** menu.

2. Click the **Copying** tab in the Editor.

**Figure 119–41** Copying Tab

Use the following information to enter data in each field on the tab:

Field	Description
<b>Use Instantiation Policy</b>	Creates a new instance of the object using the descriptor's instantiation policy (see <a href="#">Section 119.28, "Configuring Instantiation Policy"</a> ).
<b>Use Clone Method</b>	Specifies whether or not to call the <code>clone</code> method of the object. Select a method from the list.

## 119.29.2 How to Configure Copy Policy Using Java

Use one of the following `ClassDescriptor` methods to set the appropriate type of copy policy:

- `useCloneCopyPolicy()`: the object must provide a `clone` method
- `useCloneCopyPolicy(java.lang.String cloneMethodName)`
- `useInstantiationCopyPolicy()`

## 119.30 Configuring Change Policy

Use a change policy to specify how TopLink should track changes made to objects after you register them with a unit of work. [Table 119–34](#) summarizes which descriptors support a change policy.

**Table 119–34** Descriptor Support for Change Policy

Descriptor	Deferred Change Detection Policy	Object-Level Change Tracking Policy	Attribute Change Tracking Policy	How to Use Oracle JDeveloper	How to Use TopLink Workbench	How to Configure Change Policy Using Java
Relational Descriptors <sup>1</sup>	✓	✓	✓			✓
Object-Relational Data Type Descriptors	✓	✓	✓			✓
EIS Descriptors <sup>2</sup>	✓	✓	✓			✓
XML Descriptors						

<sup>1</sup> Relational class descriptors only (see [Section 22.2.1.1](#), "Creating Relational Class Descriptors").

<sup>2</sup> EIS root descriptors only (see [Section 75.2.1.1](#), "EIS Root Descriptors").

By default, TopLink uses the deferred change detection policy.

For JPA entities or POJO classes that you configure for weaving, TopLink weaves value holder indirection for one-to-one mappings. If you want TopLink to weave change tracking and your application includes collection mappings (one-to-many or many-to-many), then you must configure all collection mappings to use transparent indirect container indirection only (you may not configure your collection mappings to use eager loading nor value holder indirection).

Mutable basic mappings affect the overhead of change tracking. TopLink can only weave an attribute change tracking policy for immutable mappings.

TopLink logs a warning message at the `CONFIG` log level if you try to weave a descriptor that does not support change policy.

TopLink supports alternative change policies (policies other than `DeferredChangeDetectionPolicy`) for attributes that use a subset of the mappings that TopLink supports.

For CMP and JPA applications deployed to OC4J TopLink automatically uses the attribute change tracking policy.

For more information, see the following:

- [Section 113.2.3](#), "Unit of Work and Change Policy"
- [Section 113.2.3.4](#), "Change Policy Mapping Support"
- [Section 2.10](#), "Using Weaving"

- "Using EclipseLink JPA Weaving" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#Using\\_EclipseLink\\_JPA\\_Weaving](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#Using_EclipseLink_JPA_Weaving)
- Section 2.4.1.4, "Using Method and Direct Field Access"
- Section 2.8.11, "Mutability"

## 119.30.1 How to Configure Change Policy Using Java

This section describes how to configure a descriptor with a change policy using Java, and how to implement persistent classes for those change policies that are intrusive. It includes information on configuring the following:

- [Configuring Deferred Change Detection Policy](#)
- [Configuring Object Change Tracking Policy](#)
- [Configuring Attribute Change Tracking Policy](#)

### 119.30.1.1 Configuring Deferred Change Detection Policy

The `DeferredChangeDetectionPolicy` provides good unit of work commit performance for a wide range of object change characteristics. It is the default change policy. For more information, see [Section 113.2.3.1, "Deferred Change Detection Policy"](#).

Because it is the default, you do not need to explicitly configure this policy.

To configure TopLink to use a `DeferredChangeDetectionPolicy`, create a descriptor amendment method (see [Section 119.35, "Configuring Amendment Methods"](#)) that sets the change policy, as [Example 119–21](#) illustrates.

### 119.30.1.2 Configuring Object Change Tracking Policy

The `ObjectChangeTrackingPolicy` provides improved unit of work commit performance for objects with few attributes, or with many attributes and many changed attributes. For more information, see [Section 113.2.3.2, "Object-Level Change Tracking Policy"](#).

For CMP and JPA applications deployed to an application server, for which TopLink provides CMP integration (see [Section 8.1, "Introduction to the Application Server Support"](#)), when you configure an entity bean's descriptor with an `ObjectLevelChangeTrackingPolicy`, TopLink automatically generates code of a concrete subclass to implement the TopLink `ChangeTracker` interface at deploy time. Configuring an `ObjectLevelChangeTrackingPolicy` prevents TopLink from automatically applying an `AttributeChangeTrackingPolicy` (see [Section 119.30.1.3, "Configuring Attribute Change Tracking Policy"](#)).

To configure TopLink to use an `ObjectChangeTrackingPolicy`, use this procedure:

1. Create a descriptor amendment method (see [Section 119.35, "Configuring Amendment Methods"](#)) that sets the change policy, as [Example 119–21](#) illustrates.

#### **Example 119–21** Setting the `ObjectChangeTrackingPolicy`

```
descriptor.setObjectChangePolicy(new ObjectChangeTrackingPolicy());
```

2. For plain Java objects, code each of your persistent classes to implement the `ChangeTracker` interface, as [Example 119–22](#) illustrates.

**Example 119–22 Implementing the ChangeTracker Interface for the ObjectChangeTrackingPolicy**

```

public class Employee implements ChangeTracker {

    PropertyChangeListener listener;
    String firstName;

    public PropertyChangeListener getTopLinkPropertyChangeListener() {
        return this.listener;
    }

    public void setTopLinkPropertyChangeListener(PropertyChangeListener listener) {
        this.listener = listener;
    }

    ...

    public void setFirstName(String firstName) {
        propertyChange("firstName", getFirstName(), firstName);
        this.firstName = firstName;
    }

    ...

    public void propertyChange(String propertyName, Object oldValue, Object newValue) {
        if (listener != null) {
            if (oldValue != newValue) {
                listener.propertyChange(
                    new PropertyChangeEvent(this, propertyName, oldValue, newValue));
            }
        }
    }
}

```

**119.30.1.3 Configuring Attribute Change Tracking Policy**

The `AttributeChangeTrackingPolicy` provides improved unit of work commit performance for objects with many attributes and few changed attributes. In general, this is the most efficient change policy. It is the default change policy for JPA and EJB 2.*n* CMP applications deployed to OC4J. For more information, see [Section 113.2.3.3, "Attribute Change Tracking Policy"](#).

---



---

**Note:** You cannot use the `AttributeChangeTrackingPolicy` if you are using any instance of `FieldsLockingPolicy` (see [Section 16.4.4, "Optimistic Field Locking Policies"](#)).

---



---

When you deploy a TopLink-enabled EJB 3.0 persistent application or EJB 2.*n* CMP application to OC4J, TopLink automatically configures your persistent classes to use the `AttributeChangeTrackingPolicy` and, using bytecode weaving (EJB 3.0) or code generation (EJB 2.*n*), configures your persistence classes to implement the TopLink `ChangeTracker` interface. In this case, you do not need to explicitly configure this change policy.

To configure TopLink to use an `AttributeChangeTrackingPolicy` for plain Java objects or other application servers, use this procedure:

1. Create a descriptor amendment method (see [Section 119.35, "Configuring Amendment Methods"](#)) that sets the change policy as [Example 119–23](#) illustrates.

**Example 119–23 Setting the DeferredChangeDetectionPolicy**

- ```
descriptor.setObjectChangePolicy(new AttributeChangeTrackingPolicy());
```
2. Code each of your persistent classes to implement the `ChangeTracker` interface, as [Example 119–24](#) illustrates.

**Example 119–24 Implementing the ChangeTracker Interface for the AttributeChangeTrackingPolicy**

```

public class Employee implements ChangeTracker {

    PropertyChangeListener listener;
    String firstName;

    public PropertyChangeListener getTopLinkPropertyChangeListener() {
        return this.listener;
    }

    public void setTopLinkPropertyChangeListener(PropertyChangeListener listener) {
        this.listener = listener;
    }
    ...
    public void setFirstName(String firstName) {
        propertyChange("firstName", getFirstName(), firstName);
        this.firstName = firstName;
    }
    ...
    public void propertyChange(String propertyName, Object oldValue, Object newValue) {
        if (listener != null) {
            if (oldValue != newValue) {
                listener.propertyChange(
                    new PropertyChangeEvent(this, propertyName, oldValue, newValue));
            }
        }
    }
}

```

## 119.31 Configuring a History Policy

If you want to use historical sessions (see [Section 87.6, "Historical Sessions"](#)) to execute historical queries (see [Section 108.11, "Historical Queries"](#)) against a historical schema of your own design, configure your descriptors with a TopLink HistoryPolicy that describes your historical schema.

If you are using Oracle Database platform for Oracle9i Database (or later), you can query the historical versions of objects automatically maintained by Oracle Database without the need for a history policy. For more information, see [Section 93.1.1, "How to Configure Historical Sessions Using an Oracle Platform"](#).

[Table 119–35](#) summarizes which descriptors support history policy configuration.

**Table 119–35 Descriptor Support for History Policy Configuration**

| Descriptor                              | How to Use Oracle JDeveloper | How to Use TopLink Workbench | How to Configure a History Policy Using Java |
|-----------------------------------------|------------------------------|------------------------------|----------------------------------------------|
| Relational Descriptors                  |                              |                              | ✓                                            |
| Object-Relational Data Type Descriptors |                              |                              | ✓                                            |
| EIS Descriptors                         |                              |                              |                                              |
| XML Descriptors                         |                              |                              |                                              |

There are many ways to configure a historical database schema. TopLink supports several historical schema configurations that you can describe with a HistoryPolicy (see [Section 87.6.1, "Historical Session Limitations"](#)).

**Example Historical Schema**

As shown in [Table 119–36](#) and [Table 119–37](#), a common approach is to define a special history table to store past versions of an object: one history table for each regular table that requires historical persistence. The history table typically has the same fields as the corresponding regular table plus fields (such as row start and end) used to define an interval that represents the life time of a particular version.

---

**Note:** TopLink assumes that the current version of an object corresponds to the historical table row whose row end field is NULL.

---

TopLink will include the history tables described by a `HistoryPolicy` when you execute a historical query.

[Table 119–36](#) shows the schema for an `EMPLOYEE` table. The table currently contains one `EMPLOYEE` instance.

**Table 119–36 Example Table for EMPLOYEE**

| EMP_ID | F_NAME | L_NAME | SALARY |
|--------|--------|--------|--------|
| 1      | Jane   | Doe    | 55000  |

[Table 119–37](#) shows one possible history table `EMPLOYEE_HIST` that stores historical versions of employees. The table contains the current `EMPLOYEE` (the version with a `ROW_END` value of `NULL`) and one historical version.

**Table 119–37 Example History Table EMPLOYEE\_HIST**

| EMP_ID | F_NAME | L_NAME | SALARY | ROW_START  | ROW_END    |
|--------|--------|--------|--------|------------|------------|
| 1      | Jane   | Doe    | 50000  | 29/08/2004 | 31/08/2004 |
| 1      | Jane   | Doe    | 55000  | 31/08/2004 | NULL       |

Because every record has a start and end interval, the history table can store multiple versions of the same object (with the same primary key). The unique identifier of a particular version is given by the existing primary key, plus the value of the start field. For example, in [Table 119–37](#), the unique identifier of the current version is given by  $(EMP\_ID, START) = (1, 31/08/2004)$ .

### 119.31.1 How to Configure a History Policy Using Java

[Example 119–25](#) shows how to describe the schema shown in [Table 119–36](#) and [Table 119–37](#) using the TopLink `HistoryPolicy`:

**Example 119–25 HistoryPolicy for One Table**

```
HistoryPolicy policy = new HistoryPolicy();
policy.addStartFieldName("ROW_START");
policy.addEndFieldName("ROW_END");
policy.addHistoryTableName("EMPLOYEE", "EMPLOYEE_HIST");
// Assuming database triggers or stored procedures update history tables
policy.setShouldHandleWrites(false);
```

```
employeeDescriptor.setHistoryPolicy(policy);
```

You can specify more than one table with a `HistoryPolicy` as shown in [Example 119–26](#). In this example, all history tables have a start field named `ROW_START` but the `EMPLOYEE_HIST` and `SALARY_HIST` tables have different end fields.



To avoid ambiguity, the end field names are prefixed with their respective history table names.

**Example 119–26 HistoryPolicy for Multiple Tables**

```
HistoryPolicy policy = new HistoryPolicy();
policy.addStartFieldName("ROW_START");
policy.addEndFieldName("EMPLOYEE_HIST.ROW_END");
policy.addEndFieldName("SALARY_HIST.VALID_UNTIL");
policy.addHistoryTableName("EMPLOYEE", "EMPLOYEE_HIST");
policy.addHistoryTableName("SALARY", "SALARY_HIST");
// Assuming database triggers or stored procedures update history tables
policy.setShouldHandleWrites(false);

employeeDescriptor.setHistoryPolicy(policy);
```

### 119.31.1.1 Configuring Write Responsibility

Use `HistoryPolicy` method `setShouldHandleWrites` to specify whether or not `TopLink` is responsible for writing data to history tables. By default, `setShouldHandleWrites` is set to `true`.

Either the database or `TopLink` can be responsible for writing data to the history tables.

You can configure the database to write data to history tables by way of triggers or stored procedures that customize create, insert, and delete operations to modify both the regular table and the history table appropriately.

## 119.32 Configuring Wrapper Policy

`TopLink` lets you use wrappers (or proxies) in cases where the persistent class is not the same class that is to be presented to users.

For example, in the EJB specification prior to 3.0, the entity bean class (the class that implements `javax.ejb.EntityBean`) is persistent, but is hidden from users who interact with a class that implements `javax.ejb.EJBObject` (local or remote interface class). In this example, the `EJBObject` acts as a proxy (or wrapper) for the `EntityBean`.

In cases where such a wrapper is used, `TopLink` continues to make the class specified in the descriptor persistent, but returns the appropriate instance of the wrapper whenever a persistent object is requested.

Table 119–38 summarizes which descriptors support a wrapper policy.

**Table 119–38 Descriptor Support for Wrapper Policy**

| Descriptor                              | How to Use Oracle JDeveloper | How to Use TopLink Workbench | How to Configure Wrapper Policy Using Java |
|-----------------------------------------|------------------------------|------------------------------|--------------------------------------------|
| Relational Descriptors                  |                              |                              | ✓                                          |
| Object-Relational Data Type Descriptors |                              |                              | ✓                                          |
| EIS Descriptors                         |                              |                              | ✓                                          |
| XML Descriptors                         |                              |                              |                                            |

Use a wrapper policy to tell TopLink how to create wrappers for a particular persistent class, and how to obtain the underlying persistent object from a given wrapper instance.

If you specify a wrapper policy, TopLink uses the policy to *wrap* and *unwrap* persistent objects as required:

- Wrapper policies implement the interface `oracle.toplink.descriptors.WrapperPolicy`.
- A wrapper policy is specified by setting the wrapper policy for the TopLink descriptor.
- By default, no wrapper policy is used (the wrapper policy for a descriptor is null by default).

---

---

**Note:** Wrapper policies are advanced TopLink options. Using a wrapper policy may not be compatible with some TopLink Workbench features.

---

---

For CMP descriptors, the EJB wrapper policy is automatically configured during deployment, so does not need to be set in the descriptor (see [Section 119.18](#), "Configuring a Descriptor with EJB CMP and BMP Information").

For BMP descriptors, you must set a `BMPWrapperPolicy` on the descriptor. The `BMPWrapperPolicy` includes the bean's information including the bean-name, primary-key-class, home-interface, and remote-interface.

Wrapper policies cannot be set using Oracle JDeveloper or TopLink Workbench and can be set only using Java code (see [Section 119.32.1](#), "How to Configure Wrapper Policy Using Java").

### 119.32.1 How to Configure Wrapper Policy Using Java

Use the `ClassDescriptor` method `setWrapperPolicy` to set the appropriate instance of `WrapperPolicy`.

[Example 119–27](#) shows how to amend a BMP descriptor with the required BMP information.

**Example 119–27 Configuring a BMP Wrapper Policy**

```
public static void addToDescriptor(ClassDescriptor descriptor) {
    BMPWrapperPolicy policy = new BMPWrapperPolicy(
        "employee",
        EmployeeHome.class,
        EmployeePK.class,
        Employee.class,
        new Hashtable());
    descriptor.setWrapperPolicy(policy);
}
```

## 119.33 Configuring Fetch Groups

By default, when you execute an object-level read query for a particular object class, TopLink returns all the persistent attributes mapped in the object's descriptor. With this single query, all the object's persistent attributes are defined, and calling their `get` methods returns the value directly from the object.

When you are interested in only some of the attributes of an object, it may be more efficient to return only a subset of the object's attributes using a fetch group.

Using a fetch group, you can define a subset of an object's attributes and associate the fetch group with either a `ReadObjectQuery` or `ReadAllQuery` query. When you execute the query, TopLink retrieves only the attributes in the fetch group. TopLink automatically executes a query to fetch all the attributes excluded from this subset when and if you call a `get` method on any one of the excluded attributes.

You can define more than one fetch group for a class. You can optionally designate at most one such fetch group as the default fetch group. If you execute either a `ReadObjectQuery` or `ReadAllQuery` query without specifying a fetch group, TopLink will use the default fetch group, unless you configure the query otherwise (see [Section 111.3.1, "How to Configure Default Fetch Group Behavior"](#)).

You can use fetch groups in CMP or JPA projects for EJB objects, as well as for POJO classes.

Before using fetch groups, Oracle recommends that you perform a careful analysis of system use. In many cases, the extra queries required to load attributes not in the fetch group could well offset the gain from the partial attribute loading. For more information about optimizing read performance, see [Section 12.12.9, "Read Optimization Examples"](#).

[Table 119–39](#) summarizes which descriptors support fetch group configuration.

**Table 119–39 Descriptor Support for Fetch Group Configuration**

| Descriptor                              | How to Use Oracle JDeveloper | How to Use TopLink Workbench | How to Configure Fetch Groups Using Java |
|-----------------------------------------|------------------------------|------------------------------|------------------------------------------|
| Relational Descriptors                  |                              |                              | ✓                                        |
| Object-Relational Data Type Descriptors |                              |                              | ✓                                        |
| EIS Descriptors                         |                              |                              |                                          |
| XML Descriptors                         |                              |                              |                                          |

For JPA entities or POJO classes that you configure for weaving, TopLink uses fetch groups to improve performance.

This section describes how to create a fetch group, store it in a descriptor, and optionally designate a fetch group as the default fetch group for its descriptor reference class.

For more information, see the following:

- [Section 16.2.4, "Fetch Groups"](#)
- [Section 108.7.1.6, "Fetch Groups and Object-Level Read Queries"](#)
- [Section 111.3.1, "How to Configure Default Fetch Group Behavior"](#)
- [Section 2.10, "Using Weaving"](#)
- "Using EclipseLink JPA Weaving" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#Using\\_EclipseLink\\_JPA\\_Weaving](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#Using_EclipseLink_JPA_Weaving)

### 119.33.1 How to Configure Fetch Groups Using Java

To configure a fetch group, use a descriptor amendment method (see [Section 119.35](#), "Configuring Amendment Methods") as [Example 119–28](#) shows.

**Example 119–28 Configuring a Fetch Group**

```
//Create a FetchGroupManager for the descriptor
descriptor.setFetchGroupManager(new FetchGroupManager());
// Create a FetchGroup
FetchGroup group = new FetchGroup("nameOnly");
// Add attributes to FetchGroup. Alternatively, use
// FetchGroup method addAttributes, passing in a Set of String attribute names
group.addAttribute("firstName");
group.addAttribute("lastName");
// Add the FetchGroup to the FetchGroupManager
descriptor.getFetchGroupManager().addFetchGroup(group);
//Set the default fetch group
descriptor.getFetchGroupManager().setDefaultFetchGroup(group);
```

Each instance of `FetchGroup` that you store in a descriptor must be configured with a fetch group name that is unique for that descriptor (that is, each descriptor owns a set of named fetch groups).

When configuring fetch groups, note that the primary key fields and other required fields (such as inheritance type and optimistic lock version) are always included in all fetch groups.

Fetch groups can include direct and relationship attributes. Including a relationship attribute in a fetch group does not cause the relationship to be joined or instantiated: joining and indirection are set independently of fetch groups.

After you add a fetch group to a descriptor, you can configure a `ReadObjectQuery` or `ReadAllQuery` query with this fetch group by name (`nameOnly`) or rely on `TopLink` to use this fetch group by default. For more information, see [Section 111.3](#), "Using Queries with Fetch Groups".

## 119.34 Configuring a Descriptor Customizer Class

A descriptor customizer class is a Java class that implements the `oracle.toplink.tools.sessionconfiguration.DescriptorCustomizer` interface and provides a default (zero-argument) constructor. You can use a descriptor customizer to customize a descriptor at run time on a loaded session before login occurs, similar to how you can use an amendment method (see [Section 119.35](#), "Configuring Amendment Methods").

[Table 119–40](#) summarizes which sessions support customizer class configuration.

**Table 119–40 Descriptor Support for Customizer Class Configuration**

| Descriptor                              | How to Use Oracle JDeveloper | How to Use TopLink Workbench | How to Configure Customizer Class Using Java |
|-----------------------------------------|------------------------------|------------------------------|----------------------------------------------|
| Relational Descriptors                  | ✓                            |                              | ✓                                            |
| Object-Relational Data Type Descriptors | ✓                            |                              | ✓                                            |
| EIS Descriptors                         | ✓                            |                              | ✓                                            |

**Table 119–40 (Cont.) Descriptor Support for Customizer Class Configuration**

| Descriptor      | How to Use Oracle JDeveloper | How to Use TopLink Workbench | How to Configure Customizer Class Using Java |
|-----------------|------------------------------|------------------------------|----------------------------------------------|
| XML Descriptors | ✓                            |                              | ✓                                            |

For more information, see [Section 16.2.6, "Descriptor Customization"](#).

### 119.34.1 How to Configure Customizer Class Using Java

When using Java, create a customize class that implements the `oracle.toplink.tools.sessionconfiguration.DescriptorCustomizer` interface. [Example 119–29](#) illustrates the creation of a descriptor customizer.

#### **Example 119–29 Creating a SessionCustomizer Class**

```
import oracle.toplink.tools.sessionconfiguration.DescriptorCustomizer;
import oracle.toplink.descriptors.ClassDescriptor;

public class EmployeeDescriptorCustomizer implements DescriptorCustomizer {

    public void customize(ClassDescriptor descriptor) {
        descriptor.setReadOnly();
    }
}
```

## 119.35 Configuring Amendment Methods

Some TopLink descriptor features cannot be configured from Oracle JDeveloper or TopLink Workbench. To use these features, you must write a Java method to amend the descriptor after it is loaded as part of the project. This method must have the following characteristics:

- Be public static.
- Take a single parameter of type `oracle.toplink.descriptors.ClassDescriptor`.

In the implementation of this method, you can configure advanced features of the descriptor using any of the public descriptor and mapping API.

[Table 119–41](#) summarizes which descriptors support amendment methods.

**Table 119–41 Descriptor Support for Amendment Methods**

| Descriptor                              | How to Use Oracle JDeveloper | How to Configure Amendment Methods Using TopLink Workbench | How to Use Java |
|-----------------------------------------|------------------------------|------------------------------------------------------------|-----------------|
| Relational Descriptors                  | ✓                            | ✓                                                          |                 |
| Object-Relational Data Type Descriptors |                              |                                                            |                 |
| EIS Descriptors                         | ✓                            | ✓                                                          |                 |
| XML Descriptors                         | ✓                            | ✓                                                          |                 |

This section describes how to associate an amendment method with a descriptor.

For more information about how to implement an amendment method, see [Section 16.2.7, "Amendment and After-Load Methods"](#).

Alternatively, you can use a descriptor customizer class (see [Section 119.34, "Configuring a Descriptor Customizer Class"](#)

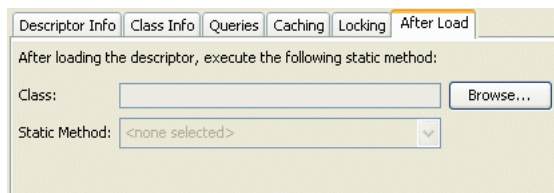
To customize a session, use a session customizer class (see [Section 89.8, "Configuring a Session Customizer Class"](#)).

### 119.35.1 How to Configure Amendment Methods Using TopLink Workbench

To use an amendment method with a descriptor (after it is loaded as part of the project) use this procedure:

1. Select a descriptor in the **Navigator**. Its properties appear in the Editor.  
 If the **After load** advanced property is not visible for the descriptor, right-click the descriptor and choose **Select Advanced Properties > After Load** from context menu or from the **Selected** menu.
2. Click the **After Load** tab in the Editor.

**Figure 119–42 After Load Tab**



| Field         | Description                                                                                                                                                                                                                                        |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Class         | Click <b>Browse</b> and choose the class of the method to execute.                                                                                                                                                                                 |
| Static Method | Use the <b>Static Method</b> list to choose the static method to execute at run time, after loading the descriptor. The method must be public static and take a single attribute of type <code>oracle.toplink.descriptors.ClassDescriptor</code> . |

# Part XXVIII

---

## Creation and Configuration of Mappings

This part describes general mapping information. It contains the following chapters:

- [Chapter 120, "Creating a Mapping"](#)

This chapter contains procedures for creating TopLink mappings.

- [Chapter 121, "Configuring a Mapping"](#)

This chapter explains how to configure TopLink mapping options common to two or more mapping types.





---

---

## Creating a Mapping

This chapter describes how to create TopLink mappings.

This chapter includes the following sections:

- [Introduction to Mapping Creation](#)
- [Creating Mappings Manually During Development](#)
- [Creating Mappings Automatically During Development](#)
- [Creating Mappings Automatically During Deployment](#)
- [Creating Mappings to Oracle LOB Database Objects](#)
- [Removing Mappings](#)

### 120.1 Introduction to Mapping Creation

You can create a database mapping using Oracle JDeveloper, TopLink Workbench, or Java code. Oracle recommends using either Oracle JDeveloper or TopLink Workbench to create and manage your mappings.

For more information on creating mappings in Java, see Oracle Fusion Middleware Java API Reference for Oracle TopLink

For complete information on the various types of mapping that TopLink supports, see [Section 17.1, "Mapping Types"](#).

During development, you can create mappings individually (see [Section 120.2, "Creating Mappings Manually During Development"](#)) or you can allow TopLink Workbench to automatically map all attributes (see [Section 120.3, "Creating Mappings Automatically During Development"](#)).

For CMP projects using OC4J, you can also configure TopLink to create mappings automatically at deployment time (see [Section 120.4, "Creating Mappings Automatically During Deployment"](#)).

For JPA projects, you can create mappings using JPA annotations (see "Using Metadata Annotations" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Introduction\\_to\\_EclipseLink\\_JPA\\_%28ELUG%29#Using\\_Metadata\\_Annotations](http://wiki.eclipse.org/Introduction_to_EclipseLink_JPA_%28ELUG%29#Using_Metadata_Annotations)). JPA lets you create mappings automatically at run time.

After you create a mapping, you must configure its various options (see [Chapter 121, "Configuring a Mapping"](#)).

## 120.2 Creating Mappings Manually During Development

You can manually create a mapping from each persistent field of a class to a data source using Oracle JDeveloper, TopLink Workbench, or Java code. Oracle recommends that you use Oracle JDeveloper or TopLink Workbench.

### 120.2.1 How to Create Mappings Manually During Development Using TopLink Workbench

To manually create a mapping using TopLink Workbench, use this procedure:

1. Select a descriptor in the **Navigator**. Its properties appear in the Editor.
2. On the **Descriptor Info** tab, associate the descriptor with its data source:
  - a. For a relational project, in the **Editor**, select the table for this descriptor from the **Associated Table** menu.

The TopLink Workbench populates this menu with tables from the database login you associated with your project. For more information, see [Section 20.4, "Configuring Login Information at the Project Level"](#).

- b. For an XML project, in the **Editor**, select the appropriate schema context for this descriptor by clicking on the **Browse** button next to the **Schema Context** field.

The TopLink Workbench displays the schema context from the XML schema you associated with your project. For more information, see [Section 5.6, "Using XML Schemas"](#).

3. In the **Navigator**, expand the descriptor to display its attributes.
4. Select an attribute and do one of the following:
  - a. Click the appropriate mapping on the toolbar (see [Section 5.3.2.2, "Using Context Toolbar"](#)).
  - b. Right-click the attribute and select **Map As >** to choose a specific mapping type from the context menu (see [Section 5.3.1.2, "Using Context Menus"](#)).

Continue with [Chapter 121, "Configuring a Mapping"](#) to complete the mapping.

### 120.2.2 How to Create Mappings During Development Using Java

You create mappings using the constructor of the particular mapping type, as the following examples show:

**Example 120–1 Creating Relational One-to-One Mapping**

```
oracle.toplink.mappings.OneToOneMapping oom = new OneToOneMapping();
```

**Example 120–2 Creating Relational Direct Collection Mapping**

```
oracle.toplink.mappings.DirectCollectionMapping dcm =
    new DirectCollectionMapping();
```

**Example 120–3 Creating Object-Relational Data Type Structure Mapping**

```
oracle.toplink.objectrelational.StructureMapping sm = new StructureMapping();
```

**Example 120–4 Creating Object-Relational Data Type Array Mapping**

```
oracle.toplink.objectrelational.ArrayMapping am = new ArrayMapping();
```

## 120.3 Creating Mappings Automatically During Development

For relational database projects, Oracle JDeveloper and TopLink Workbench can automatically map class attributes to a similarly named database field. For example, these tools can map the attribute `province` to the database field `PROV`, the attribute `street` to the field `ST`, and the attribute `postalCode` to the field `POSTAL_CODE`.

The Automap function creates mappings only for unmapped attributes—it does not change previously defined mappings.

You can automatically map classes for an entire project or for specific classes or descriptors.

---

**Note:** Associating a descriptor with a database table (see [Section 23.2, "Configuring Associated Tables"](#)) before using the Automap function can aid the automapper if it cannot guess the correct table for a class.

---

### 120.3.1 How to Create Mappings Automatically During Development Using TopLink Workbench

To automatically map *all* the descriptors in a project, right-click the project icon in the **Navigator** window and choose **Automap** from the context menu or choose **Selected > Automap** from the menu.

To automatically map a *specific* descriptor or attribute, choose the descriptor or attributes and right-click, and then select **Automap** from the context menu or choose **Selected > Automap** from the menu.

## 120.4 Creating Mappings Automatically During Deployment

If you create a project from an OC4J EJB CMP EAR at deployment time, you can take advantage of the TopLink default mapping feature to automatically create mappings at deployment time.

For more information, see the following:

- [Section 19.5, "Creating a Project from an OC4J EJB CMP EAR at Deployment Time"](#)
- [Section 17.2.3.4, "Default Mapping in EJB 2.n CMP Projects Using OC4J at Run Time"](#)

## 120.5 Creating Mappings to Oracle LOB Database Objects

In Oracle Database, large amounts of binary or character data is stored as a BLOB (binary large object) or CLOB (character large object), respectively. Depending on the size of the LOB value and your Oracle Database version, the value may be stored either inside or outside of the table, as follows:

- With Oracle8i Database Release 1.6 and earlier, LOB values less than 4K are stored inline; values more than 4K are stored outside the table.
- With Oracle8i Database Release 1.7 and later, LOB values less than 5.9K are stored inline; values more than 5.9K are stored outside the table.

A client application (such as Oracle TopLink) must use a LOB locator to write a LOB value, if the value is stored outside of the database. The Oracle JDBC OCI driver and

server driver handle these LOB (large object) values differently than the Oracle JDBC thin driver.

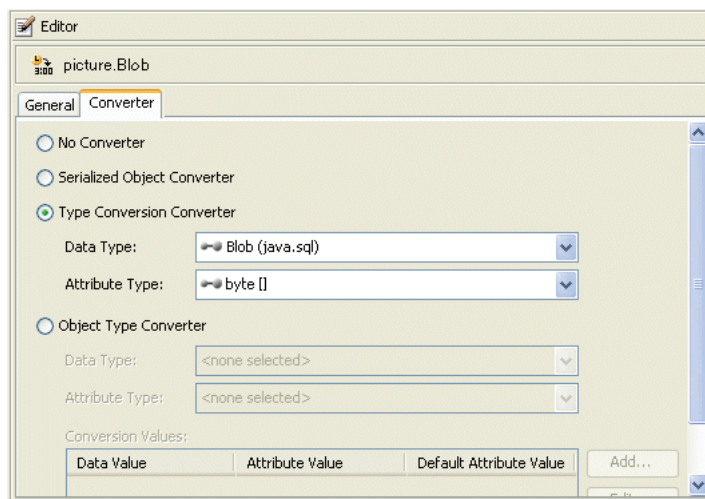
## 120.5.1 How to Create Mappings to Oracle LOB Database Objects Using the Oracle JDBC Thin Driver

When using the Oracle JDBC thin driver, TopLink is responsible for acquiring the LOB locator before writing the value. You can use a type conversion mapping (see [Section 121.10, "Configuring a Type Conversion Converter"](#)) to retrieve the locator during a commit operation.

Use this procedure to map LOB values using the Oracle JDBC thin driver:

1. Use a type conversion mapping to map the attributes of a TopLink descriptor to a LOB value. [Figure 120–1](#) shows a type conversion mapping to a BLOB value in TopLink Workbench. [Example 120–5](#) shows the Java code for the same mapping.

**Figure 120–1 Mapping a BLOB in TopLink Workbench**



**Example 120–5 Mapping a BLOB in Java Code**

```
TypeConversionMapping pictureMapping = new TypeConversionMapping();
pictureMapping.set.AttributeName("picture");
pictureMapping.set.FieldName("IMAGE.PICTURE");
pictureMapping.set.FieldClassification(java.sql.Blob.class);
descriptor.addMapping(pictureMapping);
```

2. Acquire the DatabaseLogin from the session.

```
DatabaseLogin login = session.getLogin();
```

3. Configure a platform that provides locator support, as follows:

- For Oracle8i Database, use the `oracle.toplink.platform.database.oracle.Oracle8Platform` class:

```
login.usePlatform(new Oracle8Platform());
```

- For Oracle9i Database, use the `oracle.toplink.platform.database.oracle.Oracle9Platform` class:

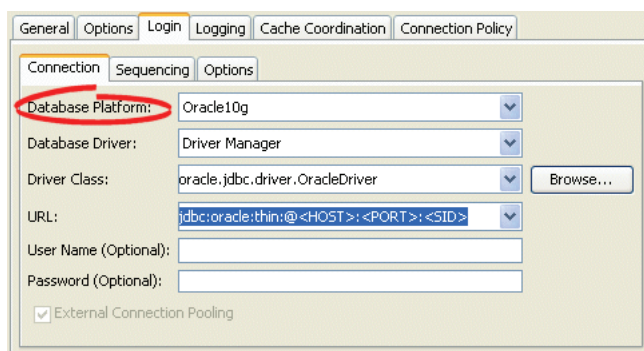
```
login.usePlatform(new Oracle9Platform());
```

- Oracle Database 10g, use the `oracle.toplink.platform.database.oracle.Oracle10Platform` class:
 

```
login.usePlatform(new Oracle10Platform());
```
  - Oracle Database 11g, use the `oracle.toplink.platform.database.oracle.Oracle11Platform` class:
 

```
login.usePlatform(new Oracle10Platform());
```
- In TopLink Workbench, select the appropriate platform in the Database editor.

**Figure 120–2 Selecting Database Platform in TopLink Workbench**



## 120.6 Removing Mappings

If you are using Oracle JDeveloper or TopLink Workbench, you can unmap any mapped attribute.

### 120.6.1 How to Remove Mappings Using TopLink Workbench

To unmap an attribute (that is, remove its mapping), use this procedure:



Select the attribute in the **Navigator** window and click **Unmap**. You can also unmap the attribute by right-clicking the attribute and selecting **Map As > Unmapped** from the context menu.

To unmap all the attributes in a descriptor or Java package, use this procedure:



Right-click the descriptor or Java package in the **Navigator** window and select **Unmap > Unmap Selected Descriptor** or **Unmap All Descriptors in Package** from the context menu.

### 120.6.2 How to Remove Mappings Using Java

Use the `ClassDescriptor` method `removeMappingForAttributeName` to unmap an attribute.



---

---

## Configuring a Mapping

This chapter describes how to configure TopLink mapping options common to two or more mapping types.

This chapter includes the following sections:

- [Configuring Common Mapping Options](#)
- [Configuring Read-Only Mappings](#)
- [Configuring Indirection \(Lazy Loading\)](#)
- [Configuring XPath](#)
- [Configuring a Default Null Value at the Mapping Level](#)
- [Configuring Method or Direct Field Accessing at the Mapping Level](#)
- [Configuring Private or Independent Relationships](#)
- [Configuring Mapping Comments](#)
- [Configuring a Serialized Object Converter](#)
- [Configuring a Type Conversion Converter](#)
- [Configuring an Object Type Converter](#)
- [Configuring a Simple Type Translator](#)
- [Configuring a JAXB Typesafe Enumeration Converter](#)
- [Configuring Container Policy](#)
- [Configuring Attribute Transformer](#)
- [Configuring Field Transformer Associations](#)
- [Configuring Mutable Mappings](#)
- [Configuring Bidirectional Relationship](#)
- [Configuring the Use of a Single Node](#)
- [Configuring the Use of CDATA](#)

Table 121–1 lists the types of TopLink mappings that you can configure and provides a cross-reference to the type-specific chapter that lists the configurable options supported by that type.

**Table 121–1** *Configuring TopLink Mappings*

---

| <b>If you are creating...</b> | <b>See...</b>                                  |
|-------------------------------|------------------------------------------------|
| Relational Mappings           | Chapter 28, "Configuring a Relational Mapping" |

---

**Table 121–1 (Cont.) Configuring TopLink Mappings**

| If you are creating...               | See...                                                           |
|--------------------------------------|------------------------------------------------------------------|
| Object-Relational Data Type Mappings | Chapter 41, "Configuring an Object-Relational Data Type Mapping" |
| EIS Mappings                         | Chapter 78, "Configuring an EIS Mapping"                         |
| XML Mappings                         | Chapter 54, "Configuring an XML Mapping"                         |

Table 121–2 lists the configurable options shared by two or more TopLink mapping types.

For more information, see the following:

- Section 120.1, "Introduction to Mapping Creation"
- Chapter 17, "Introduction to Mappings"

## 121.1 Configuring Common Mapping Options

Table 121–2 lists the configurable options shared by two or more TopLink mapping types. In addition to the configurable options described here, you must also configure the options described for the specific mapping types (see Section 17.1, "Mapping Types"), as shown in Table 121–1

**Table 121–2 Common Mapping Options**

| Option to Configure                                                                                                    | Oracle JDeveloper | TopLink Workbench | Java |
|------------------------------------------------------------------------------------------------------------------------|-------------------|-------------------|------|
| Read-only (see Section 121.2, "Configuring Read-Only Mappings")                                                        | ✓                 | ✓                 | ✓    |
| Indirection (lazy loading) (see Section 121.3, "Configuring Indirection (Lazy Loading)")                               | ✓                 | ✓                 | ✓    |
| XPath (see Section 121.4, "Configuring XPath")                                                                         | ✓                 | ✓                 | ✓    |
| Default null value (see Section 121.5, "Configuring a Default Null Value at the Mapping Level")                        | ✓                 | ✓                 | ✓    |
| Method or direct field access (see Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level") | ✓                 | ✓                 | ✓    |
| Private or independent relationships (see Section 121.7, "Configuring Private or Independent Relationships")           | ✓                 | ✓                 | ✓    |
| Comments (see Section 121.8, "Configuring Mapping Comments")                                                           | ✓                 | ✓                 |      |
| Serialized object converter (see Section 121.9, "Configuring a Serialized Object Converter")                           | ✓                 | ✓                 | ✓    |
| Serialized type conversion converter (see Section 121.10, "Configuring a Type Conversion Converter")                   | ✓                 | ✓                 | ✓    |
| Object type converter (see Section 121.11, "Configuring an Object Type Converter")                                     | ✓                 | ✓                 | ✓    |
| Simple type translator (see Section 121.12, "Configuring a Simple Type Translator")                                    | ✓                 | ✓                 | ✓    |
| JAXB typesafe enumeration converter (see Section 121.13, "Configuring a JAXB Typesafe Enumeration Converter")          |                   |                   | ✓    |
| Container policy (see Section 121.14, "Configuring Container Policy")                                                  | ✓                 | ✓                 | ✓    |
| Attribute transformer (see Section 121.15, "Configuring Attribute Transformer")                                        | ✓                 | ✓                 | ✓    |



**Table 121–2 (Cont.) Common Mapping Options**

| Option to Configure                                                                               | Oracle JDeveloper | TopLink Workbench | Java |
|---------------------------------------------------------------------------------------------------|-------------------|-------------------|------|
| Field transformer associations (see Section 121.16, "Configuring Field Transformer Associations") | ✓                 | ✓                 | ✓    |
| Mutable mappings (see Section 121.17, "Configuring Mutable Mappings")                             | ✓                 | ✓                 | ✓    |
| Bidirectional relationship (see Section 121.18, "Configuring Bidirectional Relationship")         | ✓                 | ✓                 | ✓    |
| Use of a single node (see Section 121.19, "Configuring the Use of a Single Node")                 | ✓                 | ✓                 | ✓    |
| Use of CDATA (see Section 121.20, "Configuring the Use of CDATA")                                 |                   |                   | ✓    |

## 121.2 Configuring Read-Only Mappings

Mappings that are read-only will not be affected during insert, update, and delete operations.

Use read-only mappings when multiple attributes in an object map to the same fields in the database but only one of the mappings can write to the field.

You can also use read-only mappings with bi-directional many-to-many mappings to designate which mapping will be responsible for updating the many-to-many join table.

---

**Note:** The primary key mappings cannot not be read-only.

---

Mappings defined for the write-lock or class indicator field must be read-only, unless the write-lock is configured not to be stored in the cache or the class indicator is part of the primary key.

Use read-only mappings only if specific mappings in a descriptor are read-only. If the *entire descriptor* is read-only, use the descriptor-level setting (see Section 119.3, "Configuring Read-Only Descriptors").

---

**Note:** Even though read-only mappings are not written to the database, they are merged into the TopLink cache.

---

Table 121–3 summarizes which mappings support this option.

**Table 121–3 Mapping Support for Read-Only**

| Mapping                              | How to Use Oracle JDeveloper | How to Configure Read-Only Mappings Using TopLink Workbench | How to Configure Read-Only Mappings Using Java |
|--------------------------------------|------------------------------|-------------------------------------------------------------|------------------------------------------------|
| Relational Mappings                  | ✓                            | ✓                                                           | ✓                                              |
| Object-Relational Data Type Mappings |                              |                                                             | ✓                                              |
| EIS Mappings                         | ✓                            | ✓                                                           | ✓                                              |
| XML Mappings                         | ✓                            | ✓                                                           | ✓                                              |

## 121.2.1 How to Configure Read-Only Mappings Using TopLink Workbench

To specify a mapping as read-only, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

**Figure 121–1 General Tab, Read-Only Option**

The screenshot shows the 'General' tab of the TopLink Workbench. The 'Database Field' is set to 'ADDRESS.CITY (VARCHAR2)'. There are sections for 'Method Accessing' (with 'Get Method' and 'Set Method' dropdowns), 'Default Null Value' (with 'Type' and 'Value' dropdowns), and 'Read-Only' (with a checked checkbox circled in red). A 'Comment' text area is at the bottom.

Select the **Read-Only** option to set the mapping to be read-only and not affected during update and delete operations.

## 121.2.2 How to Configure Read-Only Mappings Using Java

Use the following `DatabaseMapping` methods to configure the read access of a mapping:

- `readOnly`—configures mapping read access to read-only;
- `readWrite`—configures mapping read access to read and write (default).

[Example 121–1](#) shows how to use these methods with a class that has a read-only attribute named `phones`.

**Example 121–1 Configuring Read Only Mappings in Java**

```
// Map the phones attribute
phonesMapping.setAttributeName("phones");

// Specify read-only
phonesMapping.readOnly();
```

## 121.3 Configuring Indirection (Lazy Loading)

If indirection is not enabled, when TopLink retrieves a persistent object, it retrieves all of the dependent objects to which it refers. When you enable indirection (lazy loading) for an attribute mapped with a relationship mapping, TopLink uses an indirection object as a placeholder for the referenced object: TopLink defers reading the dependent object until you access that specific attribute. This can result in a significant performance improvement, especially if the application is interested only in the contents of the retrieved object rather than the objects to which it refers.

Oracle strongly recommends using indirection for all relationship mappings. Not only does this allow you to optimize data source access, but it also allows TopLink to optimize the unit of work processing, cache access, and concurrency.

Table 121–4 summarizes which mappings support this option.

**Table 121–4 Mapping Support for Indirection**

| Mapping                                          | Value Holder Indirection | Transparent Indirect Container Indirection | Proxy Indirection | How to Use Oracle JDeveloper | How to Configure Indirection Using TopLink Workbench | How to Configure Indirection Using Java |
|--------------------------------------------------|--------------------------|--------------------------------------------|-------------------|------------------------------|------------------------------------------------------|-----------------------------------------|
| Relational Mappings                              |                          |                                            |                   |                              |                                                      |                                         |
| Direct-to-Field Mapping                          |                          | ✓                                          | ✓                 |                              |                                                      | ✓                                       |
| Transformation Mapping                           | ✓                        |                                            |                   | ✓                            | ✓                                                    | ✓                                       |
| One-to-One Mapping                               | ✓                        |                                            | ✓                 | ✓                            | ✓                                                    | ✓                                       |
| Variable One-to-One Mapping                      | ✓                        |                                            | ✓                 | ✓                            | ✓                                                    | ✓                                       |
| One-to-Many Mapping                              | ✓                        | ✓                                          | ✓                 | ✓                            | ✓                                                    | ✓                                       |
| Many-to-Many Mapping                             | ✓                        | ✓                                          | ✓                 | ✓                            | ✓                                                    | ✓                                       |
| Aggregate Collection Mapping                     | ✓                        | ✓                                          |                   |                              |                                                      | ✓                                       |
| Direct Collection Mapping                        |                          | ✓                                          |                   | ✓                            | ✓                                                    | ✓                                       |
| Direct Map Mapping                               |                          | ✓                                          |                   | ✓                            | ✓                                                    | ✓                                       |
| Object-Relational Data Type Mappings             |                          |                                            |                   |                              |                                                      |                                         |
| Object-Relational Data Type Reference Mapping    | ✓                        | ✓                                          | ✓                 |                              |                                                      | ✓                                       |
| Object-Relational Data Type Nested Table Mapping | ✓                        | ✓                                          | ✓                 |                              |                                                      | ✓                                       |
| EIS Mappings                                     |                          |                                            |                   |                              |                                                      |                                         |
| EIS One-to-One Mapping                           | ✓                        | ✓                                          | ✓                 | ✓                            | ✓                                                    | ✓                                       |
| EIS One-to-Many Mapping                          | ✓                        | ✓                                          | ✓                 | ✓                            | ✓                                                    | ✓                                       |
| XML Mappings                                     |                          |                                            |                   |                              |                                                      |                                         |
| XML Transformation Mapping                       | ✓                        | ✓                                          |                   |                              |                                                      | ✓                                       |

In general, Oracle recommends that you use value holder indirection for one-to-one mappings and transparent indirect container indirection for collection mappings. Enable indirection for transformation mappings if the execution of the transformation is a resource-intensive task (such as accessing a database, in a relational project).

When using indirection with EJB, the version of EJB and application server you use affects how indirection is configured and what types of indirection are applicable.

When using indirection with an object that your application serializes, you must consider the effect of any untriggered indirection objects at deserialization time.

For JPA entities or POJO classes that you configure for weaving, TopLink weaves value holder indirection for one-to-one mappings. If you want TopLink to weave change tracking and your application includes collection mappings (one-to-many or many-to-many), then you must configure all collection mappings to use transparent indirect container indirection only (you may not configure your collection mappings to use eager loading nor value holder indirection).

For more information, see the following:

- [Section 17.2.4, "Indirection \(Lazy Loading\)"](#)

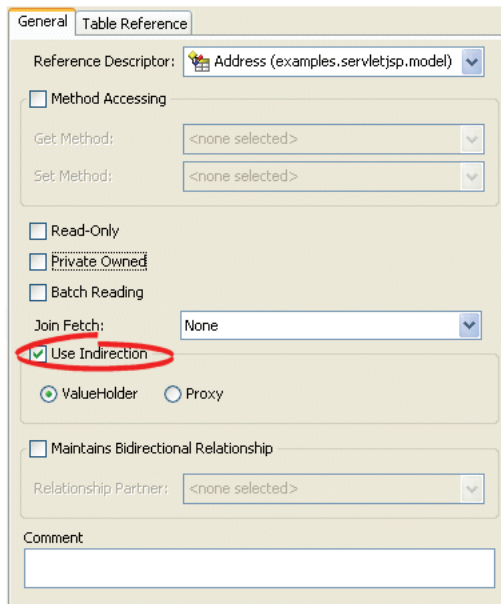
- Section 17.2.4.1, "Value Holder Indirection"
- Section 17.2.4.2, "Transparent Indirect Container Indirection"
- Section 17.2.4.6, "Indirection and EJB 2.n CMP"
- Section 17.2.4.7, "Indirection, Serialization, and Detachment"
- Section 2.10, "Using Weaving"
- "Using EclipseLink JPA Weaving" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#Using\\_EclipseLink\\_JPA\\_Weaving](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#Using_EclipseLink_JPA_Weaving)

### 121.3.1 How to Configure Indirection Using TopLink Workbench

To complete the indirection options on a mapping's **General** tab use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

**Figure 121–2 General Tab, Indirection Options**



Use the following information to complete the **Indirection** fields on the tab:

| Field                  | Description                                                                                                                                           |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Use Indirection</b> | Specify if this mapping uses indirection.                                                                                                             |
| <b>ValueHolder</b>     | Specify that the mapping uses <b>Value Holder</b> indirection. See <a href="#">Section 17.2.4.1, "Value Holder Indirection"</a> for more information. |
| <b>Proxy</b>           | Specify that the mapping uses <b>Proxy</b> indirection. See <a href="#">Section 17.2.4.3, "Proxy Indirection"</a> for more information.               |

## 121.3.2 How to Configure Indirection Using Java

When creating mappings through the Java API, all foreign reference mappings default to using value-holder indirection and all transformation mappings default to not using indirection.

To disable indirection use `ForeignReferenceMapping` method `dontUseIndirection`.

To enable value holder indirection, use `ForeignReferenceMapping` method `useBasicIndirection`.

To enable transparent container indirection, use one of the following `CollectionMapping` methods:

- `useTransparentCollection`
- `useTransparentList`
- `useTransparentMap`
- `useTransparentSet`

To enable proxy indirection, use `ObjectReferenceMapping` method `useProxyIndirection`.

This section provides additional information on the following:

- [Configuring Value Holder Indirection](#)
- [Configuring Value Holder Indirection with Method Accessing](#)
- [Configuring Value Holder Indirection with JPA](#)
- [Configuring IndirectContainer Indirection](#)
- [Configuring Proxy Indirection](#)

### 121.3.2.1 Configuring Value Holder Indirection

Instances of `oracle.toplink.mappings.ForeignReferenceMapping` and `oracle.toplink.mappings.foundation.AbstractTransformationMapping` provide the `useBasicIndirection` method to configure a mapping to an attribute that you code with an `oracle.toplink.indirection.ValueHolderInterface` between it and the real object.

If the attribute is of a `Collection` type (such as a `Vector`), then you can either use an `IndirectContainer` (see [Section 121.3.2.4, "Configuring IndirectContainer Indirection"](#)) or define the `ValueHolder` in the constructor as follows:

```
addresses = new ValueHolder(new Vector());
```

[Example 121–2](#) illustrates the `Employee` class using `ValueHolder` indirection. The class definition conceals the use of `ValueHolder` within the existing getter and setter methods.

#### **Example 121–2 Class Using ValueHolder Indirection**

```
public class Employee {

    protected ValueHolderInterface address;

    // Initialize ValueHolders in constructor
    public Employee() {
        address = new ValueHolder();
    }
}
```

```

    public Address getAddress() {
        return (Address) this.addressHolder.getValue();
    }

    public void setAddress(Address address) {
        this.addressHolder.setValue(address);
    }
}

```

[Example 121–3](#) shows how to configure a one-to-one mapping to the address attribute.

### **Example 121–3 Mapping Using ValueHolder Indirection**

```

OneToOneMapping mapping = new OneToOneMapping();
mapping.useBasicIndirection();
mapping.setReferenceClass(Employee.class);
mapping.setAttributeName("address");

```

The application uses `Employee` methods `getAddress` and `setAddress` to access the `Address` object. Because basic indirection is enabled, `TopLink` expects the persistent fields to be of type `ValueHolderInterface`.

### **121.3.2.2 Configuring Value Holder Indirection with Method Accessing**

If you are using `ValueHolder` indirection with method accessing (see [Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"](#)), in addition to changing your attributes types in your Java code to `ValueHolderInterface`, you must also provide `TopLink` with two pairs of getter and setter methods:

- getter and setter of the *indirection* object that are registered with the mapping and used only by `TopLink`. They include a `get` method that returns an instance that conforms to `ValueHolderInterface`, and a `set` method that accepts one argument that conforms to the same interface;
- getter and setter of the actual attribute value used by the application.

[Example 121–4](#) illustrates the `Employee` class using `ValueHolder` indirection with method access. The class definition is modified so that the `address` attribute of `Employee` is a `ValueHolderInterface` instead of an `Address`, and appropriate getter and setter methods are supplied.

### **Example 121–4 Class Using ValueHolder Indirection with Method Accessing**

```

public class Employee {

    protected ValueHolderInterface address;

    // Initialize ValueHolders in constructor
    public Employee() {
        address = new ValueHolder();
    }

    // getter and setter registered with the mapping and used only by TopLink
    public ValueHolderInterface getAddressHolder() {
        return address;
    }
    public void setAddressHolder(ValueHolderInterface holder) {
        address = holder;
    }

    // getter and setter methods used by the application to access the attribute
    public Address getAddress() {
        return (Address) address.getValue();
    }
}

```

```

    }

    public void setAddress(Address theAddress) {
        address.setValue(theAddress);
    }
}

```

[Example 121–5](#) shows how to configure a one-to-one mapping to the address attribute.

#### **Example 121–5 Mapping Using ValueHolder Indirection with Method Accessing**

```

OneToOneMapping mapping = new OneToOneMapping();
mapping.useBasicIndirection();
mapping.setReferenceClass(Employee.class);
mapping.setAttributeName("address");
mapping.setGetMethodName("getAddressHolder");
mapping.setSetMethodName("setAddressHolder");

```

The application uses `Employee` methods `getAddress` and `setAddress` to access the `Address` object. Because basic indirection is enabled, `TopLink` uses `Employee` methods `getAddressHolder` and `setAddressHolder` methods when performing persistence operations on instances of `Employee`.

### 121.3.2.3 Configuring Value Holder Indirection with JPA

When using indirection with JPA, if your application serializes any indirection-enabled (lazily loaded) entity (see [Section 17.2.4.7, "Indirection, Serialization, and Detachment"](#)), then, to preserve untriggered indirection objects on deserialization, configure your client to use `TopLink` agent, as follows:

1. Include the following JAR files (from `<TOPLINK_HOME>\jlib`) in your client classpath:
  - `toplink.jar`
  - `<your-application-persistence>.jar`
2. Add the following argument to the Java command line you use to start your client:

```
-javaagent:toplink.jar
```

You can also use static weaving (see "How to Configure Static Weaving for JPA Entities" section of *EclipseLink Developer's Guide* at [http://wiki.eclipse.org/Using\\_EclipseLink\\_JPA\\_Extensions\\_%28ELUG%29#How\\_to\\_Configure\\_Static\\_Weaving\\_for\\_JPA\\_Entities](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_%28ELUG%29#How_to_Configure_Static_Weaving_for_JPA_Entities)). This will provide you with better error messages and will resolve merging issues.

---

**Note:** The use of static weaving will not affect serialization as it functions without static weaving enabled.

---

### 121.3.2.4 Configuring IndirectContainer Indirection

Instances of `oracle.toplink.mappings.ForeignReferenceMapping` and `oracle.toplink.mappings.foundation.AbstractTransformationMapping` provide the `useContainerIndirection` method to configure a mapping to an attribute that you code with an `oracle.toplink.indirection.IndirectContainer` between it and the real object.

Using an `IndirectContainer`, a `java.util.Collection` class can act as a `TopLink` indirection object: the `Collection` will only read its contents from the database when necessary (typically, when a `Collection` accessor is invoked).

Without an `IndirectContainer`, all members of the `Collection` must be retrieved when the `Collection` attribute is accessed.

[Example 121–6](#) illustrates the `Employee` class using `IndirectContainer` indirection with method access. Note that the fact of using indirection is transparent.

**Example 121–6 Class Using IndirectContainer Indirection**

```
public class Employee {

    protected List addresses;

    public Employee() {
        this.addresses = new ArrayList();
    }

    public List getAddresses() {
        return this.addresses;
    }

    public void setAddresses(List addresses) {
        this.addresses = addresses;
    }
}
```

[Example 121–7](#) shows how to configure a one-to-one mapping to the `addresses` attribute.

**Example 121–7 Mapping Using IndirectContainer Indirection**

```
OneToOneMapping mapping = new OneToOneMapping();
mapping.useBasicIndirection();
mapping.setReferenceClass(Employee.class);
mapping.setAttributeName("addresses");
mapping.setGetMethodName("getAddresses");
mapping.setSetMethodName("setAddresses");
```

### 121.3.2.5 Configuring Proxy Indirection

[Example 121–8](#) illustrates an `Employee` to `Address` one-to-one relationship.

**Example 121–8 Classes Using Proxy Indirection**

```
public interface Employee {

    public String getName();
    public Address getAddress();
    public void setName(String value);
    public void setAddress(Address value);
    . . .
}

public class EmployeeImpl implements Employee {

    public String name;
    public Address address;
    . . .
    public Address getAddress() {
        return this.address;
    }

    public void setAddress(Address value) {
        this.address = value;
    }
}
```



```

public interface Address {

    public String getStreet();
    public void setStreet(String value);
    . . .
}

public class AddressImpl implements Address {

    public String street;
    . . .
}

```

In [Example 121–8](#), both the `EmployeeImpl` and the `AddressImpl` classes implement public interfaces (`Employee` and `Address` respectively). Therefore, because the `AddressImpl` class is the target of the one-to-one relationship, it is the only class that must implement an interface. However, if the `EmployeeImpl` is ever to be the target of another one-to-one relationship using transparent indirection, it must also implement an interface, as shown in the following example:

```

Employee emp = (Employee)session.readObject(Employee.class);
System.out.println(emp.toString());
System.out.println(emp.getAddress().toString());
// Would print:
[Employee] John Smith
{ IndirectProxy: not instantiated }

String street = emp.getAddress().getStreet();
// Triggers database read to get Address information
System.out.println(emp.toString());
System.out.println(emp.getAddress().toString());
// Would print:
[Employee] John Smith
{ [Address] 123 Main St. }

```

Using proxy indirection does not change how you instantiate your own domain objects for an insert operation. You still use the following code:

```

Employee emp = new EmployeeImpl("John Smith");
Address add = new AddressImpl("123 Main St.");
emp.setAddress(add);

```

[Example 121–9](#) illustrates an `Employee` to `Address` one-to-one relationship mapping.

#### **Example 121–9 Mapping Using Proxy Indirection**

```

OneToOneMapping mapping = new OneToOneMapping();
mapping.useProxyIndirection();
mapping.setReferenceClass(Employee.class);
mapping.setAttributeName("address");
mapping.setGetMethodName("getAddress");
mapping.setSetMethodName("setAddress");

```

## 121.4 Configuring XPath

TopLink uses XPath statements to map the attributes of a Java object to locations in an XML document. When you create an XML mapping or EIS mapping using XML records, you can specify the XPath based on any of the following:

- Name
- Position
- Path and name

[Table 121–5](#) summarizes which mappings support this option.

**Table 121–5 Mapping Support for XPath**

| Mapping                                   | How to Use Oracle JDeveloper | How to Configure XPath Using TopLink Workbench | How to Use Java |
|-------------------------------------------|------------------------------|------------------------------------------------|-----------------|
| EIS Mappings <sup>1</sup>                 |                              |                                                |                 |
| EIS Direct Mapping                        | ✓                            | ✓                                              | ✓               |
| EIS Composite Direct Collection Mapping   | ✓                            | ✓                                              | ✓               |
| EIS Composite Object Mapping <sup>2</sup> | ✓                            | ✓                                              | ✓               |
| EIS Composite Collection Mapping          | ✓                            | ✓                                              | ✓               |
| XML Mappings                              |                              |                                                |                 |
| XML Direct Mapping                        | ✓                            | ✓                                              | ✓               |
| XML Composite Direct Collection Mapping   | ✓                            | ✓                                              | ✓               |
| XML Composite Object Mapping <sup>2</sup> | ✓                            | ✓                                              | ✓               |
| XML Composite Collection Mapping          | ✓                            | ✓                                              | ✓               |
| XML Any Object Mapping                    | ✓                            | ✓                                              | ✓               |
| XML Any Collection Mapping                | ✓                            | ✓                                              | ✓               |
| XML Binary Data Mapping                   | ✓                            | ✓                                              | ✓               |
| XML Binary Data Collection Mapping        | ✓                            | ✓                                              | ✓               |
| XML Fragment Mapping                      | ✓                            | ✓                                              | ✓               |
| XML Fragment Collection Mapping           | ✓                            | ✓                                              | ✓               |
| XML Choice Collection Mapping             | ✓                            | ✓                                              | ✓               |
| XML Any Attribute Mapping                 | ✓                            | ✓                                              | ✓               |

<sup>1</sup> When used with XML records only (see Section 76.4, "Configuring Record Format").

<sup>2</sup> Supports the self XPath (".") so that the TopLink runtime performs all read and write operations in the parent's element and not an element nested within it (see Section 17.2.9, "Mappings and the jaxb:class Customization").

Before you can select an XPath for a mapping, you must associate the descriptor with a schema context (see Section 76.2, "Configuring Schema Context for an EIS Descriptor" or Section 52.2, "Configuring Schema Context for an XML Descriptor").

For more information, see Section 17.2.7, "Mappings and XPath".

### 121.4.1 How to Configure XPath Using TopLink Workbench

Use this table to select the XPath for an XML mapping or EIS mapping using XML records:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. If necessary, click the **General** tab. The General tab appears.

**Figure 121–3 General Tab, XPath Options**

The screenshot shows the 'General' tab of a configuration dialog. At the top, there are two tabs: 'General' and 'Converter'. Under the 'XML Field' section, there is a text box labeled 'XPath:' which is circled in red, and a 'Browse...' button to its right. Below this is a checkbox labeled 'Use XML Schema "type" attribute'. The 'Method Accessing' section has a checkbox and two dropdown menus for 'Get Method:' and 'Set Method:', both currently set to '<none selected>'. The 'Default Null Value' section has a checkbox, a 'Type:' dropdown menu set to '<none selected>', and a 'Value:' text box. At the bottom, there is a 'Read-Only' checkbox and a 'Comment' text box.

**Figure 121–4 XPath Options for Composite Object Mappings**

The screenshot shows a configuration dialog for composite object mappings. Under the 'XML Field' section, there are two radio buttons: 'Specify XPath:' (which is selected and circled in red) and 'Aggregate into parent element' (also circled in red). To the right of the 'Specify XPath:' radio button is a text box and a 'Browse...' button. Below this is a 'Reference Descriptor:' dropdown menu set to '<none select...'. The 'Method Accessing' section has a checkbox and two dropdown menus for 'Get Method:' and 'Set Method:', both set to '<none select...'. The 'Read-Only' section has a checkbox. At the bottom, there is a 'Comment' text box.

Click **Browse** and select the XPath to map to this attribute (see [Section 121.4.1.1, "Choosing the XPath"](#)).

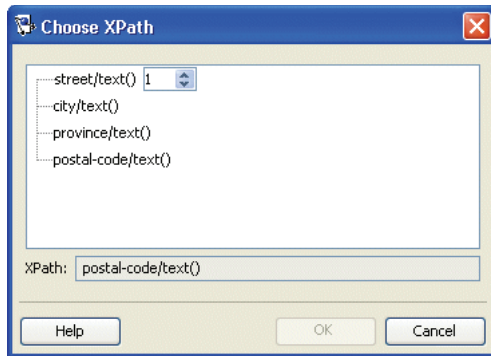
For an EIS composite object mapping using XML records or an XML composite object mapping, you can choose one of the following:

- **Specify XPath:** select the XPath to map to this attribute (see [Section 121.4.1.1, "Choosing the XPath"](#)).
- **Aggregate into parent element:** select the self XPath (".") (see [Section 17.2.7.4, "Self XPath"](#)) so that the TopLink runtime performs all read and write operations in the parent's element, and not an element nested within it (see [Section 17.2.9, "Mappings and the jaxb:class Customization"](#)).

### 121.4.1.1 Choosing the XPath

From the Choose XPath dialog box, select the XPath and click **OK**. TopLink Workbench builds the complete XPath name.

**Figure 121–5 Choose XPath Dialog Box**



## 121.5 Configuring a Default Null Value at the Mapping Level

A default null value is the Java Object type and value that TopLink uses instead of `null` when TopLink reads a null value from a data source.

When you configure a default null value at the mapping level, TopLink uses it to translate in the following two directions:

- When TopLink reads `null` from the data source, it converts this `null` to the specified type and value.
- When TopLink writes or queries to the data source, it converts the specified type and value back to `null`.

Table 121–6 summarizes which mappings support this option.

**Table 121–6 Mapping Support for Default Null Values**

| Mapping                            | How to Use Oracle JDeveloper | How to Configure a Default Null Value at the Mapping Level Using TopLink Workbench | How to Configure a Default Null Value at the Mapping Level Using Java |
|------------------------------------|------------------------------|------------------------------------------------------------------------------------|-----------------------------------------------------------------------|
| Relational Mappings                |                              |                                                                                    |                                                                       |
| Direct-to-Field Mapping            | ✓                            | ✓                                                                                  | ✓                                                                     |
| Direct-to-XMLType Mapping          | ✓                            | ✓                                                                                  | ✓                                                                     |
| EIS Mappings                       |                              |                                                                                    |                                                                       |
| EIS Direct Mapping                 | ✓                            | ✓                                                                                  | ✓                                                                     |
| XML Mappings                       |                              |                                                                                    |                                                                       |
| XML Direct Mapping                 | ✓                            | ✓                                                                                  | ✓                                                                     |
| XML Binary Data Mapping            | ✓                            | ✓                                                                                  | ✓                                                                     |
| XML Binary Data Collection Mapping | ✓                            | ✓                                                                                  | ✓                                                                     |
| XML Fragment Mapping               | ✓                            | ✓                                                                                  | ✓                                                                     |
| XML Fragment Collection Mapping    | ✓                            | ✓                                                                                  | ✓                                                                     |

---

**Note:** A default null value must be an `Object`. To specify a primitive value (such as `int`), you must use the corresponding `Object` wrapper (such as `Integer`).

---

You can also use `TopLink` to set a default null value for all mappings used in a session (see [Section 97.6, "Configuring a Default Null Value at the Login Level"](#)).

## 121.5.1 How to Configure a Default Null Value at the Mapping Level Using TopLink Workbench

To configure a default null value for a mapping, use this procedure.

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

**Figure 121–6 General Tab, Default Null Value Options**

The screenshot shows the 'General' tab in the TopLink Workbench. The 'Database Field' is set to 'ADDRESS.CITY (VARCHAR2)'. There are checkboxes for 'Method Accessing', 'Default Null Value', and 'Read-Only'. The 'Default Null Value' checkbox is highlighted with a red circle. Below it, there are fields for 'Type' and 'Value'. A 'Comment' field is also present at the bottom.

Use the following information to complete the **Default Null Value** fields on the tab:

| Field                     | Description                                                                                                                                                                                 |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Default Null Value</b> | Specify if this mapping contains a default value in the event that the data source is <code>null</code> . If selected, you must enter both the <b>Type</b> and <b>Value</b> of the default. |
| <b>Type</b>               | Select the Java type of the default value.                                                                                                                                                  |
| <b>Value</b>              | Enter the default value.                                                                                                                                                                    |

## 121.5.2 How to Configure a Default Null Value at the Mapping Level Using Java

To configure a mapping null value using Java API, use the `AbstractDirectMapping` method `setNullValue`.

For example:

```
// Defaults a null salary to 0
salaryMapping.setNullValue(new Integer(0));
```

## 121.6 Configuring Method or Direct Field Accessing at the Mapping Level

By default, TopLink uses direct access to access public attributes. Alternatively, you can use getter and setter methods to access object attributes when writing the attributes of the object to the database, or reading the attributes of the object from the database. This is known as method access.

Using private, protected or package variable or method access requires you to enable the Java reflect security setting. This is enabled by default in most application servers (see [Section 8.2.3, "How to Set Security Permissions"](#)), but may need to be enabled explicitly in certain JVM configurations. If necessary, use the `java.policy` file to grant `ReflectPermission` to the entire application or the application's code base. For example:

```
grant{
    permission java.lang.reflect.ReflectPermission;
};
```

Oracle recommends using *direct access* whenever possible to improve performance and avoid executing any application-specific behavior while building objects.

[Table 121-7](#) summarizes which mappings support this option.

**Table 121-7 Mapping Support for Method Accessing**

| Mapping                              | How to Use Oracle JDeveloper | How to Configure Method or Direct Field Accessing Using TopLink Workbench | How to Configure Method or Direct Field Accessing Using Java |
|--------------------------------------|------------------------------|---------------------------------------------------------------------------|--------------------------------------------------------------|
| Relational Mappings                  | ✓                            | ✓                                                                         | ✓                                                            |
| Object-Relational Data Type Mappings |                              |                                                                           | ✓                                                            |
| EIS Mappings                         | ✓                            | ✓                                                                         | ✓                                                            |
| XML Mappings                         | ✓                            | ✓                                                                         | ✓                                                            |

For information on configuring method accessing at the project level, see [Section 117.4, "Configuring Method or Direct Field Access at the Project Level"](#).

If you enable change tracking on a property (for example, you decorate method `getPhone` with `@ChangeTracking`) and you access the field (`phone`) directly, note that TopLink does not detect the change. For more information, see [Section 2.4.1.4, "Using Method and Direct Field Access"](#).

### 121.6.1 How to Configure Method or Direct Field Accessing Using TopLink Workbench

To complete the field access method for a mapping, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

**Figure 121–7 General Tab, Method Accessing Options**

The screenshot shows the 'General' tab of a mapping editor. At the top, there are two tabs: 'General' and 'Converter'. Below them, the 'Database Field' is set to 'ADDRESS.CITY (VARCHAR2)'. The 'Method Accessing' checkbox is checked and highlighted with a red circle. Below it, there are two dropdown menus for 'Get Method' and 'Set Method', both currently set to '<none selected>'. Further down, there is a 'Default Null Value' section with a 'Type' dropdown set to '<none selected>' and an empty 'Value' text box. At the bottom, there is a 'Read-Only' checkbox and a 'Comment' text area.

Use the following information to complete the **Method Accessing** fields on this tab:

| Field                   | Description                                                                                                                                                                             |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Method Accessing</b> | Specify if this mapping uses specific accessor methods instead directly accessing public attributes. By default, this option is not selected (that is, the mapping uses direct access). |
| <b>Get Method</b>       | Select a specific <code>get</code> method.                                                                                                                                              |
| <b>Set Method</b>       | Select a specific <code>set</code> method.                                                                                                                                              |

To change the default access type used by all new mappings, use the Defaults tab on the project Editor window. See [Section 117.4, "Configuring Method or Direct Field Access at the Project Level"](#) for more information.

## 121.6.2 How to Configure Method or Direct Field Accessing Using Java

Use the following `DatabaseMapping` methods to configure the user-defined getters and setters that TopLink will use to access the mapped attribute:

For mappings not supported in Oracle JDeveloper and TopLink Workbench, use the `setGetMethodName` and `setSetMethodName` methods to access the attribute through user-defined methods, rather than directly, as follows:

- `setGetMethodName`—set the `String` name of the user-defined method to get the mapped attribute;
- `setSetMethodName`—set the `String` name of the user-defined method to set the mapped attribute.

[Example 121–10](#) shows how to use these methods with a class that has an attribute `phones` and accessor methods `getPhones` and `setPhones` in an object-relational data type mapping.

### **Example 121–10 Configuring Access Method in Java**

```
// Map the phones attribute
phonesMapping.setAttributeName("phones");

// Specify access method
phonesMapping.setGetMethodName("getPhones");
phonesMapping.setSetMethodName("setPhones");
```

## 121.7 Configuring Private or Independent Relationships

In TopLink, object relationships can be either private or independent:

- In a private relationship, the target object is a private component of the source object. The target object cannot exist without the source and is accessible only through the source object. Destroying the source object will also destroy the target object.
- In an independent relationship, the source and target objects are public ones that exist independently. Destroying one object does not necessarily imply the destruction of the other.

---

**Tip:** TopLink automatically manages private relationships. Whenever an object is written to the database, any private objects it owns are also written to the database. When an object is removed from the database, any private objects it owns are also removed. Be aware of this when creating new systems, since it may affect both the behavior and the performance of your application.

---

Table 121–8 summarizes which mappings support this option.

**Table 121–8 Mapping Support for Private or Independent Relationships**

| Mapping                                          | Implicitly Private | Private or Independent | How to Use Oracle JDeveloper | How to Configure Private or Independent Relationships Using TopLink Workbench | How to Configure Private or Independent Relationships Using Java |
|--------------------------------------------------|--------------------|------------------------|------------------------------|-------------------------------------------------------------------------------|------------------------------------------------------------------|
| Relational Mappings                              |                    |                        |                              |                                                                               |                                                                  |
| One-to-One Mapping                               |                    | ✓                      | ✓                            | ✓                                                                             | ✓                                                                |
| Variable One-to-One Mapping                      |                    | ✓                      | ✓                            | ✓                                                                             | ✓                                                                |
| One-to-Many Mapping                              |                    | ✓                      | ✓                            | ✓                                                                             | ✓                                                                |
| Many-to-Many Mapping                             |                    | ✓                      | ✓                            | ✓                                                                             | ✓                                                                |
| Aggregate Collection Mapping                     | ✓                  |                        |                              |                                                                               |                                                                  |
| Direct Collection Mapping                        | ✓                  |                        |                              |                                                                               |                                                                  |
| Direct Map Mapping                               | ✓                  | ✓                      | ✓                            | ✓                                                                             | ✓                                                                |
| Aggregate Object Mapping                         | ✓                  |                        |                              |                                                                               |                                                                  |
| Object-Relational Data Type Mappings             |                    |                        |                              |                                                                               |                                                                  |
| Object-Relational Data Type Structure Mapping    | ✓                  |                        |                              |                                                                               |                                                                  |
| Object-Relational Data Type Reference Mapping    |                    | ✓                      |                              |                                                                               | ✓                                                                |
| Object-Relational Data Type Array Mapping        | ✓                  |                        |                              |                                                                               |                                                                  |
| Object-Relational Data Type Object Array Mapping | ✓                  |                        |                              |                                                                               |                                                                  |
| Object-Relational Data Type Nested Table Mapping |                    | ✓                      |                              |                                                                               | ✓                                                                |
| EIS Mappings                                     |                    |                        |                              |                                                                               |                                                                  |
| EIS Composite Direct Collection Mapping          | ✓                  |                        |                              |                                                                               |                                                                  |
| EIS Composite Object Mapping                     | ✓                  |                        |                              |                                                                               |                                                                  |



**Table 121–8 (Cont.) Mapping Support for Private or Independent Relationships**

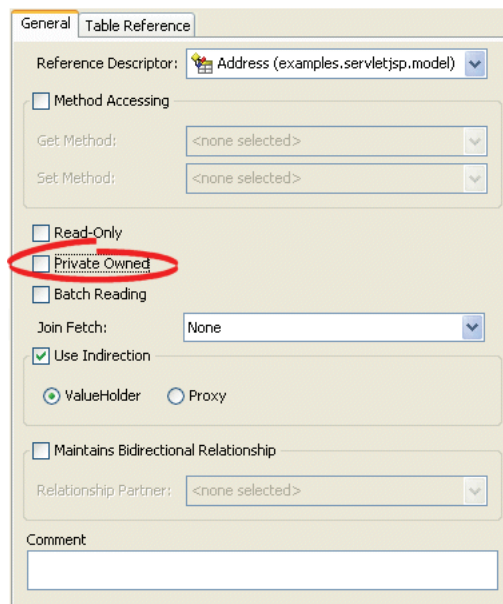
| Mapping                                 | Implicitly Private | Private or Independent | How to Use Oracle JDeveloper | How to Configure Private or Independent Relationships Using TopLink Workbench | How to Configure Private or Independent Relationships Using Java |
|-----------------------------------------|--------------------|------------------------|------------------------------|-------------------------------------------------------------------------------|------------------------------------------------------------------|
| EIS Composite Collection Mapping        | ✓                  |                        |                              |                                                                               |                                                                  |
| EIS One-to-One Mapping                  |                    | ✓                      | ✓                            | ✓                                                                             | ✓                                                                |
| EIS One-to-Many Mapping                 |                    | ✓                      | ✓                            | ✓                                                                             | ✓                                                                |
| XML Mappings                            |                    |                        |                              |                                                                               |                                                                  |
| XML Composite Direct Collection Mapping | ✓                  |                        |                              |                                                                               |                                                                  |
| XML Composite Object Mapping            | ✓                  |                        |                              |                                                                               |                                                                  |
| XML Composite Collection Mapping        | ✓                  |                        |                              |                                                                               |                                                                  |

### 121.7.1 How to Configure Private or Independent Relationships Using TopLink Workbench

To create a privately owned mapping, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

**Figure 121–8 General Tab, Private Owned option**



To create private ownership, select the **Private Owned** option.

## 121.7.2 How to Configure Private or Independent Relationships Using Java

For mappings not supported in Oracle JDeveloper and TopLink Workbench, use the `independentRelationship` (default), `privateOwnedRelationship`, and `setIsPrivateOwned` methods.

[Example 121–10](#) shows how to use these methods with a class that has a privately owned attribute, `phones`, in a mapping.

### **Example 121–11** Configuring Access Method in Java

```
// Map the phones attribute
phonesMapping.setAttributeName("phones");

// Specify as privately owned
phonesMapping.privateOwnedRelationship();
```

## 121.8 Configuring Mapping Comments

You can define a free-form textual comment for each mapping. You can use these comments however you wish: for example, to record important project implementation details such as the purpose or importance of a mapping.

Comments are stored in the Oracle JDeveloper or TopLink Workbench project, in the TopLink deployment XML file. There is no Java API for this feature.

[Table 121–9](#) summarizes which mappings support this option.

**Table 121–9** Mapping Support for Comments

| Mapping                             | How to Use Oracle JDeveloper | How to Configure Mapping Comments Using TopLink Workbench | How to Use Java |
|-------------------------------------|------------------------------|-----------------------------------------------------------|-----------------|
| <a href="#">Relational Mappings</a> |                              | ✓                                                         |                 |
| <a href="#">EIS Mappings</a>        |                              | ✓                                                         |                 |
| <a href="#">XML Mappings</a>        |                              | ✓                                                         |                 |

### 121.8.1 How to Configure Mapping Comments Using TopLink Workbench

To add a comment for a mapping, use this procedure.

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

**Figure 121–9 General Tab, Comment**

The screenshot shows the 'General' tab of a configuration window. At the top, there are two tabs: 'General' and 'Converter'. Below the tabs, the 'Database Field' is set to 'ADDRESS.CITY (VARCHAR2)'. There are three sections, each with a checkbox and a dropdown menu:

- Method Accessing:** The checkbox is unchecked. Below it are two dropdown menus for 'Get Method' and 'Set Method', both set to '<none selected>'. The 'Set Method' dropdown is highlighted with a blue selection bar.
- Default Null Value:** The checkbox is unchecked. Below it are two dropdown menus for 'Type' (set to '<none selected>') and a text input field for 'Value'.
- Read-Only:** The checkbox is unchecked. Below it is a text input field for 'Comment', which is circled in red.

Enter a comment that describes this mapping.

## 121.9 Configuring a Serialized Object Converter

A serialized object converter can be used to store an arbitrary object or set of objects into a data source binary large object (BLOB) field. It uses the Java serializer so the target must be serializable.

For more information about the serialized object converter, see [Section 17.2.6.1, "Serialized Object Converter"](#).

Table 121–10 summarizes which mappings support this option.

**Table 121–10 Mapping Support for Serialized Object Converter**

| Mapping                                   | How to Use Oracle JDeveloper | How to Configure a Serialized Object Converter Using TopLink Workbench | How to Configure a Serialized Object Converter Using Java |
|-------------------------------------------|------------------------------|------------------------------------------------------------------------|-----------------------------------------------------------|
| Relational Mappings                       |                              |                                                                        |                                                           |
| Direct-to-Field Mapping                   | ✓                            | ✓                                                                      | ✓                                                         |
| Direct Collection Mapping                 | ✓                            | ✓                                                                      | ✓                                                         |
| Direct Map Mapping                        | ✓                            | ✓                                                                      | ✓                                                         |
| Object-Relational Data Type Mappings      |                              |                                                                        |                                                           |
| Object-Relational Data Type Array Mapping |                              |                                                                        | ✓                                                         |
| EIS Mappings                              |                              |                                                                        |                                                           |
| EIS Direct Mapping                        | ✓                            | ✓                                                                      | ✓                                                         |
| EIS Composite Direct Collection Mapping   | ✓                            | ✓                                                                      | ✓                                                         |
| XML Mappings                              |                              |                                                                        |                                                           |
| XML Direct Mapping                        | ✓                            | ✓                                                                      | ✓                                                         |
| XML Composite Direct Collection Mapping   | ✓                            | ✓                                                                      | ✓                                                         |

**Table 121–10 (Cont.) Mapping Support for Serialized Object Converter**

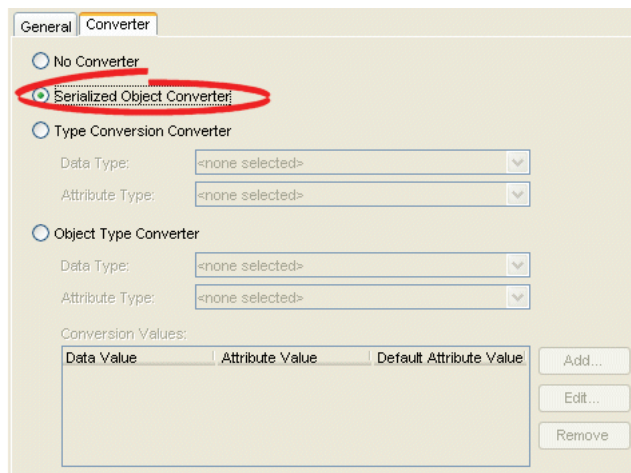
| Mapping                            | How to Use Oracle JDeveloper | How to Configure a Serialized Object Converter Using TopLink Workbench | How to Configure a Serialized Object Converter Using Java |
|------------------------------------|------------------------------|------------------------------------------------------------------------|-----------------------------------------------------------|
| XML Binary Data Mapping            | ✓                            | ✓                                                                      | ✓                                                         |
| XML Binary Data Collection Mapping | ✓                            | ✓                                                                      | ✓                                                         |
| XML Fragment Mapping               | ✓                            | ✓                                                                      | ✓                                                         |
| XML Fragment Collection Mapping    | ✓                            | ✓                                                                      | ✓                                                         |

### 121.9.1 How to Configure a Serialized Object Converter Using TopLink Workbench

To create an serialized object direct mapping, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **Converter** tab. The Converter tab appears.

**Figure 121–10 Converter Tab, Serialized Object Converter Option**



To specify a serialized object converter, select the **Serialized Object Converter** option.

### 121.9.2 How to Configure a Serialized Object Converter Using Java

You can set an `oracle.toplink.converters.SerializedObjectConverter` on any instance of `oracle.toplink.mappings.foundation.AbstractCompositeDirectCollectionMapping` or its subclasses using the `AbstractCompositeDirectCollectionMapping` method `setValueConverter`, as [Example 121–12](#) shows.

**Example 121–12 Configuring a SerializedObjectConverter**

```
// Create SerializedObjectConverter instance
SerializedObjectConverter serializedObjectConvter = new SerializedObjectConverter();
```

```
// Set SerializedObjectConverter on ArrayMapping
ArrayMapping arrayMapping = new ArrayMapping();
arrayMapping.setValueConverter(serializedObjectConvter);
arrayMapping.setAttributeName("responsibilities");
arrayMapping.setStructureName("Responsibilities_t");
arrayMapping.setFieldName("RESPONSIBILITIES");
orDescriptor.addMapping(arrayMapping);
```

You can also set a `SerializedObjectConverter` on any instance of `oracle.toplink.mappings.foundation.AbstractDirectMapping` or its subclasses using the `AbstractDirectMapping` method `setConverter`.

## 121.10 Configuring a Type Conversion Converter

A type conversion converter is used to explicitly map a data source type to a Java type.

For more information about the type conversion converter, see [Section 17.2.6.2, "Type Conversion Converter"](#).

Table 121–11 summarizes which mappings support this option.

**Table 121–11 Mapping Support for Type Conversion Converter**

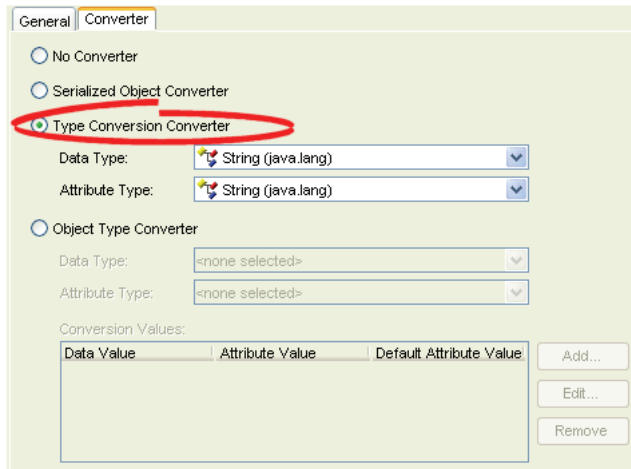
| Mapping                                   | How to Use Oracle JDeveloper | How to Configure a Type Conversion Converter Using TopLink Workbench | How to Configure a Type Conversion Converter Using Java |
|-------------------------------------------|------------------------------|----------------------------------------------------------------------|---------------------------------------------------------|
| Relational Mappings                       |                              |                                                                      |                                                         |
| Direct-to-Field Mapping                   | ✓                            | ✓                                                                    | ✓                                                       |
| Object-Relational Data Type Mappings      |                              |                                                                      |                                                         |
| Object-Relational Data Type Array Mapping |                              |                                                                      | ✓                                                       |
| EIS Mappings                              |                              |                                                                      |                                                         |
| EIS Direct Mapping                        | ✓                            | ✓                                                                    | ✓                                                       |
| EIS Composite Direct Collection Mapping   | ✓                            | ✓                                                                    | ✓                                                       |
| XML Mappings                              |                              |                                                                      |                                                         |
| XML Direct Mapping                        | ✓                            | ✓                                                                    | ✓                                                       |
| XML Composite Direct Collection Mapping   | ✓                            | ✓                                                                    | ✓                                                       |
| XML Binary Data Mapping                   | ✓                            | ✓                                                                    | ✓                                                       |
| XML Binary Data Collection Mapping        | ✓                            | ✓                                                                    | ✓                                                       |
| XML Fragment Mapping                      | ✓                            | ✓                                                                    | ✓                                                       |
| XML Fragment Collection Mapping           | ✓                            | ✓                                                                    | ✓                                                       |

### 121.10.1 How to Configure a Type Conversion Converter Using TopLink Workbench

To create an type conversion direct mapping, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **Converter** tab. The Converter tab appears.
3. Select the **Type Conversion Converter** option.

**Figure 121–11 Converter Tab, Type Conversion Converter Option**



Use the following information to complete the Type Conversion Converter fields on the Converter tab:

| Field          | Description                                              |
|----------------|----------------------------------------------------------|
| Data Type      | Select the Java type of the data in the data source.     |
| Attribute Type | Select the Java type of the attribute in the Java class. |

## 121.10.2 How to Configure a Type Conversion Converter Using Java

You can set an `oracle.toplink.converters.TypeConversionConverter` on any instance of `oracle.toplink.mappings.foundation.AbstractCompositeDirectCollectionMapping` or its subclasses using the `AbstractCompositeDirectCollectionMapping` method `setValueConverter`, as [Example 121–13](#) shows.

### Example 121–13 Configuring a TypeConversionConverter

```
// Create TypeConversionConverter instance
TypeConversionConverter typeConversionConverter = new TypeConversionConverter();
typeConversionConverter.setDataClass(java.util.Calendar.class);
typeConversionConverter.setObjectClass(java.sql.Date.class);
```

```
// Set TypeConversionConverter on ArrayMapping
ArrayMapping arrayMapping = new ArrayMapping();
arrayMapping.setValueConverter(typeConversionConverter);
arrayMapping.setAttributeName("date");
arrayMapping.setStructureName("Date_t");
arrayMapping.setFieldName("DATE");
orDescriptor.addMapping(arrayMapping);
```

You can also set a `TypeConversionConverter` on any instance of `oracle.toplink.mappings.foundation.AbstractDirectMapping` or its subclasses using the `AbstractDirectMapping` method `setConverter`.

Configure the `TypeConversionConverter` instance using the following API:

- `setDataClass(java.lang.Class dataClass)`—to specify the data type class.

- `setObjectClass(java.lang.Class objectClass)`—to specify the object type class.

## 121.11 Configuring an Object Type Converter

An object type converter is used to match a fixed number of data source data values to Java object values. It can be used when the values in the data source and in Java differ.

For more information about the object type converter, see [Section 17.2.6.3, "Object Type Converter"](#).

Table 121–12 summarizes which mappings support this option.

**Table 121–12 Mapping Support for Object Type Converter**

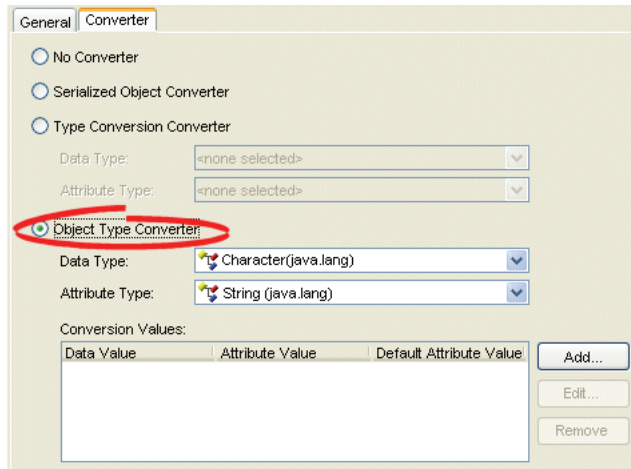
| Mapping                                   | How to Use Oracle JDeveloper | How to Configure an Object Type Converter Using TopLink Workbench | How to Configure an Object Type Converter Using Java |
|-------------------------------------------|------------------------------|-------------------------------------------------------------------|------------------------------------------------------|
| Relational Mappings                       | ✓                            | ✓                                                                 | ✓                                                    |
| Object-Relational Data Type Mappings      |                              |                                                                   |                                                      |
| Object-Relational Data Type Array Mapping |                              |                                                                   | ✓                                                    |
| EIS Mappings                              |                              |                                                                   |                                                      |
| EIS Direct Mapping                        | ✓                            | ✓                                                                 | ✓                                                    |
| EIS Composite Direct Collection Mapping   | ✓                            | ✓                                                                 | ✓                                                    |
| XML Mappings                              |                              |                                                                   |                                                      |
| XML Direct Mapping                        | ✓                            | ✓                                                                 | ✓                                                    |
| XML Composite Direct Collection Mapping   | ✓                            | ✓                                                                 | ✓                                                    |
| XML Binary Data Mapping                   | ✓                            | ✓                                                                 | ✓                                                    |
| XML Binary Data Collection Mapping        | ✓                            | ✓                                                                 | ✓                                                    |
| XML Fragment Mapping                      | ✓                            | ✓                                                                 | ✓                                                    |
| XML Fragment Collection Mapping           | ✓                            | ✓                                                                 | ✓                                                    |

### 121.11.1 How to Configure an Object Type Converter Using TopLink Workbench

To add an object type converter to a direct mapping, use this procedure:

1. Select the mapping in the **Navigator**. Its properties appear in the Editor.
2. Click the **Converter** tab. The Converter tab appears.

**Figure 121–12 Converter Tab, Object Type Converter**



Use the following fields on the mapping’s **Converter** tab to specify the object type converter options:

| Field                          | Description                                                                                                                                                                                                                                                                                                                                                         |
|--------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Data Type</b>               | Select the Java type of the data in the data source.                                                                                                                                                                                                                                                                                                                |
| <b>Attribute Type</b>          | Select the Java type of the attribute in the Java class.                                                                                                                                                                                                                                                                                                            |
| <b>Conversion Values</b>       | Click <b>Add</b> to add a new conversion value. Click <b>Edit</b> to modify an existing conversion value. Click <b>Remove</b> to delete an existing conversion value.<br><br>Use to specify the selected value as the default value. If TopLink retrieves a value from the database that is not mapped as a valid Conversion Value, the default value will be used. |
| <b>Data Value</b>              | Specify the value of the attribute in the data source.                                                                                                                                                                                                                                                                                                              |
| <b>Attribute Value</b>         | Specify the value of the attribute in the Java class                                                                                                                                                                                                                                                                                                                |
| <b>Default Attribute Value</b> | Specify whether or not to use the selected value as the default value. If TopLink retrieves a value from the database that is not mapped as a valid Conversion Value, the default value will be used.                                                                                                                                                               |

### 121.11.2 How to Configure an Object Type Converter Using Java

You can set an `oracle.toplink.converters.ObjectTypeConverter` on any instance of `oracle.toplink.mappings.foundation.AbstractCompositeDirectCollectionMapping` using `AbstractCompositeDirectCollectionMapping` method `setValueConverter`.

You can also set an `ObjectTypeConverter` on any instance of `oracle.toplink.mappings.foundation.AbstractDirectMapping` or its subclasses using the `AbstractDirectMapping` method `setConverter`, as [Example 121–14](#) shows.

**Example 121–14 Configuring an ObjectTypeConverter**

```
// Create ObjectTypeConverter instance
ObjectTypeConverter objectTypeConverter = new ObjectTypeConverter();
objectTypeConverter.addConversionValue("F", "Female");
```



```
// Set ObjectConverter on DirectToFieldMapping
DirectToFieldMapping genderMapping = new DirectToFieldMapping();
genderMapping.setConverter(objectTypeConverter);
genderMapping.setFieldName("F");
genderMapping.setAttributeName("Female");
descriptor.addMapping(genderMapping);
```

Configure the `ObjectConverter` instance using the following API:

- `addConversionValue(java.lang.Object fieldValue, java.lang.Object attributeValue)`—to associate data-type values to object-type values.
- `addToAttributeOnlyConversionValue(java.lang.Object fieldValue, java.lang.Object attributeValue)`—to add one-way conversion values.
- `setDefaultAttributeValue(java.lang.Object defaultAttributeValue)`—to set the default value.

## 121.12 Configuring a Simple Type Translator

The simple type translator allows you to automatically translate an XML element value to an appropriate Java type based on the element's `<type>` attribute, as defined in your XML schema. You can use a simple type translator only when the mapping's XPath goes to an element. You cannot use a simple type translator if the mapping's XPath goes to an attribute.

For more information, see [Section 17.2.6.4, "Simple Type Translator"](#).

Table 121–13 summarizes which mappings support this option.

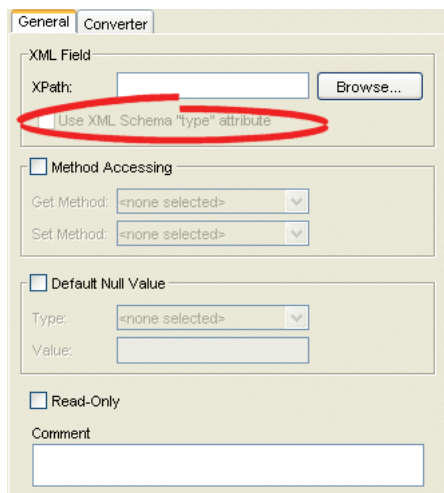
**Table 121–13 Mapping Support for Simple Type Translator**

| Mapping                                 | How to Use Oracle JDeveloper | How to Configure a Simple Type Translator Using TopLink Workbench | How to Configure a Simple Type Translator Using Java |
|-----------------------------------------|------------------------------|-------------------------------------------------------------------|------------------------------------------------------|
| EIS Mappings                            |                              |                                                                   |                                                      |
| EIS Direct Mapping                      | ✓                            | ✓                                                                 | ✓                                                    |
| EIS Composite Direct Collection Mapping | ✓                            | ✓                                                                 | ✓                                                    |
| XML Mappings                            |                              |                                                                   |                                                      |
| XML Direct Mapping                      | ✓                            | ✓                                                                 | ✓                                                    |
| XML Composite Direct Collection Mapping | ✓                            | ✓                                                                 | ✓                                                    |
| XML Binary Data Mapping                 | ✓                            | ✓                                                                 | ✓                                                    |
| XML Binary Data Collection Mapping      | ✓                            | ✓                                                                 | ✓                                                    |

### 121.12.1 How to Configure a Simple Type Translator Using TopLink Workbench

Use this table to qualify elements from the XML schema

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

**Figure 121–13** General Tab, Use XML Schema "type" Attribute Option

Select the **Field Uses XML Schema "type" attribute** field to qualify elements from the XML schema.

## 121.12.2 How to Configure a Simple Type Translator Using Java

To create an XML mapping with a simple type translator with Java code in your IDE, you need the following elements:

- `EISDirectMapping` or `EISCompositeDirectCollectionMapping` or `XMLDirectMapping` or `XMLCompositeDirectCollectionMapping`
- instance of `Converter`
- instance of `TypedElementField`

[Example 121–15](#) shows how to implement your own simple type translator with an `XMLDirectMapping` to override the built-in conversion for writing XML so that `TopLink` writes a Byte array (`ClassConstants.ABYTE`) as a Base64 (`XMLConstants.BASE64_BINARY`) encoded string.

### **Example 121–15** Creating a Type Translation XML Mapping

```
XMLDirectMapping mapping = new XMLDirectMapping();
mapping.setConverter(new SerializedObjectConverter());
TypedElementField field = new TypedElementField("element");
field.getSimpleTypeTranslator().addJavaConversion(
    ClassConstants.ABYTE,
    new QName(XMLConstants.SCHEMA_URL, XMLConstants.BASE64_BINARY));
mapping.setField(field);
```

## 121.13 Configuring a JAXB Typesafe Enumeration Converter

The JAXB typesafe enumeration converter allows you to automatically translate an XML element value to an appropriate typesafe enumeration value as defined in your XML schema.

For more information, see [Section 17.2.10, "Mappings and JAXB Typesafe Enumerations"](#).

[Table 121–14](#) summarizes which mappings support this option.

**Table 121–14 Mapping Support for JAXB Typesafe Enumeration Converter**

| Mapping                                 | How to Use Oracle JDeveloper | How to Use TopLink Workbench | How to Configure a JAXB Typesafe Enumeration Converter Using Java |
|-----------------------------------------|------------------------------|------------------------------|-------------------------------------------------------------------|
| EIS Mappings <sup>1</sup>               |                              |                              |                                                                   |
| EIS Direct Mapping                      |                              |                              | ✓                                                                 |
| EIS Composite Direct Collection Mapping |                              |                              | ✓                                                                 |
| XML Mappings                            |                              |                              |                                                                   |
| XML Direct Mapping                      |                              |                              | ✓                                                                 |
| XML Composite Direct Collection Mapping |                              |                              | ✓                                                                 |
| XML Binary Data Mapping                 |                              |                              | ✓                                                                 |
| XML Binary Data Collection Mapping      |                              |                              | ✓                                                                 |
| XML Fragment Mapping                    |                              |                              | ✓                                                                 |
| XML Fragment Collection Mapping         |                              |                              | ✓                                                                 |

<sup>1</sup> When used with XML records only (see Section 76.4, "Configuring Record Format").

Oracle JDeveloper and TopLink Workbench do not support the `JAXBTypesafeEnumConverter` directly: to configure a mapping with this converter, you must use Java to create an amendment method (see Section 121.13.1, "How to Configure a JAXB Typesafe Enumeration Converter Using Java").

If you create a project and object model using the TopLink JAXB compiler (see Section 48.2, "Creating an XML Project from an XML Schema"), the compiler will create the type safe enumeration class and a class with descriptor amendment methods and register the required amendment methods automatically.

### 121.13.1 How to Configure a JAXB Typesafe Enumeration Converter Using Java

To configure a mapping with a `JAXBTypesafeEnumConverter` in Java, use a descriptor amendment method (see Section 119.35, "Configuring Amendment Methods"). Example 121–16 illustrates an amendment method that configures an `XMLDirectMapping` with a `JAXBTypesafeEnumConverter`. In this example, attribute `_Val` is mapped to a JAXB typesafe enumeration corresponding to typesafe enumeration class `MyTypesafeEnum`.

#### Example 121–16 Creating a JAXB Typesafe Enumeration XML Mapping

```
public class DescriptorAfterLoads {

    public static void amendRootImplDescriptor(ClassDescriptor descriptor) {
        DatabaseMapping _ValMapping = descriptor.getMappingForAttributeName("_Val");
        JAXBTypesafeEnumConverter _ValConverter = new JAXBTypesafeEnumConverter();
        _ValConverter.setEnumClassName("MyTypesafeEnum");
        ((XMLDirectMapping) _ValMapping).setConverter(_ValConverter);
    }
}
```

## 121.14 Configuring Container Policy

Collection mapping container policy specifies the concrete class TopLink should use when reading target objects from the database.

Collection mappings can use any concrete class that implements the `java.util.List`, `java.util.Set`, `java.util.Collection`, or `java.util.Map` interface. You can map object attributes declared as `List`, `Set`, `Collection`, `Map`, or any subinterface of these interfaces, or as a class that implements one of these interfaces.

By default, the TopLink runtime uses the following concrete classes from the `oracle.toplink.indirection` package for each of these container types:

- `List-IndirectList` or `Vector`
- `Set-IndirectSet` or `HashSet`
- `Collection-IndirectList` or `Vector`
- `Map-IndirectMap` or `HashMap`

Alternatively, you can specify in the mapping the concrete container class to be used. When TopLink reads objects from the database that contain an attribute mapped with a collection mapping, the attribute is set with an instance of the concrete class specified. For example, TopLink does not sort in memory. If you want to sort in memory, override the default `Set` type (`IndirectList`) with `java.util.TreeSet` as the concrete collection type. By default, a collection mapping's container class is `java.util.Vector`.

**Note:** If you are using TopLink Workbench and you override the default `Collection` class with a custom `Collection` class of your own, you must put your custom `Collection` class on the TopLink Workbench classpath (see [Section 5.2, "Configuring the TopLink Workbench Environment"](#)).

Table 121–15 summarizes which mappings support this option.

**Table 121–15 Mapping Support for Container Policy**

| Mapping                                          | List | Set | Collection | Map | How to Use Oracle JDeveloper | How to Configure Container Policy Using TopLink Workbench | How to Configure Container Policy Using Java |
|--------------------------------------------------|------|-----|------------|-----|------------------------------|-----------------------------------------------------------|----------------------------------------------|
| Relational Mappings                              |      |     |            |     |                              |                                                           |                                              |
| One-to-Many Mapping                              | ✓    | ✓   | ✓          | ✓   | ✓                            | ✓                                                         | ✓                                            |
| Many-to-Many Mapping                             | ✓    | ✓   | ✓          | ✓   | ✓                            | ✓                                                         | ✓                                            |
| Aggregate Collection Mapping                     | ✓    | ✓   | ✓          | ✓   |                              |                                                           | ✓                                            |
| Direct Collection Mapping                        | ✓    | ✓   | ✓          |     | ✓                            | ✓                                                         | ✓                                            |
| Direct Map Mapping                               |      |     |            | ✓   | ✓                            | ✓                                                         | ✓                                            |
| Object-Relational Data Type Mappings             |      |     |            |     |                              |                                                           |                                              |
| Object-Relational Data Type Array Mapping        | ✓    | ✓   | ✓          | ✓   |                              |                                                           | ✓                                            |
| Object-Relational Data Type Object Array Mapping | ✓    | ✓   | ✓          | ✓   |                              |                                                           | ✓                                            |

**Table 121–15 (Cont.) Mapping Support for Container Policy**

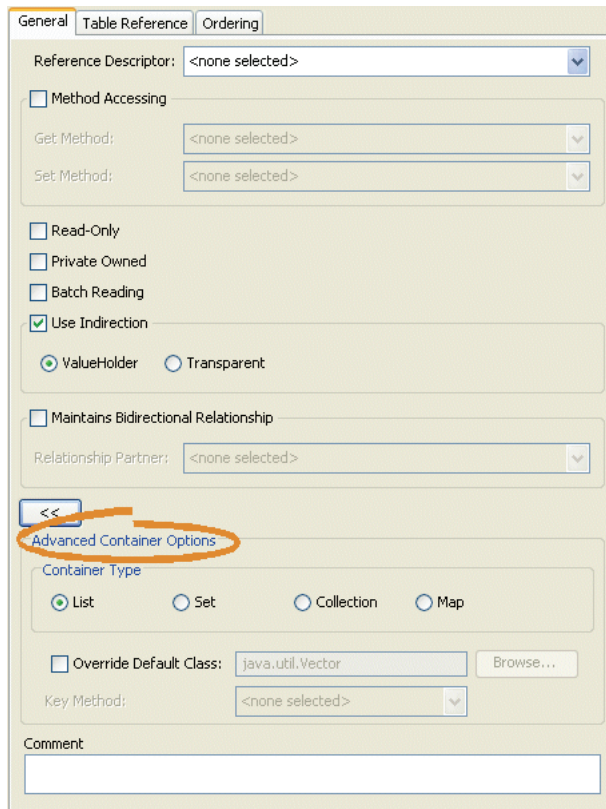
| Mapping                                          | List | Set | Collection | Map | How to Use Oracle JDeveloper | How to Configure Container Policy Using TopLink Workbench | How to Configure Container Policy Using Java |
|--------------------------------------------------|------|-----|------------|-----|------------------------------|-----------------------------------------------------------|----------------------------------------------|
| Object-Relational Data Type Nested Table Mapping | ✓    | ✓   | ✓          | ✓   |                              |                                                           | ✓                                            |
| EIS Mappings                                     |      |     |            |     |                              |                                                           |                                              |
| EIS Composite Direct Collection Mapping          | ✓    | ✓   | ✓          |     | ✓                            | ✓                                                         | ✓                                            |
| EIS Composite Collection Mapping                 | ✓    | ✓   | ✓          | ✓   | ✓                            | ✓                                                         | ✓                                            |
| EIS One-to-Many Mapping                          | ✓    | ✓   | ✓          | ✓   | ✓                            | ✓                                                         | ✓                                            |
| XML Mappings                                     |      |     |            |     |                              |                                                           |                                              |
| XML Composite Direct Collection Mapping          | ✓    | ✓   | ✓          |     | ✓                            | ✓                                                         | ✓                                            |
| XML Composite Collection Mapping                 | ✓    | ✓   | ✓          | ✓   | ✓                            | ✓                                                         | ✓                                            |
| XML Any Collection Mapping                       | ✓    | ✓   | ✓          |     | ✓                            | ✓                                                         | ✓                                            |
| XML Binary Data Collection Mapping               | ✓    | ✓   | ✓          |     | ✓                            | ✓                                                         | ✓                                            |
| XML Collection Reference Mapping                 | ✓    | ✓   | ✓          |     | ✓                            | ✓                                                         | ✓                                            |

### 121.14.1 How to Configure Container Policy Using TopLink Workbench

To specify a mapping's container policy, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.
3. Click the **Advanced** button. The Advanced Container Options appear on the General tab.

**Figure 121–14 General Tab, Advanced Container Options**



Use the following Advanced Container Options fields on the **General** tab to specify the container options:

| Field <sup>1</sup>            | Description                                                                                                                                                                                                                                                                                                                                             |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Container Type</b>         | Specify the type of <code>Collection</code> class to use: <ul style="list-style-type: none"> <li>■ <b>List</b>—use a <code>java.util.List</code></li> <li>■ <b>Set</b>—use a <code>java.util.Set</code></li> <li>■ <b>Collection</b>—use a <code>java.util.Collection</code></li> <li>■ <b>Map</b>—use a <code>java.util.Map</code></li> </ul>          |
| <b>Override Default Class</b> | Specify to use a custom class as the mapping’s container policy. Click <b>Browse</b> to select a different class.<br><br>The container class must implement (directly or indirectly) the <code>java.util.Collection</code> interface.                                                                                                                   |
| <b>Key Method</b>             | If you configure <b>Container Type</b> as <b>Map</b> , use this option to specify the name of the zero argument method whose result, when called on the target object, is used as the key in the <code>Hashtable</code> or <code>Map</code> . This method must return an object that is a valid key in the <code>Hashtable</code> or <code>Map</code> . |

<sup>1</sup> Not all mappings support all options. For more information, see [Table 121–15](#).

## 121.14.2 How to Configure Container Policy Using Java

Classes that implement the `oracle.toplink.mappings.ContainerMapping` interface provide the following methods to set the container policy:

- `useCollectionClass(java.lang.Class concreteClass)`—Configure the mapping to use an instance of the specified `java.util.Collection` container class to hold the target objects.
- `useMapClass(java.lang.Class concreteClass, java.lang.String methodName)`—Configure the mapping to use an instance of the specified `java.util.Map` container class to hold the target objects. The key used to index a value in the `Map` is the value returned by a call to the specified zero-argument method. The method must be implemented by the class (or a superclass) of any value to be inserted into the `Map`.

Classes that extend `oracle.toplink.mappings.CollectionMapping` (which implements the `ContainerMapping` interface) also provide the following methods to set the container policy:

- `useSortedSetClass(java.lang.Class concreteClass, java.util.Comparator comparator)`—Configure the mapping to use an instance of the specified `java.util.SortedSet` container class. Specify the `Comparator` to use to sort the target objects.

[Example 121–17](#) shows how to configure a `DirectCollectionMapping` to use a `java.util.ArrayList` container class.

#### **Example 121–17 Direct Collection Mapping**

```
// Create a new mapping and register it with the source descriptor
DirectCollectionMapping phonesMapping = new DirectCollectionMapping();
phonesMapping.setAttributeName("phones");
phonesMapping.setGetMethodName("getPhones");
phonesMapping.setSetMethodName("setPhones");
phonesMapping.setReferenceTableName("PHONES_TB");
phonesMapping.setDirectFieldName("PHONES");
phonesMapping.useCollectionClass(ArrayList.class); // set container policy
descriptor.addMapping(phonesMapping);
```

## 121.15 Configuring Attribute Transformer

A transformation mapping is made up of an attribute transformer for field-to-attribute transformation at read (unmarshal) time and one or more field transformers for attribute-to-field transformation at write (marshal) time (see [Section 121.16, "Configuring Field Transformer Associations"](#)).

This section describes how to configure the attribute transformer that a transformation mapping uses to perform the field-to-attribute transformation at read (unmarshal) time.

You can do this using either a method or class-based transformer.

A method-based transformer must map to a method in the domain object.

A class-based transformer allows you to place the transformation code in another class, making this approach non-intrusive: that is, your domain object does not need to implement a `TopLink` interface or provide a special transformation method

[Table 121–16](#) summarizes which mappings support this option.

**Table 121–16 Mapping Support for Attribute Transformer**

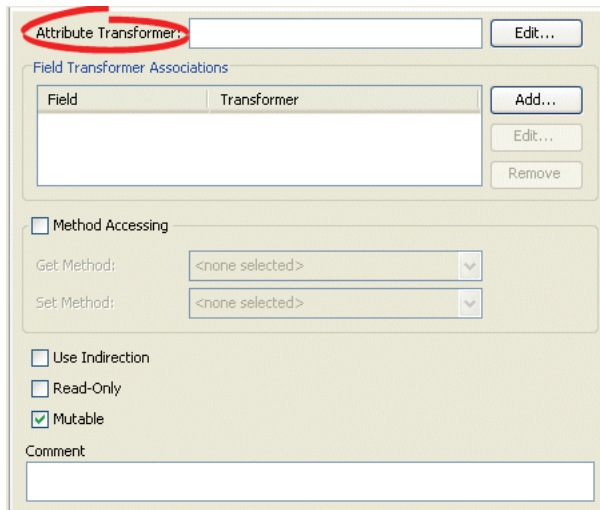
| Mapping                    | How to Use Oracle JDeveloper | How to Configure Attribute Transformer Using TopLink Workbench | How to Configure Attribute Transformer Using Java |
|----------------------------|------------------------------|----------------------------------------------------------------|---------------------------------------------------|
| Relational Mappings        |                              |                                                                |                                                   |
| Transformation Mapping     | ✓                            | ✓                                                              | ✓                                                 |
| EIS Mappings               |                              |                                                                |                                                   |
| EIS Transformation Mapping | ✓                            | ✓                                                              | ✓                                                 |
| XML Mappings               |                              |                                                                |                                                   |
| XML Transformation Mapping | ✓                            | ✓                                                              | ✓                                                 |

### 121.15.1 How to Configure Attribute Transformer Using TopLink Workbench

To specify a mapping’s attribute transformer, use this procedure:

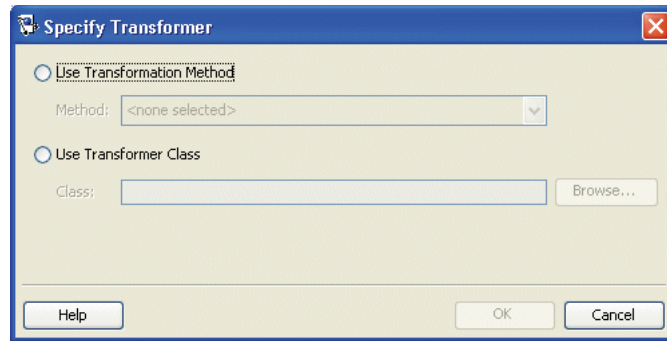
1. Select the transformation mapping in the **Navigator**. Its properties appear in the Editor.

**Figure 121–15 Transformation Mapping, Attribute Transformer Field**



2. Click **Edit**. The Specify Transformer dialog box appears.



**Figure 121–16 Specify Transformer Dialog Box**

Use the following information to enter data in each field of the dialog box and click **OK**:

| Field                     | Description                                                                                                                        |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| Use Transformation Method | Select a specific method to control the transformation. A method based transformer must map to a method in the domain object.      |
| Use Transformer Class     | Select a specific class to control the transformation. The class must be available on the TopLink Workbench application classpath. |

### 121.15.2 How to Configure Attribute Transformer Using Java

You can configure a method-based attribute transformer using `AbstractTransformationMapping` method `setAttributeTransformation`, passing in the name of the domain object method to use.

You can configure a class-based attribute transformer using `AbstractTransformationMapping` method `setAttributeTransformer`, passing in an instance of `oracle.toplink.mappings.Transformers.AttributeTransformer`.

A convenient way to create an `AttributeTransformer` is to extend `AttributeTransformerAdapter`.

## 121.16 Configuring Field Transformer Associations

A transformation mapping is made up of an attribute transformer for field-to-attribute transformation at read (unmarshall) time (see [Section 121.15, "Configuring Attribute Transformer"](#)) and one or more field transformers for attribute-to-field transformation at write (marshall) time.

This section describes how to configure the field transformers that a transformation mapping uses to perform the object attribute-to-field transformation at write (marshal) time.

You can do this using either a method or class-based transformer.

A method-based transformer must map to a method in the domain object.

A class-based transformer allows you to place the transformation code in another class, making this approach non-intrusive: that is, your domain object does not need to implement a TopLink interface or provide a special transformation method.

[Table 121–17](#) summarizes which mappings support this option.

**Table 121–17 Mapping Support for Field Transformer**

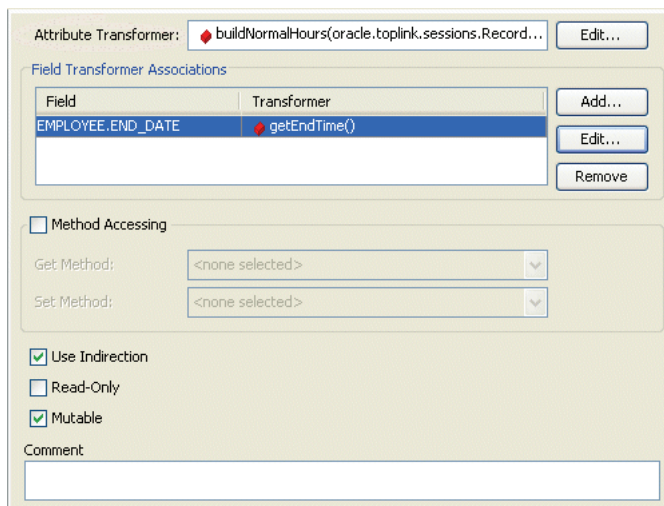
| Mapping                    | How to Use Oracle JDeveloper | How to Configure Field Transformer Associations Using TopLink Workbench | How to Configure Field Transformer Associations Using Java |
|----------------------------|------------------------------|-------------------------------------------------------------------------|------------------------------------------------------------|
| Relational Mappings        |                              |                                                                         |                                                            |
| Transformation Mapping     | ✓                            | ✓                                                                       | ✓                                                          |
| EIS Mappings               |                              |                                                                         |                                                            |
| EIS Transformation Mapping | ✓                            | ✓                                                                       | ✓                                                          |
| XML Mappings               |                              |                                                                         |                                                            |
| XML Transformation Mapping | ✓                            | ✓                                                                       | ✓                                                          |

### 121.16.1 How to Configure Field Transformer Associations Using TopLink Workbench

Use this procedure to complete the **Object->Field Method** fields:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.

**Figure 121–17 Transformation Mapping, Field Transformer Associations**



To add a new association, click **Add**. Continue with [Section 121.16.1.1, "Specifying Field-to-Transformer Associations"](#).

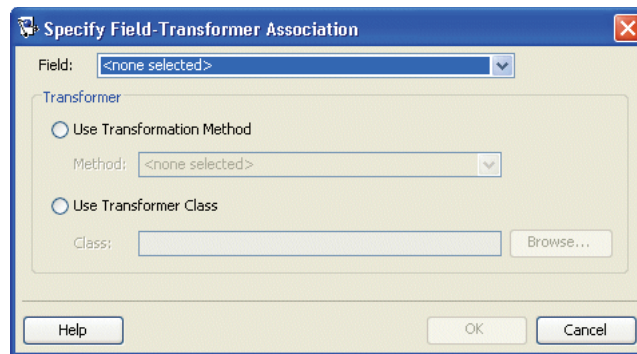
To change an existing association, click **Edit**. Continue with [Section 121.16.1.1, "Specifying Field-to-Transformer Associations"](#).

To delete an existing association, select the field transformation association and click **Delete**.

#### 121.16.1.1 Specifying Field-to-Transformer Associations

To specify the actual transformation method or class used for the field of a transformation mapping, use this procedure.

1. From the Transformation Mapping, Field Transformer Associations, click **Add** or **Edit**. The Specify Field-Transformer Association dialog box appears.

**Figure 121-18 Specify Field-Transformer Association Dialog Box**

Use the following information to complete each field on the dialog box:

| Field                     | Description                                                                                                                    |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| Field                     | Select the database field (from the descriptor's associated table) for this transformation.                                    |
| Transformer               | Select one of the following methods to control the transformation:                                                             |
| Use Transformation Method | Select a specific method to control the transformation. A method based transformer must map to a method in the domain object.  |
| Use Transformer Class     | Select a specific class to control the transformation. The class must be available on TopLink Workbench application classpath. |

### 121.16.2 How to Configure Field Transformer Associations Using Java

You can specify a specific transformation method on your domain object or an instance of `oracle.toplink.mappings.Transformers.FieldTransformer` (you can also extend the `FieldTransformerAdapter`). Using a `FieldTransformer` is non-intrusive: that is, your domain object does not need to implement a `TopLink` interface or provide a special transformation method.

You can configure a method-based field transformer using `AbstractTransformationMapping` method `addFieldTransformation`, passing in the name of the database field and the name of the domain object method to use.

You can configure a class-based field transformer using `AbstractTransformationMapping` method `addFieldTransformer`, passing in the name of the database field and an instance of `oracle.toplink.mappings.Transformers.FieldTransformer`.

A convenient way to create a `FieldTransformer` is to extend `FieldTransformerAdapter`.

## 121.17 Configuring Mutable Mappings

Direct mappings typically map simple, nonmutable values such as `String` or `Integer`. Transformation mappings can potentially map complex mutable object values, such as mapping several database field values to an instance of a Java class.

If a transformation mapping maps a mutable value, `TopLink` must clone and compare the value in a unit of work (see [Section 119.29, "Configuring Copy Policy"](#)).

By default, TopLink assumes that all transformation mappings are mutable. If the mapping maps a simple immutable value, you can improve the unit of work performance by configuring the `IsMutable` option to `false`.

By default, TopLink also assumes that all direct mappings are mutable unless a serialized converter is used. These mappings can also set the `IsMutable` option. You should set it if you want to modify `Date` or `Calendar` fields.

Table 121–18 summarizes which mappings support this option.

For more information, see Section 2.8.11, "Mutability".

**Table 121–18 Mapping Support for Mutable Mappings**

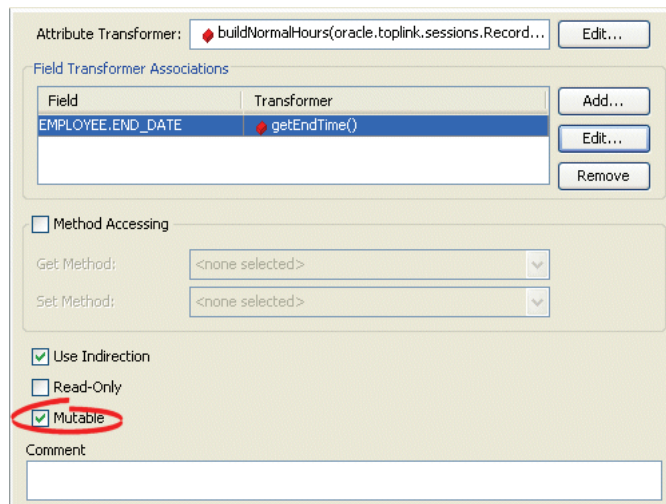
| Mapping                    | How to Use Oracle JDeveloper | How to Configure Mutable Mappings Using TopLink Workbench | How to Configure Mutable Mappings Using Java |
|----------------------------|------------------------------|-----------------------------------------------------------|----------------------------------------------|
| Relational Mappings        |                              |                                                           |                                              |
| Transformation Mapping     | ✓                            | ✓                                                         | ✓                                            |
| Direct-to-Field Mapping    |                              |                                                           | ✓                                            |
| EIS Mappings               |                              |                                                           |                                              |
| EIS Transformation Mapping | ✓                            | ✓                                                         | ✓                                            |
| EIS Direct Mapping         |                              |                                                           | ✓                                            |
| XML Mappings               |                              |                                                           |                                              |
| XML Transformation Mapping | ✓                            | ✓                                                         | ✓                                            |
| XML Direct Mapping         |                              |                                                           | ✓                                            |

### 121.17.1 How to Configure Mutable Mappings Using TopLink Workbench

Use this table to complete the **Object->Field Method** fields:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.

**Figure 121–19 Transformation Mapping, Mutable Option**



By default, the **IsMutable** option is selected in all transformation mappings. If the mapping maps to a simple atomic value, unselect this option.

## 121.17.2 How to Configure Mutable Mappings Using Java

You can specify whether or not a mapping is mutable using `AbstractTransformationMapping` method `setIsMutable` for transformation mappings, and `AbstractDirectMapping` method `isMutable` for direct mappings.

## 121.18 Configuring Bidirectional Relationship

TopLink can automatically manage the bidirectional relationship (see [Section 2.14.3.4, "Maintaining Bidirectional Relationships"](#)): if one side of the relationship is set or modified, TopLink will automatically set the other side. To enable this functionality, use the value holder indirection (see [Section 17.2.4.1, "Value Holder Indirection"](#)) for one-to-one mappings, and transparent collections (see [Section 17.2.4.2, "Transparent Indirect Container Indirection"](#))—for one-to-many and many-to-many mappings.

---

**Note:** Oracle does not recommend using this TopLink feature: if the object model is used outside the persistence context it must be responsible for managing the bidirectional relationship.

Instead, your application should maintain the bidirectional relationship in its getter and setter methods.

---

Table 121–18 summarizes which mappings support this option.

**Table 121–19 Mapping Support for Mutable Mappings**

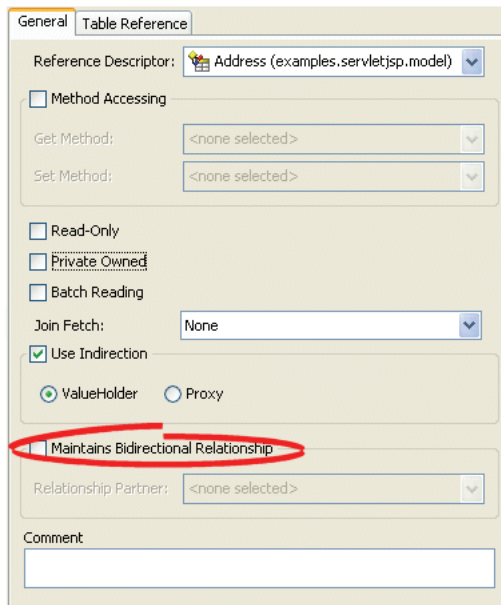
| Mapping                 | How to Use Oracle JDeveloper | How to Configure Bidirectional Relationship Using TopLink Workbench | How to Configure Bidirectional Relationship Using Java |
|-------------------------|------------------------------|---------------------------------------------------------------------|--------------------------------------------------------|
| Relational Mappings     |                              |                                                                     |                                                        |
| One-to-One Mapping      | ✓                            | ✓                                                                   | ✓                                                      |
| One-to-Many Mapping     | ✓                            | ✓                                                                   | ✓                                                      |
| Many-to-Many Mapping    | ✓                            | ✓                                                                   | ✓                                                      |
| EIS Mappings            |                              |                                                                     |                                                        |
| EIS One-to-One Mapping  | ✓                            | ✓                                                                   | ✓                                                      |
| EIS One-to-Many Mapping | ✓                            | ✓                                                                   | ✓                                                      |

### 121.18.1 How to Configure Bidirectional Relationship Using TopLink Workbench

To maintain a bidirectional relationship for a mapping, use this procedure:

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

**Figure 121–20** General tab, Maintains Bidirectional Relationship option



Use this table to enter data in the following fields on the tab:

| Field                                       | Description                                                                                                                                   |
|---------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Maintains Bidirectional Relationship</b> | Specify if TopLink should maintain the bidirectional link for this relational mapping.                                                        |
| <b>Relationship Partner</b>                 | Select the relationship partner (from the list of mapped attributes of the <b>Reference Descriptor</b> ) for this bidirectional relationship. |

### 121.18.2 How to Configure Bidirectional Relationship Using Java

If a mapping has a bidirectional relationship (see [Section 2.14.3.4, "Maintaining Bidirectional Relationships"](#)) in which the two classes in the relationship reference each other with one-to-one mappings, then set up the foreign key information as follows:

- One mapping must call the `setForeignKeyFieldName` method.
- The other mapping must call the `setTargetForeignKeyFieldName` method.

You can also set up the composite foreign key information by calling the `addForeignKeyFieldName` and `addTargetForeignKeyFieldName` methods. Because TopLink enables indirection (lazy loading) by default, the attribute must be a `ValueHolderInterface`.

---

**Note:** When your application does not use a cache, enable indirection for at least one object in a bidirectional relationship. In rare cases, disabling indirection on both objects in the bidirectional relationship can lead to infinite loops. For more information, see the following:

- [Section 27.2.1, "Directionality"](#)
  - [Section 2.14.3.4, "Maintaining Bidirectional Relationships"](#)
  - [Section 17.2.4, "Indirection \(Lazy Loading\)"](#)
-

[Example 121–18](#) demonstrates setting of bidirectional relationship between the `Policy` and `Carrier` classes. The foreign key is stored in the `Policy`'s table referencing the composite primary key of the `Carrier`.

**Example 121–18 Implementing a Bidirectional Mapping Between Two Classes that Reference Each Other**

```
public class Policy {
    ...
    // create the mapping that references the Carrier class
    OneToOneMapping carrierMapping = new OneToOneMapping();
    carrierMapping.setAttributeName("carrier");
    carrierMapping.setReferenceClass(Carrier.class);
    carrierMapping.addForeignKeyFieldName("INSURED_ID", "CARRIER_ID");
    carrierMapping.addForeignKeyFieldName("INSURED_TYPE", "TYPE");
    descriptor.addMapping(carrierMapping);
    ...
}

public class Carrier {
    ...
    // create the mapping that references the Policy class
    OneToOneMapping policyMapping = new OneToOneMapping();
    policyMapping.setAttributeName("masterPolicy");
    policyMapping.setReferenceClass(Policy.class);
    policyMapping.addTargetForeignKeyFieldName("INSURED_ID", "CARRIER_ID");
    policyMapping.addTargetForeignKeyFieldName("INSURED_TYPE", "TYPE");
    descriptor.addMapping(policyMapping);
    ...
}
```

## 121.19 Configuring the Use of a Single Node

For the XML-based mappings that [Table 121–6](#) summarizes, when you map a list value, you can configure whether or not the mapping unmarshalls (writes) the list to a single node, like `<item>aaa bbb ccc</item>`, or to multiple nodes, like the following:

```
<item>aaa</item>
<item>bbb</item>
<item>ccc</item>
```

[Table 121–6](#) summarizes which mappings support this option.

**Table 121–20 Mapping Support for Use Single Node**

| Mapping                                 | How to Use Oracle JDeveloper | How to Configure the Use of a Single Node Using TopLink Workbench | How to Configure the Use of a Single Node Using Java |
|-----------------------------------------|------------------------------|-------------------------------------------------------------------|------------------------------------------------------|
| EIS Mappings <sup>1</sup>               |                              |                                                                   |                                                      |
| EIS Direct Mapping                      |                              |                                                                   | ✓                                                    |
| EIS Composite Direct Collection Mapping | ✓                            | ✓                                                                 | ✓                                                    |
| XML Mappings                            |                              |                                                                   |                                                      |
| XML Direct Mapping                      |                              |                                                                   | ✓                                                    |

**Table 121–20 (Cont.) Mapping Support for Use Single Node**

| Mapping                                 | How to Use Oracle JDeveloper | How to Configure the Use of a Single Node Using TopLink Workbench | How to Configure the Use of a Single Node Using Java |
|-----------------------------------------|------------------------------|-------------------------------------------------------------------|------------------------------------------------------|
| XML Composite Direct Collection Mapping | ✓                            | ✓                                                                 | ✓                                                    |
| XML Binary Data Mapping                 |                              |                                                                   | ✓                                                    |
| XML Binary Data Collection Mapping      | ✓                            | ✓                                                                 | ✓                                                    |

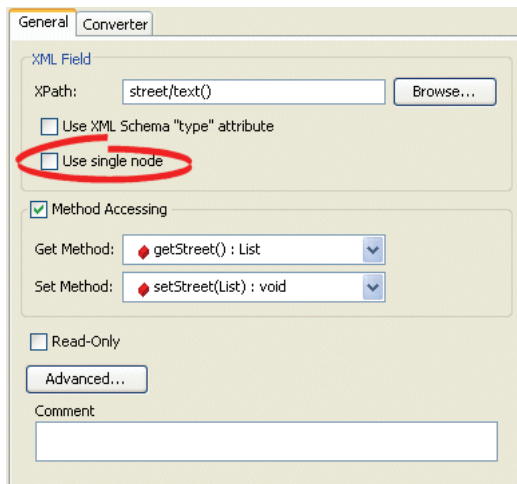
<sup>1</sup> When used with XML records only (see Section 76.4, "Configuring Record Format").

### 121.19.1 How to Configure the Use of a Single Node Using TopLink Workbench

To configure a mapping to use a single node, use this procedure.

1. Select the mapped attribute in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The General tab appears.

**Figure 121–21 General Tab, Use Single Node Option**



To configure the mapping to unmarshal (write) a list value to a single node (like `<item>aaa bbb ccc</item>`), click **Use single node**.

By default, the mapping unmarshalls a list value to separate nodes.

### 121.19.2 How to Configure the Use of a Single Node Using Java

Use `AbstractCompositeDirectCollectionMapping` method `setUsesSingleNode` to configure the mapping to write a list value to a single node by passing in a value of `true`. To configure the mapping to write a list value to multiple nodes, pass in a value of `false`.

For any mapping that takes an `XMLField`, use `XMLField` method `setUsesSingleNode` to configure the mapping to write a list value to a single node by passing in a value of `true`. To configure the mapping to write a list value to multiple nodes, pass in a value of `false`. [Example 121–19](#) shows how to use this method with an `XMLDirectMapping`:



**Example 121–19 Using XMLField Method setUsesSingleNode**

```
XMLDirectMapping tasksMapping = new XMLDirectMapping();
tasksMapping.setAttributeName("tasks");
XMLField myField = new XMLField("tasks/text()"); // pass in the XPath
myField.setUsesSingleNode(true);
tasksMapping.setField(myField);
```

## 121.20 Configuring the Use of CDATA

For the XML-based mappings that [Table 121–6](#) summarizes, when you create a mapping, you can configure whether or not the mapping’s text is wrapped in a `<![CDATA[ . . . ]>` statement.

[Table 121–6](#) summarizes which mappings support this option.

**Table 121–21 Mapping Support for Use of CDATA**

| Mapping                                 | How to Use Oracle JDeveloper | How to Use TopLink Workbench | How to Configure the Use of CDATA Using Java |
|-----------------------------------------|------------------------------|------------------------------|----------------------------------------------|
| XML Mappings                            |                              |                              |                                              |
| XML Direct Mapping                      |                              |                              | ✓                                            |
| XML Composite Direct Collection Mapping |                              |                              | ✓                                            |
| XML Binary Data Mapping                 |                              |                              | ✓                                            |
| XML Binary Data Collection Mapping      |                              |                              | ✓                                            |

### 121.20.1 How to Configure the Use of CDATA Using Java

Use the `isCDATA()` method on an `XMLDirectMapping` or `XMLCompositeDirectCollectionMapping` to specify if the mapping’s text is wrapped in a `<![CDATA[ . . . ]>` statement.

[Example 121–19](#) shows the results of using this method:

**Example 121–20 Using CDATA**

When `isCDATA = false` on the name mapping, TopLink writes the text as a regular text node:

```
<employee>
  <name>Jane Doe</name>
</employee>
```

When `isCDATA = true` on the name mapping, TopLink wraps the text in a `<![CDATA[ . . . ]>` statement:

```
<employee>
  <name>
    <![CDATA[Jane Doe]]>
  </name>
</employee>
```



---

---

# Troubleshooting a TopLink Application

This appendix includes the following sections:

- [TopLink Support for Oracle Application Server Manageability and Diagnosability](#)
- [TopLink Exception Error Reference](#)—lists the types exceptions that may occur at time of deployment of a TopLink application, as well as at run time.
- [TopLink Workbench Error Reference](#)—describes common problems and their solutions when using TopLink Workbench.

## A.1 TopLink Support for Oracle Application Server Manageability and Diagnosability

In this release, TopLink-enabled applications deployed to the Oracle Application Server support Oracle Application Server Manageability and Diagnosability to simplify the management of applications and to simplify problem diagnosis and resolution.

TopLink Manageability and Diagnosability support includes the following:

- [Oracle Application Server Manageability and Diagnosability Logging Enhancements](#)
- [Oracle Dynamic Monitoring System \(DMS\) Sensor Enhancements](#)
- [Manageability and Diagnosability JMX Enhancements](#)

For more information about using Manageability and Diagnosability in Oracle Application Server, see the following:

- *Oracle Fusion Middleware 2 Day Administration Guide*
- *Oracle Fusion Middleware Administrator's Guide*
- *Oracle Fusion Middleware Administration and Application Deployment Guide for Oracle Containers for Java EE*
- *Oracle Fusion Middleware Performance Guide*

### A.1.1 Oracle Application Server Manageability and Diagnosability Logging Enhancements

TopLink integrates its logs with the Oracle Application Server Manageability and Diagnosability logging infrastructure to make TopLink messages visible to Manageability and Diagnosability-enabled management tools like Oracle Enterprise Manager. This allows Oracle Application Server to include TopLink log information in end-to-end transaction tracing, log correlation, and incident generation.

When you deploy a TopLink-enabled application to an application server or EJB container, TopLink JPA and TopLink CMP default to `ServerLog` with no log level so that TopLink uses the configuration in `j2ee-logging.xml`. When you deploy a TopLink-enabled application to Oracle Application Server, this default ensures that TopLink log messages integrate with Oracle Application Server Manageability and Diagnosability.

When you deploy a TopLink-enabled application outside of an EJB container, the logging defaults revert to `DefaultSessionLog` and `WARNING` log level.

TopLink exports all its loggers to Oracle Enterprise Manager to consolidate all logging configuration under Oracle Enterprise Manager. Logger configuration is no longer dependent upon TopLink session name. This allows logger configuration from Oracle Enterprise Manager even when there is no deployed TopLink session.

For more information, see the following:

- [Section 87.2.6, "Logging"](#)
- [Section 89.4, "Configuring Logging"](#)

### A.1.2 Oracle Dynamic Monitoring System (DMS) Sensor Enhancements

TopLink provides DMS sensors to supply Manageability and Diagnosability-enabled management tools like Oracle Enterprise Manager with advanced TopLink operational data. This allows Oracle Application Server to provide improved TopLink application management and optimization.

For more information, see the following:

- [Section 12.4, "Measuring TopLink Performance with the Oracle Dynamic Monitoring System \(DMS\)"](#)
- [Section 12.4.1, "How to Configure the Oracle DMS Profiler"](#)
- [Section A.1.3, "Manageability and Diagnosability JMX Enhancements"](#)
- *Oracle Fusion Middleware Administrator's Guide*

### A.1.3 Manageability and Diagnosability JMX Enhancements

TopLink integrates with Oracle Application Server MBeans to ensure that TopLink applications are manageable using Oracle Enterprise Manager.

For more information, see the following:

- [Section 12.4.2, "How to Access Oracle DMS Profiler Data Using JMX"](#)
- *Oracle Fusion Middleware Administrator's Guide*

## A.2 TopLink Exception Error Reference

This section lists the types of TopLink exceptions. For detailed information on each exception including the likely cause of the problem and possible corrective actions, see *Oracle Fusion Middleware Error Messages Reference*.

Each TopLink exception has a code assigned to it. The code corresponds to an exception class and includes the following information:

- The exception number in the format of `EXCEPTION [TOP-XXXX]`
- A description of the problem, taken from the raised exception

This section contains information on the following types of TopLink exceptions:

- Descriptor Exceptions
- Concurrency Exceptions
- Conversion Exceptions
- Database Exceptions
- Optimistic Lock Exceptions
- Query Exceptions
- Validation Exceptions
- EJB QL Exceptions
- Session Loader Exceptions
- EJB Exception Factory Exceptions
- EIS Exceptions
- JMS Processing Exceptions
- Default Mapping Exceptions
- Discovery Exceptions
- Remote Command Manager Exceptions
- Transaction Exceptions
- XML Conversion Exceptions
- XML Marshal Exceptions
- Migration Utility Exceptions
- XML Platform Exceptions
- Entity Manager Setup Exceptions
- EJB JAR XML Exceptions

### A.2.1 Descriptor Exceptions

`DescriptorException` is a development exception that is raised when insufficient information is provided to the descriptor. The message that is returned includes the name of the descriptor or mapping that caused the exception. If a mapping within the descriptor caused the error, then the name and parameters of the mapping are part of the returned message, as [Example A-1](#) demonstrates.

Internal exception, mapping and descriptor appear only if TopLink has enough information about the source of the problem to provide this information.

#### Format

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
INTERNAL EXCEPTION: Message
MAPPING: Database mapping
DESCRIPTOR: Descriptor
```

#### **Example A-1** Descriptor Exception

```
EXCEPTION [TOP - 75]: oracle.toplink.exceptions.DescriptorException
EXCEPTION DESCRIPTION: The reference class is not specified.
```

## A.2.2 Concurrency Exceptions

`ConcurrencyException` is a development exception that is raised when a Java concurrency violation occurs. Only when a running thread is interrupted, causing the JVM to throw an `InterruptedException`, is an internal exception information displayed with the error message, as [Example A-2](#) shows.

### Format

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
INTERNAL EXCEPTION: Message
```

#### **Example A-2 Concurrency Exception**

```
EXCEPTION [TOP - 2004]: oracle.toplink.exceptions.ConcurrencyException
EXCEPTION DESCRIPTION: Signal attempted before wait on concurrency manager.
This usually means that an attempt was made to commit or roll back a transaction
before being started, or rolled back twice.
```

## A.2.3 Conversion Exceptions

`ConversionException` is a development exception that is raised when a conversion error occurs by an incompatible type conversion. The message that is returned indicates which type cast caused the exception, as shown in [Example A-3](#).

### Format

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
INTERNAL EXCEPTION: Message
```

#### **Example A-3 Conversion Exception**

```
EXCEPTION [TOP - 3006]: oracle.toplink.exceptions.ConversionException
EXCEPTION DESCRIPTION: object must be of even length to be converted to a
ByteArray
```

## A.2.4 Database Exceptions

`DatabaseException` is a run-time exception that is raised when data read from the database, or the data that is to be written to the database, is incorrect. The exception may also act as a wrapper for `SQLException`. If this is the case, the message contains a reference to the error code and error message, as shown in [Example A-4](#).

This exception can occur on any database operation. If an execution of a SQL script is involved in a database operation causing `DatabaseException`, the exception's message, accessible through the `getMessage` method, contains the SQL that caused this exception.

This exception includes internal exception and error code information when the exception is wrapping a `SQLException`.

### Format

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
INTERNAL EXCEPTION: Message
ERROR CODE: Error code
```

**Example A-4 Database Exception**

```
EXCEPTION [TOP - 4002]: oracle.toplink.exceptions.DatabaseException
EXCEPTION DESCRIPTION: java.sql.SQLException: [INTERSOLV][ODBC dBase driver]
Incompatible datatypes in expression: >
INTERNAL EXCEPTION: java.sql.SQLException: [INTERSOLV][ODBC dBase driver]
Incompatible datatypes in expression: >
ERROR CODE: 3924
```

**A.2.5 Optimistic Lock Exceptions**

`OptimisticLockException` is a run-time exception that is raised when the row on the database that matches the desired object is missing or when the value on the database does not match the registered number. It is used in conjunction with the optimistic locking feature. This applies only on an update or delete operation, as shown in [Example A-5](#).

For more information about optimistic locking, see the section on optimistic locking in a stateless environment in [Chapter 2, "Introduction to TopLink Application Development"](#). These exceptions should be handled in a try-catch block.

**Format**

```
EXCEPTION [TOP - error code]: Exception Name
EXCEPTION DESCRIPTION: Message
```

**Example A-5 Optimistic Lock Exception**

```
EXCEPTION [TOP - 5003]: oracle.toplink.exceptions.OptimisticLockException
EXCEPTION DESCRIPTION: The object, object.toString() cannot be deleted because it
has changed or been deleted since it was last read.
```

**A.2.6 Query Exceptions**

`QueryException` is a development exception that is raised when insufficient information has been provided to the query. If possible, the message indicates the query that caused the exception. A query is optional and is displayed if TopLink is able to determine the query that caused this exception, as shown in [Example A-6](#).

**Format**

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
QUERY:
```

**Example A-6 Query Exception**

```
EXCEPTION [TOP - 6026]: oracle.toplink.exceptions.QueryException
EXCEPTION DESCRIPTION: The query is not defined. When executing a query on the
session, the parameter that takes the query is null.
```

**A.2.7 Validation Exceptions**

`ValidationException` is a development exception that is raised when an incorrect state is detected or an API is used incorrectly.

**Format**

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
```

**Example A-7 Validation Exception**

```
EXCEPTION [TOP - 7008]: oracle.toplink.exceptions.ValidationException
EXCEPTION DESCRIPTION: The Java type javaClass is not a valid database type. The
Java type of the field to be written to the database has no corresponding type on
the database.
```

## A.2.8 EJB QL Exceptions

`EJBQLException` is a run-time exception that is raised when the EJB QL string does not parse properly, or the contents cannot be resolved within the context of the TopLink session. The associated message typically includes a reference to the EJB QL string that caused the problem.

**Format**

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
```

**Example A-8 EJB QL Exception**

```
EXCEPTION [TOP - 8002]: oracle.toplink.exceptions.EJBQLException
EXCEPTION DESCRIPTION: TopLink has encountered a problem while parsing the EJB QL
string.
```

## A.2.9 Session Loader Exceptions

`SessionLoaderException` is a run-time exception that is raised if the session manager encounters a problem loading session information from a `sessions.xml` (for non-EJB applications) or `toplink-ejb-jar.xml` (for EJB applications) properties file.

**Format**

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
```

**Example A-9 Session Loader Exception**

```
EXCEPTION [TOP - 9004]: oracle.toplink.exceptions.SessionLoaderException
EXCEPTION DESCRIPTION: The <project-xml> file MyProject was not found on the
classpath, nor on the filesystem.
```

## A.2.10 EJB Exception Factory Exceptions

An EJB exception factory generates run-time exceptions that are raised if a container provider specific to a given application server encounters a problem during any stage of the life cycle of an EJB bean.

**Format**

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
```

**Example A-10 EJB Exception Factory Exception**

```
EXCEPTION [TOP - 10008]: javax.ejb.CreateException
EXCEPTION DESCRIPTION: Cannot find bean.
```



## A.2.11 Communication Exceptions

`CommunicationException` is a run-time exception that wraps all RMI, CORBA, or input and output exceptions that occur.

### Format

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
```

### *Example A-11 Communication Exception*

```
EXCEPTION [TOP - 12000]: oracle.toplink.exceptions.CommunicationException
EXCEPTION DESCRIPTION: Error Sending connection service to myService.
```

## A.2.12 EIS Exceptions

`EISException` is a run-time exception that is raised when invoking EIS interactions. For more information on EIS interactions, see [Section 108.9.3, "Enterprise Information System \(EIS\) Interactions"](#).

### Format

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
```

### *Example A-12 JMS Processing Exception*

```
EXCEPTION [TOP - 17010]: oracle.toplink.eis.EISException
EXCEPTION DESCRIPTION: Output record contains an unsupported message type.
```

## A.2.13 JMS Processing Exceptions

`JMSProcessingException` is a run-time exception that is raised when processing JMS messages.

### Format

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
```

### *Example A-13 JMS Processing Exception*

```
EXCEPTION [TOP - 18001]: oracle.toplink.exceptions.JMSProcessingException
EXCEPTION DESCRIPTION: Error while processing incoming JMS message.
```

## A.2.14 Default Mapping Exceptions

`DefaultMappingException` is a run-time exception that is raised when an error occurs during OC4J CMP default mapping.

### Format

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
```

### *Example A-14 Default Mapping Exception*

```
EXCEPTION [TOP - 20002]: oracle.toplink.exceptions.DefaultMappingException
EXCEPTION DESCRIPTION: The finder method with the parameters as defined in the
ejb-jar.xml file, is not found in the home of bean.
```

## A.2.15 Discovery Exceptions

`DiscoveryException` is a run-time exception that is raised when `DiscoveryManager` is operating.

### Format

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
```

### *Example A-15 Discovery Exception*

```
EXCEPTION [TOP - 22001]: oracle.toplink.exception.DiscoveryException
EXCEPTION DESCRIPTION: Could not join multicast group.
```

## A.2.16 Remote Command Manager Exceptions

`RemoteCommandManagerException` is a run-time exception that is raised when the remote command module is used.

### Format

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
```

### *Example A-16 Remote Command Manager Exception*

```
EXCEPTION [TOP - 22104]: oracle.toplink.exceptions.RemoteCommandManagerException
EXCEPTION DESCRIPTION: Could not look up host name.
```

## A.2.17 Transaction Exceptions

`TransactionException` is a run-time exception that is raised when an error is encountered during a transaction. When this occurs, the message contains a reference to the error code and error message.

### Format

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
```

### *Example A-17 Transaction Exception*

```
EXCEPTION [TOP - 23001]: oracle.toplink.exceptions.TransactionException
EXCEPTION DESCRIPTION: Error looking up external Transaction resource under JNDI
name.
```

## A.2.18 XML Conversion Exceptions

`XMLConversionException` is a run-time exception that is raised when a conversion between TopLink instances and XML fails. This exception is used in cache coordination that uses XML change sets.

### Format

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
```

### *Example A-18 XML Conversion Exception*

```
EXCEPTION [TOP - 25001]: oracle.toplink.exceptions.XMLConversionException
EXCEPTION DESCRIPTION: Cannot create URL for file [\\FILE_SERVER\command.xml].
```

## A.2.19 XML Marshal Exceptions

`XMLMarshalException` is raised when an error is encountered during the XML marshalling process.

### Format

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
```

### Example A-19 XML Marshal Exception

```
EXCEPTION [TOP - 25001]: oracle.toplink.exceptions.XMLMarshalException
EXCEPTION DESCRIPTION: Error while trying to create session.
```

## A.2.20 Migration Utility Exceptions

`MigrationUtilityException` is a run-time exception that is raised when an error is encountered during the use of the TopLink migration utility.

### Format

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
```

### Example A-20 Migration Utility Exception

```
EXCEPTION [TOP - 26002]: oracle.toplink.exceptions.MigrationUtilityException
EXCEPTION DESCRIPTION: The program security manager prevents the migration utility
from creating a JAR class loader for the JAR file.
```

## A.2.21 XML Platform Exceptions

`XMLPlatformException` is raised when an error related to XML platform is encountered.

### Format

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
```

### Example A-21 EJB JAR XML Exception

```
EXCEPTION [TOP - 27001]: oracle.toplink.platform.xml.XMLPlatformException
EXCEPTION DESCRIPTION: The XML platform class ClassName was not found.
```

## A.2.22 Entity Manager Setup Exceptions

`EntityManagerSetupException` is raised when an error is encountered during the process of setting up an entity manager.

### Format

```
EXCEPTION [TOP - error code]: Exception name
EXCEPTION DESCRIPTION: Message
```

### Example A-22 Entity Manager Setup Exception

```
EXCEPTION [TOP - 28001]: oracle.toplink.exceptions.EntityManagerSetupException
EXCEPTION DESCRIPTION: Error while trying to create session.
```

## A.2.23 EJB JAR XML Exceptions

`EJBJARXMLException` is a run-time exception that is raised at deployment time when the `ejb-jar.xml` file is read and the required concrete EJB classes code is generated.

### Format

EXCEPTION [TOP - error code]: Exception name  
EXCEPTION DESCRIPTION: Message

#### **Example A-23 EJB JAR XML Exception**

EXCEPTION [TOP - 72000]: oracle.toplink.exceptions.EJBJarXMLException  
EXCEPTION DESCRIPTION: Error reading ejb-jar.xml file.

## A.3 TopLink Workbench Error Reference

TopLink checks each project, descriptor, and mapping to ensure that you have properly defined the required settings. Errors and warnings are displayed in the Problems window (see [Section 5.3.5, "How to Use the Problems Window"](#)) of TopLink Workbench.

You can also create a project status report (see [Section 116.2.3, "How to Generate the Project Status Report"](#)) that contains all errors in a specific project.

This section contains information on the following Oracle TopLink Workbench errors:

- [Miscellaneous Errors \(1 – 89, 106 – 133\)](#)
- [Project Errors \(100 – 102\)](#)
- [Descriptor Errors \(200 – 399\)](#)
- [Mapping Errors \(400 – 483\)](#)
- [Table Errors \(500 – 610\)](#)
- [XML Schema Errors \(700 – 706\)](#)
- [Session Errors \(800 – 812\)](#)

This section also includes information on the following:

- [Common Classpath Problems](#)
- [Database Connection Problems](#)

### A.3.1 Miscellaneous Errors (1 – 89, 106 – 133)

This section lists TopLink Workbench error codes, information about the likely cause of the problem, and a possible corrective action.

#### **13: No class indicator field should be defined for the abstract class [class].**

**Cause:** Abstract classes should not be included or contain an **Indicator Value** on a descriptor's **Inheritance** tab.

**Action:** You must either remove the **Include** option for the class on the **Inheritance** tab, or remove the **abstract** modifier option on the descriptor's **Class Info – Class** tab. See [Section 16.2.2, "Descriptors and Inheritance"](#) and [Chapter 119, "Configuring a Descriptor"](#).

#### **54: No class indicator field is selected for this root class.**

**Cause:** You selected the **Use Class Indicator Field** option for the root descriptor in the inheritance hierarchy, but did not specify an indicator field for the root and its children.

**Action:** Use the **Field Selection** list on the **Inheritance** tab for the root class. See [Section 16.2.2, "Descriptors and Inheritance"](#) and [Chapter 119, "Configuring a Descriptor"](#).

**55: No class indicator value is defined for this included descriptor [class]**

**Cause:** You selected the **Use Class Indicator Dictionary** option for the root descriptor in the inheritance hierarchy, but did not specify an indicator value for the root and its children.

**Action:** Use the **Indicator Type** list on the **Inheritance** tab for the root class. See [Section 16.2.2, "Descriptors and Inheritance"](#) and [Chapter 119, "Configuring a Descriptor"](#).

**89: Root class does not include an indicator mapping for this descriptor.**

**Cause:** The root class in the inheritance hierarchy is set to the **Use Class Indicator Dictionary** option. The dictionary does not contain an indicator value for this child class.

**Action:** Select an **Indicator Type** on the **Inheritance** tab of the root class that includes the child types. See [Section 16.2.2, "Descriptors and Inheritance"](#) and [Chapter 119, "Configuring a Descriptor"](#).

**106: Multiple mapping [mappings] write to the database field [db field].**

**Cause:** One database field is populated by more than one mapping

**Action:** Ensure the "one mapping per field" ratio for write operations.

**118: The selected parent descriptor for this descriptor's inheritance policy does not have an associated inheritance policy.**

**Cause:** Missing descriptor's inheritance policy.

**Action:** Ensure that the descriptor you are using has a valid associated inheritance policy (`InheritancePolicy`). See [Section 16.2.2, "Descriptors and Inheritance"](#) and [Section 119.21, "Configuring Inheritance for a Parent \(Root\) Descriptor"](#).

**123: This root class has no class indicator mappings for its hierarchy.**

**Cause:** You created an inheritance policy with the **Use Class Indicator Dictionary** option but did not specify the indicator values for all subclasses.

**Action:** Specify the indicator values for all subclasses on the descriptor's **Inheritance** tab. See [Section 16.2.2, "Descriptors and Inheritance"](#) and [Section 119.20, "Configuring Inheritance for a Child \(Branch or Leaf\) Class Descriptor"](#).

---

**Note:** TopLink displays a list of each subclass and indicator value if you have identified the subclasses' parent descriptor.

---

**126: Writable mappings defined for the class indicator field [field name].**

**Cause:** You selected the **Use Class Indicator Field** option for the root descriptor in the inheritance hierarchy, but the mappings for this field are writable.

**Action:** Select a **Use Class Indicator Field** on the descriptor's **Inheritance** tab that does not contain any writable mappings. See the following:

- [Section 16.2.2, "Descriptors and Inheritance"](#)
- [Chapter 119, "Configuring a Descriptor"](#)
- [Section 119.20, "Configuring Inheritance for a Child \(Branch or Leaf\) Class Descriptor"](#)
- [Section 119.23, "Configuring Inherited Attribute Mapping in a Subclass"](#)

**132: The implemented interface [interface] is not an interface.**

**Cause:** You selected a noninterface (a class) as an implemented interface.

**Action:** Ensure that you select an interface. See [Section 16.2.2, "Descriptors and Inheritance"](#) and [Chapter 119, "Configuring a Descriptor"](#).

**133: The superclass for [class] is an interface, classes cannot extend interfaces.**

**Cause:** You selected an interface instead of a class as a parent for your child class.

**Action:** Use the **Inheritance** tab for the root class. See [Section 16.2.2, "Descriptors and Inheritance"](#) and [Chapter 119, "Configuring a Descriptor"](#).

### A.3.2 Project Errors (100 – 102)

This section lists TopLink Workbench project errors.

**100: The project caches all statements by default for queries, but does not bind all parameters.**

**Cause:** A named query that caches statements must also bind parameters.

**Action:** On the **Named Queries – Options** tab, change the **Cache Statement** field to **False**, or change the **Bind Parameters** field to **True**. See [Section 119.7.1.9, "Configuring Named Query Options"](#).

**101: The project uses a custom sequence table, but the counter field is not specified.**

**Cause:** On the project's **Sequencing** tab, you selected **Use Custom Sequence Table**, but did not complete the **Counter Field** field.

**Action:** Select the field to use as the Counter Field for this sequence table. See [Section 20.3, "Configuring Sequencing at the Project Level"](#) for details.

**102: The project uses a custom sequence table, but the name field is not specified.**

**Cause:** On the project's **Sequencing** tab, you selected **Use Custom Sequence Table**, but did not complete the **Name Field** field.

**Action:** Select the field to use as the **Name Field** for this sequence table. See [Section 20.3, "Configuring Sequencing at the Project Level"](#) for details.

### A.3.3 Descriptor Errors (200 – 399)

This section lists TopLink Workbench descriptor errors.

**200: The descriptor's class is not public.**

**Cause:** The descriptor must use a public access modifier.

**Action:** On the descriptor's **Class Info – Class** tab, change the **Access Modifier** option to **Public**. See [Section 5.7.2, "How to Configure Classes"](#) and [Section 5.7.2.2, "Configuring Class Modifiers"](#).

**201: This class is a subclass of a final class.**

**Cause:** If you select the **Final** option on the descriptor's **Class Info – Class** tab for a class, then the class cannot contain subclasses.

- Action:** See [Section 5.7.2, "How to Configure Classes"](#) and [Section 5.7.2.1, "Configuring Class Information"](#).
- 210: Two methods *[method name1]* *[method name2]* cannot have the same signature.**  
**Cause:** You created methods with identical signatures.  
**Action:** On the **Class Info – Methods** tab, change the information for one of the methods. See [Section 5.7.2, "How to Configure Classes"](#) and [Section 5.7.2.8, "Adding Methods"](#).
- 211: The format for *[date]* must be in the format HH-MM-SS or HH:MM:SS. Literal argument of expression *[line number]* on query *[query name]* is invalid.**  
**Cause:** You attempted to use an invalid argument on a query.  
**Action:** Use HH-MM-SS or HH:MM:SS format. See [Section 17.2.6.2, "Type Conversion Converter"](#) and [Section 121.10, "Configuring a Type Conversion Converter"](#).
- 212: The format for *[date]* must be in the format YYYY/MM/DD HH:MM:SS or YYYY-MM-DD HH:MM:SS. Literal argument of expression *[line number]* on query *[query name]* is invalid.**  
**Cause:** You attempted to use an invalid argument on a query.  
**Action:** Use YYYY/MM/DD HH:MM:SS or YYYY-MM-DD HH:MM:SS format. See [Section 17.2.6.2, "Type Conversion Converter"](#) and [Section 121.10, "Configuring a Type Conversion Converter"](#).
- 213: The format for *[date]* must be in the format YYYY/MM/DD or YYYY-MM-DD. Literal argument of expression *[line number]* on query *[query name]* is invalid.**  
**Cause:** You attempted to use an invalid argument on a query.  
**Action:** Use YYYY/MM/DD or YYYY-MM-DD format. See [Section 17.2.6.2, "Type Conversion Converter"](#) and [Section 121.10, "Configuring a Type Conversion Converter"](#).
- 214: The format for *[date]* must be in the format YYYY/MM/DD HH:MM:SS, YYYY/MM/DD, or YYYY-MM-DD. Literal argument of expression *[line number]* on query *[query name]* is invalid.**  
**Cause:** You attempted to use an invalid argument on a query.  
**Action:** Use YYYY/MM/DD HH:MM:SS, YYYY/MM/DD, or YYYY-MM-DD format. See [Section 17.2.6.2, "Type Conversion Converter"](#) and [Section 121.10, "Configuring a Type Conversion Converter"](#).
- 215: The format for *[argument]* must be an even length HEX string. Literal argument of expression *[line number]* on query *[query name]* is invalid.**  
**Cause:** You attempted to use an invalid argument on a query.  
**Action:** Use HEX format. See [Section 17.2.6.2, "Type Conversion Converter"](#).
- 216: The format for *[argument]* must be a string. Literal argument of expression *[line number]* on query *[query name]* is invalid.**  
**Cause:** You attempted to use an invalid argument on a query.  
**Action:** Use a String. See [Section 17.2.6.2, "Type Conversion Converter"](#).
- 217: Literal argument of expression *[line number]* on query *[query name]* is invalid. The format is illegal.**  
**Cause:** You attempted to use an invalid argument on a query.  
**Action:** Use a valid format.

**220: An aggregate shared by multiple source descriptors cannot have one-to-many or many-to-many mappings.**

**Cause:** You attempted to create multiple one-to-many and many-to-many, or one-to-one mappings in which the target is the aggregate. Aggregate descriptors that are shared by multiple source descriptors cannot have mappings that contain a target object that references the descriptor.

**Action:** For aggregate descriptors that are shared by multiple source descriptors, remove mappings that contain a target object that references the descriptor. See [Part VII, "Descriptors"](#) and [Section 22.2.1.2, "Creating Relational Aggregate Descriptors"](#).

**221: Classes cannot reference an aggregate target with one-to-one, one-to-many, or many-to-many mappings.**

**Cause:** You tried to select an aggregate descriptor as a reference.

**Action:** Do not select an aggregate descriptor as the **Reference Descriptor** for a one-to-one, one-to-many, or many-to-many mapping. See [Part VII, "Descriptors"](#) and [Section 22.2.1.2, "Creating Relational Aggregate Descriptors"](#).

**225: The implementor [*implementor name*] no longer implements this interface.**

**Cause:** One descriptor listed as an implementation method for this interface descriptor no longer implements this descriptor's interface.

**Action:** Either remove the descriptor from the list of implementation methods or alter the descriptor's class so that it implements this descriptor's interface. See [Part VII, "Descriptors"](#) and [Section 22.2.1.3, "Creating Relational Interface Descriptors"](#).

**230: No primary table is specified.**

**Cause:** The descriptor is not associated with a database table.

**Action:** On the descriptor's **Descriptor Info** tab, use the **Associated Table** field to select a primary table. See [Section 23.2, "Configuring Associated Tables"](#).

**231: No primary key(s) specified in [*table name*] table.**

**Cause:** You did not specify a primary key for each database table. When importing tables from a database into TopLink Workbench, the primary key information will be retained only if the JDBC driver supports the `getPrimaryKeys` method.

**Action:** Ensure that a primary key is specified for each descriptor on the **Descriptor Info** tab. See [Section 23.2, "Configuring Associated Tables"](#).

**232: The following primary key field is unmapped [*field name*].**

**Cause:** The primary key field does not have a writable mapping.

**Action:** Ensure that the primary key(s) are mapped. See [Section 23.2, "Configuring Associated Tables"](#).

**233: The number of primary keys does not match the number of primary keys on the parent.**

**Cause:** In an inheritance hierarchy, the child class does not have the same number of primary keys as the parent class.

**Action:** Ensure that the parent and child class have the same number of primary keys on the descriptor's **Descriptor Info** tab. See [Section 23.2, "Configuring Associated Tables"](#).

**234: The primary keys do not match parent's primary keys.**



**Cause:** In an inheritance hierarchy, the child's primary key(s) does not match the root's primary key(s).

**Action:** Ensure that each child's **Primary Key** on the **Descriptor Info** tab matches the parent's primary key. Ensure that the parent and child class have the same primary keys on the descriptor's **Descriptor Info** tab. See [Section 23.2, "Configuring Associated Tables"](#).

**235: The following primary field does not have writable mappings: [field name].**

**Cause:** You attempted to have multiple mappings write to the same database field.

**Action:** Ensure that each database field has a single, writable mapping. See [Chapter 23, "Configuring a Relational Descriptor"](#).

**236: No sequence field is selected.**

**Cause:** You selected **Use Sequencing** on a descriptor's **Descriptor Info** tab, but did not specify the sequence information.

**Action:** Specify a **Name, Table, and Field**. See [Section 23.3, "Configuring Sequencing at the Descriptor Level"](#).

**237: No sequence name is selected.**

**Cause:** You selected **Use Sequencing** on a descriptor's **Descriptor Info** tab but did not specify the sequence information.

**Action:** Specify a **Name, Table, and Field**. See [Section 23.3, "Configuring Sequencing at the Descriptor Level"](#).

**238: No sequence table is selected.**

**Cause:** You selected **Use Sequencing** on a descriptor's **Descriptor Info** tab but did not specify the sequence information.

**Action:** Specify a **Name, Table, and Field**. See [Section 23.3, "Configuring Sequencing at the Descriptor Level"](#).

**239: The selected sequence table is not one of the descriptor's associated tables.**

**Cause:** The table used for sequencing is not associated with the descriptor.

**Action:** You must either associate the sequencing table with the descriptor, or select a table that is already associated with the descriptor. See [Section 23.3, "Configuring Sequencing at the Descriptor Level"](#) and [Section 23.2, "Configuring Associated Tables"](#).

**240: Two queries [query name1] [query name2] cannot have the same signature.**

**Cause:** Two queries for this descriptor share the same signature (query name + parameter names). This is not allowed.

**Action:** You must either remove one of the queries, rename one of the queries, or change the parameters so that the signatures no longer match.

**241: The query [query name] has Cache Statement set to true, but does not bind parameters.**

**Cause:** A named query that caches statements does not bind parameters. It must do so.

**Action:** On the **Named Queries – Options** tab, either change the **Cache Statements** field to **False**, or change the **Bind Parameters** field to **True**. See [Section 119.7.1.9, "Configuring Named Query Options"](#).

- 242: The query *[query name]* does not maintain cache but does refresh the remote identity map results.**  
**Cause:** The query has **Refresh Remote Identity Map** selected, but does not have **Maintain Cache** selected.  
**Action:** You must either select **Maintain Cache** for the descriptor, or deselect **Refresh Remote Identity Map**. See [Section 119.7.1.9, "Configuring Named Query Options"](#).
- 243: The query *[query name]* does not maintain cache but does refresh the identity map results.**  
**Cause:** The query has **Refresh Identity Map** selected but does not have **Maintain Cache** selected.  
**Action:** You must either select **Maintain Cache** for the descriptor, or deselect **Refresh Identity Map**. See [Section 119.7.1.9, "Configuring Named Query Options"](#).
- 245: The query *[query name]* refreshes identity map results but does not refresh remote identity map results.**  
**Cause:** **Refresh Identity Map Results** is selected for the query, but **Refresh Remote Identity Map Results** is not.  
**Action:** You must either select **Refresh Remote Identity Maps** or deselect **Refresh Identity Maps**. See [Section 119.7.1.9, "Configuring Named Query Options"](#).
- 246: The query key *[query key]* does not have an associated database field.**  
**Cause:** The query key is missing an associated database field. Each query key must be associated with a database field.  
**Action:** On the **Query Keys** tab, use the **Field** option to select a database field for the query key. See [Section 119.10, "Configuring Query Keys"](#).
- 247: The database field selected for query key *[query key]* does not exist on this descriptor's associated tables.**  
**Cause:** The database field selected for this query key does not exist on this descriptor's associated tables. Each database field associated with a query key must exist on database table associated with the query key's descriptor.  
**Action:** You must either change the database field associated with the query key, or associate the descriptor with a database table that includes the database field associated with the query key. See [Section 119.10, "Configuring Query Keys"](#).
- 248: The expression *[line number]* on query *[query name]* is invalid because a parameter has not been specified.**  
**Cause:** One of the arguments in the query expression is missing or invalid.  
**Action:** Edit the query and ensure that all query keys and parameters have been specified. See [Section 119.10, "Configuring Query Keys"](#).
- 249: The expression *[line number]* on query *[query name]* is invalid because a query key has not been specified.**  
**Cause:** One of the arguments in the query expression is missing or invalid.  
**Action:** Edit the query and ensure that all query keys and parameters have been specified. See [Section 119.10, "Configuring Query Keys"](#).
- 250: The expression *[line number]* on query *[query name]* is invalid because the chosen query key is not a valid mapping type in an expression.**  
**Cause:** One of the arguments in the query expression is invalid.

- Action:** Edit the query and ensure that all query keys and parameters have been specified. See [Section 119.10, "Configuring Query Keys"](#).
- 251: The expression [line number] on query [query name] is invalid. When querying on a reference mapping, only unary operators (Is Null, Not Null) are supported.**  
**Cause:** You created an expression node that includes a reference mapping with an invalid operator.  
**Action:** On the **Expression Builder** dialog box, select the node and change the **Operator** field to **IS NULL** or **NOT NULL**.
- 252: The query [query name] has no attribute chosen for the ordering attribute at index [index].**  
**Cause:** The ordering attribute is missing from the query.  
**Action:** Edit the query and add an ordering attribute. See [Section 119.7.1.4, "Configuring Read All Query Order"](#).
- 253: The ordering attribute {0} for query {1} is not valid. ReadAllQuery ordering items must be either query keys or direct-to-field mappings.**  
**Cause:** The ordering attribute is invalid.  
**Action:** Edit the query and ensure that ordering attribute is a query key or has a direct-to-field mapping. See [Section 119.7.1.4, "Configuring Read All Query Order"](#).
- 254: The query {0} has no attribute chosen for the joined attribute at index {1}.**  
**Cause:** The joined attribute is missing from the query.  
**Action:** Edit the query and add a joined attribute. See [Section 119.7.1.5, "Configuring Named Query Optimization"](#).
- 255: The joined attribute {0} for query {1} is not valid. Joined attributes must be 1-1, 1-m, m-m, direct collection, or aggregate collection mappings.**  
**Cause:** The joined attribute is invalid.  
**Action:** Edit the query and ensure that joined attribute has a one-to-one, one-to-many, many-to-many, direct collection, or aggregate collection mapping. See [Section 119.7.1.5, "Configuring Named Query Optimization"](#).
- 256: The query {0} has no attribute chosen for the batch read attribute at index {1}.**  
**Cause:** The batch read attribute is missing from the query.  
**Action:** Edit the query and add a batch read attribute. See [Section 119.7.1.5, "Configuring Named Query Optimization"](#).
- 257: The batch read attribute {0} for query {1} is not valid. Batch read attributes must be 1-1, 1-m, m-m, direct collection, aggregate collection, or direct-to-field mappings.**  
**Cause:** The batch read attribute is invalid.  
**Action:** Edit the query and ensure that batch attribute has a one-to-one, one-to-many, many-to-many, direct collection, aggregate collection or direct-to-field mapping. See [Section 119.7.1.5, "Configuring Named Query Optimization"](#).
- 258: The query {0} has no attribute chosen for the grouping attribute at index {1}.**  
**Cause:** The grouping attribute is missing from the query.  
**Action:** Edit the query and add a grouping attribute. See [Section 119.7.1.7, "Configuring Named Query Group/Order Options"](#).

- 259: The query {0} has no attribute chosen for the report attribute {1}.**  
**Cause:** The report query attribute is missing from the query.  
**Action:** Edit the query and add a report query attribute. See [Section 119.7.1.7, "Configuring Named Query Group/Order Options"](#).
- 260: The report attribute {0} for query {1} is not valid. Report query attributes must be either query keys or direct mappings.**  
**Cause:** The report attribute is invalid.  
**Action:** Edit the query and ensure that report attribute is a query key or has a direct-to-field mapping. See [Section 119.7.1.7, "Configuring Named Query Group/Order Options"](#).
- 262: The format for {2} must be between 0 and 127 inclusive. Literal argument of expression (line {0}) on query {1} is invalid.**  
**Cause:** The literal value of the expression is invalid.  
**Action:** Edit the literal value of second argument in Expression Builder. The literal value of a Byte type must be between 1 and 127. See [Section 110.7, "Creating an Expression"](#).
- 263: The format for {2} must be either 'true' or 'false'. Literal argument of expression (line {0}) on query {1} is invalid.**  
**Cause:** The literal value of the expression is invalid.  
**Action:** Edit the literal value of Second Argument in Expression Builder. The literal value of a Boolean type must be *true* or *false*. See [Section 110.7, "Creating an Expression"](#).
- 264: The format for {2} must be a single character. Literal argument of expression (line {0}) on query {1} is invalid.**  
**Cause:** The literal value of the expression is invalid.  
**Action:** Edit the literal value of second argument in Expression Builder. The literal value of a Character type must be a single character. See [Section 110.7, "Creating an Expression"](#).
- 265: The format for {2} must be between {3} and {4}. Literal argument of expression (line {0}) on query {1} is invalid.**  
**Cause:** The literal value of the expression is invalid.  
**Action:** Edit the literal value of second argument in Expression Builder. The literal value must be between 3 and 4. See [Section 110.7, "Creating an Expression"](#).
- 266: The format for {2} must be a string. Literal argument of expression (line {0}) on query {1} is invalid.**  
**Cause:** The literal value of the expression is invalid.  
**Action:** Edit the literal value of second argument in Expression Builder. The literal value of a String type must be a string. See [Section 110.7, "Creating an Expression"](#).
- 267: The format for {2} must contain only digits, '-', and '!'. Literal argument of expression (line {0}) on query {1} is invalid.**  
**Cause:** The literal value of the expression is invalid..  
**Action:** Edit the literal value of second argument in Expression Builder. The literal value of a Double or Float type must contain digits, '-', and '!'. See [Section 110.7, "Creating an Expression"](#).

**268: The format for {2} must contain only digits, '-', and '.'. Literal argument of expression (line {0}) on query {1} is invalid.**

**Cause:** The literal value of the expression is invalid.

**Action:** Edit the literal value of second argument in Expression Builder. The literal value of a Double or Float type must contain digits, '-', and '.'. See [Section 110.7, "Creating an Expression"](#).

**269: The format for {2} must be in the format YYYY/MM/DD or YYYY-MM-DD. Literal argument of expression (line {0}) on query {1} is invalid.**

**Cause:** The literal value of the expression is invalid.

**Action:** Edit the literal value of second argument in Expression Builder. The literal value of a Date type must be in YYYY/MM/DD or YYYY-MM-DD format. See [Section 110.7, "Creating an Expression"](#).

**270: No schema context is specified.**

**Cause:** Each descriptor in an XML or EIS project must be associated with an XML schema context.

**Action:** Select the EIS or XML descriptor in the **Navigator** and complete the **Schema Context** field on the **Descriptor Info** tab.

**271: The descriptor represents a document root object, but no default root element is chosen.**

**Cause:** Each root descriptor must have a default root element.

**Action:** On the descriptor's **Descriptor Info** tab, complete the **Default Root Element** field.

**280: A descriptor that represents \"anyType\" cannot support inheritance.**

**Cause:** The descriptor was supporting inheritance.

**Action:** Edit the descriptor properties and remove the inheritance support.

**281: A descriptor that represents \"anyType\" may contain only a single Any (Object or Collection) mapping.**

**Cause:** The descriptor was supporting more than one Any (Object or Collection) mapping.

**Action:** Edit the descriptor properties and ensure that descriptor supports only one Any (Object or Collection) mapping.

**282: A default root element type has been selected and the default root element is not. Either select a default root element or clear the default root element type**

**Cause:** The default root element was not selected.

**Action:** Select the default root element or clear the default root element type. See [Section 52.4, "Configuring Default Root Element"](#).

**290: No primary keys specified.**

**Cause:** Although you have associated the descriptor with a database table, you have not identified the primary keys.

**Action:** Use the **Primary Keys** area of the descriptor's **Descriptor Info** tab to select the primary keys for the descriptor.

**291–304: The event policy's [method type] method is no longer a visible member of this descriptor's associated class.**

**Cause:** You changed the class hierarchy within the project, causing the method to no longer be visible to the class.

**Action:** Ensure that the selected method is visible to the class.

**305: The write-lock field is stored in an object, but there is not a writable mapping to the field.**

**Cause:** If the write lock field is stored in object, there must be a non-read-only mapping to it.

**Action:** On the mapping's **General** tab, ensure that **Read-Only** is *not* selected.

**306: Database fields specified for Selected Fields type Locking Policy must be mapped: [field name]**

**Cause:** You selected an unmapped database field for a descriptor's locking policy.

**Action:** On the descriptor's **Locking** tab, ensure that you have selected a mapped database field as the **Selected Field**. See [Section 119.26, "Configuring Locking Policy"](#).

**307: Database fields specified for Selected Fields type Locking Policy must not be primary key fields: [field name]**

**Cause:** The database fields you selected for the optimistic locking policy (by fields) contains the primary keys for the database table.

**Action:** In the **By Fields** area of the descriptor's **Locking** tab, select different fields. See [Section 119.26, "Configuring Locking Policy"](#).

**308: Version locking is chosen as the Locking Policy, but the field is not specified.**

**Cause:** If you select to use version locking with an optimistic locking policy, you must identify which database field to use for version control.

**Action:** Use the **Database Field** field on the descriptor's **Locking** tab to select a field to use for version control. See [Section 119.26, "Configuring Locking Policy"](#).

**309: The Version Locking database field selected does not exist on this descriptor's associated tables.**

**Cause:** The database field you selected for optimistic version locking does not exist on the descriptor's associated table.

**Action:** You must either select a different database field on the descriptor's **Locking** tab, or associate the descriptor with a different database table. See [Section 119.26, "Configuring Locking Policy"](#).

**310: Database fields specified for Selected Fields type Locking Policy do not exist on this descriptor's associated tables: [field name]**

**Cause:** The database fields you selected for the optimistic locking policy (by fields) do not exist on the descriptor's associated table.

**Action:** You must either select a different database field on the descriptor's **Locking** tab, or associate the descriptor with a different database table. See [Section 119.26, "Configuring Locking Policy"](#).

**311: The method you have specified for the instantiation policy's method on this descriptor is no longer a visible member of this class.**

**Cause:** The method selected as the instantiation method has either been removed, or its visibility has been reduced so that it is no longer publicly visible.

**Action:** Deselect this method as the instantiation method. See [Section 119.26, "Configuring Locking Policy"](#).

- 312: The method you have specified for the instantiation policy's factory instantiation method on this descriptor is no longer a visible member of this class.**  
**Cause:** The method selected as the factory instantiation method has either been removed, or its visibility reduced so that it is no longer publicly visible.  
**Action:** Deselect this method as the factory instantiation method. See [Section 119.28, "Configuring Instantiation Policy"](#).
- 313: The method you have specified for the instantiation policy's factory method on this descriptor is no longer a visible member of this class.**  
**Cause:** The method selected as the factory method has either been removed, or its visibility reduced so that it is no longer publicly visible.  
**Action:** Deselect this method as the factory method. See [Section 119.28, "Configuring Instantiation Policy"](#).
- 314: "Use factory" is specified for the Instantiation policy, but all required information is not specified.**  
**Cause:** You selected the **Use Factory** option on the descriptor's **Instantiation Policy** tab, but did not specify the **Factory Class**, **Factory Method**, or **Instantiation Method** fields.  
**Action:** Complete the **Factory Class**, **Factory Method**, or **Instantiation Method** fields on the descriptor's **Instantiation** tab. See [Section 119.28, "Configuring Instantiation Policy"](#).
- 315: "Use method" is selected for the Instantiation policy, but no method is selected.**  
**Cause:** You selected the **Use Method** option on the descriptor's **Instantiation Policy** tab, but did not specify the field.  
**Action:** Select the **Method** on the descriptor's **Instantiation** tab. See [Section 119.28, "Configuring Instantiation Policy"](#).
- 316: The class does not have an accessible zero argument constructor.**  
**Cause:** No accessible zero argument constructor exists for the class associated with this descriptor.  
**Action:** Make the zero argument constructor accessible if it exists, or create a accessible zero argument constructor if it doesn't exist.
- 317: No method was specified for the copying policy.**  
**Cause:** You specified that the descriptor should use a specific clone method for copying, but you did not select a method.  
**Action:** Complete the **Use Clone Method** field on the descriptor's **Copying** tab to select a method.
- 318: The method specified for the copy policy on this descriptor is no longer a visible member of this class.**  
**Cause:** You changed the class hierarchy within the project, causing the copy policy to no longer be visible to the class.  
**Action:** Ensure that the copy policy is visible to the class.
- 319: Primary keys do not match across associated tables and no reference(s) specified in multiple table policy information.**  
**Cause:** You attempted to associate multiple tables using a primary key.

**Action:** Primary key field names must match across associated tables, or references must be defined from the base table to each derived table.

**320: The multiple table reference should be defined from the base table [table name] to the derived table.**

**Cause:** This descriptor has **Inheritance** and **Multitable** advanced properties defined on it.

**Action:** The multiple table relationship that is defined between the base class' table and this derived class' table must be defined from base to derived.

**321: The multiple table reference should not be defined on the database.**

**Cause:** When using multitable with differently named primary keys, you must set a reference from the TOP table to the BOTTOM table. This reference must not be an actual constraint on the database.

**Action:** Select the table in which this is defined, and deselect the **On Database** option.

**322: A class containing the desired after loading method should be specified.**

**Cause:** You added an after-load method to a descriptor, but you did not specify a class.

**Action:** Complete the **After Load** tab. See [Section 119.35, "Configuring Amendment Methods"](#).

**323: An after-load method must be specified.**

**Cause:** You added an after-load method to a descriptor, but did not select an amendment method.

**Action:** Complete the **After Load** tab. See [Section 119.35, "Configuring Amendment Methods"](#).

**324: An interface class must be specified for the interface alias.**

**Cause:** You added an interface alias to a descriptor, but did not select an amendment method.

**Action:** Complete the **Interface Alias** tab.

**325: The inheritance hierarchy originating in this descriptor cannot contain both aggregate and nonaggregate child descriptors.**

**Cause:** Aggregate and class descriptors cannot be in the same inheritance hierarchy.

**Action:** Ensure that the inheritance hierarchy contains either aggregate *or* nonaggregate children, but not both.

**326: The inheritance hierarchy originating in this descriptor cannot contain both root and composite child descriptors.**

**Cause:** There is a mixture of root and composite descriptors among the descendents of this descriptor.

**Action:** Make all descendents of this descriptor the same type by either making them all root, or making them all composite. You can do this by removing the differing descriptor from the hierarchy, or changing their type to be consistent with the other descriptors in the hierarchy.

**330: The returning policy insert fields do not exist on this descriptor's associated tables: [field name]**



**Cause:** The field you selected on the descriptor's **Returning** tab does not exist on the database table associated with the descriptor.

**Action:** Select a different database table in the **Insert** area of the descriptor's **Returning** tab.

**331: The returning policy update field [field name] does not exist on this descriptor's associated tables.**

**Cause:** The field you selected on the descriptor's **Returning** tab does not exist on the database table associated with the descriptor.

**Action:** Select a different database table in the **Update** area of the descriptor's **Returning** tab.

**350: Descriptors with Unknown Primary Keys must use sequencing.**

**Cause:** **Unknown Primary Key Class** is selected for this descriptor, but the descriptor does not use sequencing.

**Action:** Change the descriptor so that it uses sequencing, or so that it no longer uses an unknown primary key class.

### A.3.4 Mapping Errors (400 – 483)

This section lists TopLink Workbench mapping errors.

**400: Method accessors have not been selected.**

**Cause:** You selected **Use Method Accessing** for a mapping, but you did not select a method.

**Action:** You must select a **Get** and **Set** method on the mapping's **General** tab. See [Section 121.6, "Configuring Method or Direct Field Accessing at the Mapping Level"](#).

**401, 402: The [get/set access method] method for this mapping's method accessing field is no longer visible to this descriptor.**

**Cause:** You changed the class hierarchy within the project, causing the method access type (`get` or `set`) to no longer be visible to the class.

**Action:** Ensure that the selected method is visible to the class.

**403: Mappings for EJB 2.0 CMP descriptors that use Value Holder Indirection must not use method accessing.**

**Cause:** You cannot use method accessing on mappings for EJB 2.0 CMP descriptors that use **ValueHolder Indirection**.

**Action:** Because EJB attributes are code-generated, reference mappings *should not* be set to use method access. The attributes are code-generated to be of type **ValueHolder** but the abstract methods are defined to return the local interface type of the related bean.

**404: Mapping references a write-lock field, but it is not read-only.**

**Cause:** You specified a locking policy for a descriptor, but one of the attribute mappings is not read-only.

**Action:** Select the **Read Only** option on the mapping's **General** tab.

**410: No direct field is specified.**

**Cause:** For direct collection mappings, you must specify the direct collection information.

**Action:** Select a **Target Table** and **Direct Field** that the direct collection specifies.

**415: No direct key field is specified.**

**Cause:** For direct map mappings, you must specify a direct key field in the reference table that stores the primitive data value of the map key.

**Action:** On the direct map mapping's **General** tab, select a **Direct Key Field**. See [Section 38.3, "Configuring Direct Key Field"](#).

**420: No database field is selected.**

**Cause:** You created a direct-to-field or type conversion mapping without selecting a database field.

**Action:** For attributes with direct-to-field mappings, you must specify a **Database Field** on the mapping's **General** tab. For attributes with type conversion mappings, you must specify a **Database Field** on the mapping's **General** tab.

**421: The selected database field does not exist on this descriptor's associated tables.**

**Cause:** The database field mapped to an attribute is not included in the table associated with the attribute's descriptor.

**Action:** Ensure that the **Database Field** field on a mappings **General** tab is included in the table that you associated with the attribute's descriptor. See [Section 23.2, "Configuring Associated Tables"](#) and [Section 28.3, "Configuring a Database Field"](#).

**430, 431: No null value type has been selected.**

**Cause:** You selected to **Use Default Value When Database Field is Null** for a mapping, but did not specify the value.

**Action:** Specify a default **Type** or **Value**, or both on the mapping's **General** tab. See [Section 121.5, "Configuring a Default Null Value at the Mapping Level"](#).

This message may also appear after using the Package Rename tool when upgrading an older TopLink Workbench project.

**440: XML type mappings are supported only on the Oracle9i Platform.**

**Cause:** You created a Direct to XML Type mapping in relational project that uses a non-Oracle9i database.

**Action:** Select an Oracle9i platform as the database platform for the data source. See [Section 20.2, "Configuring Relational Database Platform at the Project Level"](#).

**450: No reference descriptor is selected.**

**Cause:** You created a mapping, but did not specify the reference descriptor

**Action:** You must select a **Reference Descriptor** for each relationship mapping on the mapping's **General** tab.

**451: [*descriptor name*]references [*descriptor name*], which is not active.**

**Cause:** You tried to select an inactive descriptor as a **Reference Descriptor** on the mapping's **General** tab.

**Action:** You must either select a new **Reference Descriptor**, or make the descriptor active.

**460: No table reference is selected.**

**Cause:** You created a relationship mapping, but did not specify a reference table.

**Action:** Select (or create) a table reference for each relationship mapping on the mapping's **Table Reference** tab.

**461: Table reference is invalid.**

**Cause:** The table reference selected for this mapping is invalid.

**Action:** Select a different table reference for this mapping.

**462: The reference [table reference] does not have any field associations.**

**Cause:** You selected a table reference for a mapping, but did not add a key pair.

**Action:** You must specify source and target key pairs for the reference.

**463: A key pair has not been completely specified for a reference.**

**Cause:** You created a table reference without a key pair.

**Action:** You must specify a foreign key reference for the database table. Use the database table's **Reference** tab to add a key pair.

**464: No relationship partner is specified.**

**Cause:** You selected the **Maintains Bidirectional Relationship** option for a relationship mapping, but did not select a mapping to use as the relationship partner.

**Action:** Select a mapped attribute (from the reference descriptor) for this relationship. See [Section 121.18, "Configuring Bidirectional Relationship"](#).

**465: The relationship partner must be a one-to-one, one-to-many, or many-to-many mapping.**

**Cause:** You selected an invalid attribute as the Relationship Partner in a bidirectional relationship.

**Action:** In the **Relationship Partner** field, select a one-to-one, one-to-many, or many-to-many mapping. See [Section 121.18, "Configuring Bidirectional Relationship"](#).

**466: The specified relationship partner mapping does not specify this mapping as its own relationship partner.**

**Cause:** **Maintains Bidirectional Relationship** is selected for this mapping, but the mapping selected as the relationship partner does not have this mapping selected as its relationship partner.

**Action:** You must either select a different mapping for this mappings relationship partner, which has this mapping selected as it bidirectional relationship partner, or select this mapping as the bidirectional relationship partner of the mapping selected as the bidirectional relationship partner for this mapping.

**467: The chosen reference descriptor is not a valid reference descriptor for this mapping.**

**Cause:** The descriptor selected as the reference descriptor for this mapping is not a valid reference descriptor.

**Action:** Select a valid reference descriptor for this mapping.

**470: No container class is selected.**

**Cause:** No container class has been selected for this collection mapping.

**Action:** Select a `Container` class for this `Collection` mapping.

**471: The container policy uses a Collection class, but the container class is not a Collection.**

**Cause:** The selected container class for this collection mapping is not a `Collection`, but **Use Collection Class** is selected.

**Action:** Select a `Container` class that is a `Collection` for this mapping.

**472: The container policy uses a Map class, but the container class is not a Map.**

**Cause:** The selected Container class for this Collection mapping is not a Map class, but **Use Map Class** is selected.

**Action:** Select a Container class that is a Map class.

**473: The container class must be instantiable.**

**Cause:** The selected Container class for this Collection mapping is not instantiable.

**Action:** Select a Container class this is instantiable, (not an Interface, Abstract class, or Primitive class).

**474: The container class does not agree with the instance variable.**

**Cause:** The selected Container class for this Collection mapping, does not agree with the instance variable that is associated with the mapping. Either the variable is a Map class and the selected Container class is a Collection or vice versa.

**Action:** You must either select a Container class that agrees with the type of instance variable with which it is associated, or change the instance variable to agree with the selected Container class.

**475: The container class is a Map, but the key method is not selected.**

**Cause:** **Use Map Class** is selected for the Container policy for this Collection mapping, but a key method has not been selected.

**Action:** You must either select a key method for this Container policy, or change the Container policy to not use a map class.

**476: The key method specified for this mapping is no longer visible to the owning descriptor's class.**

**Cause:** The selected key method for the Container policy for this Collection mapping policy is not visible to the descriptor's class.

**Action:** You must either select a different method that is visible to the descriptor's class, or change the selected method so that it is visible.

**477: The key method specified for this mapping is not valid.**

**Cause:** The selected key method for the Container policy for this Collection mapping is invalid because it does not have the correct return type, or it does not accept more than zero parameters.

**Action:** You must either select a different method that is valid, or change the selected method so that it will return the correct type and accept more than zero parameters.

**478: One-to-Many and Many-to-Many mappings in EJB 2.0 CMP descriptors may not use ValueHolder indirection.**

**Cause:** A one-to-many or many-to-many mapping in an EJB 2.0 CMP descriptor is using ValueHolder indirection.

**Action:** You must either change the mapping to use no indirection or non-ValueHolder indirection.

**480: No relation table is selected.**

**Cause:** You created a many-to-many mapping, but did not specify a relation table. The relation table represents the relationship between the primary keys of the source table and target table.

**Action:** Select or create a **Relation Table** on the mapping's **General** tab.

**481: The relation table is not dedicated to single, writable many-to-many mapping.**

**Cause:** More than one many-to-many mapping in the project are using the same relation table.

**Action:** Each relation table should be used in one and only one many-to-many mapping.

**482: No source reference is selected.**

**Cause:** You created a many-to-many mapping, but did not select (or create) a source table reference on the mapping's **Source Reference** tab.

**Action:** The source table reference must contain a **Source** field (from the mapping's relation table) and a **Target** field (from one of the descriptor's associated tables).

**483: No target reference is selected.**

**Cause:** You created a many-to-many mapping, but did not select (or create) a target table reference on the mapping's **Source Reference** tab.

**Action:** The target table reference must contain a **Source** field (from the mapping's relation table) and a **Target** field (from one of the descriptor's associated tables).

### A.3.5 Table Errors (500 – 610)

This section lists TopLink Workbench table errors.

**500: You cannot use joining because the source and target (reference) descriptors are the same type.**

**Cause:** You selected the **Use Joining** option on a one-to-one mapping in which the source and reference descriptors are the same.

**Action:** You must either deselect the **Use Joining** option or select a difference **Reference Descriptor** on the **One-to-One Mapping General** tab.

**510: No query key associations have been defined.**

**Cause:** You created a variable one-to-one mapping, but did not define a key pair.

**Action:** Create or select a key pair on the mapping's **Query Key Association** tab.

**511: Not all query key associations have foreign key fields specified.**

**Cause:** You created a query key association without a foreign key.

**Action:** You must specify a foreign key field for each query key association on the **Query Key Association** tab for variable one-to-one mapping.

**512: The following specified query key names are no longer valid: *[query key]***

**Cause:** The query keys listed for this mapping no longer refer to the reference descriptor for this mapping. The query keys are now invalid.

**Action:** You must either remove the invalid query keys, or change the reference descriptor so that it corresponds with the query keys.

**513: No indicator field is selected.**

**Cause:** You created a variable one-to-one mapping, but did not specify a database field in which to store indicator values.

**Action:** Select the **Class Indicator Field** on the **Class Indicator Info** tab.

**514: No indicator values are specified.**

**Cause:** You created a variable one-to-one mapping, but did not specify indicator values for each object type.

**Action:** Select the **Indicator Type** on the **Class Indicator Info** tab.

**515: *[descriptor name]* is not an implementor of the *[descriptor name]* interface, so it cannot have an indicator value.**

**Cause:** You included a descriptor on the **Variable One-to-One Class Indicator Info** tab that is an implementor.

**Action:** Deselect the descriptor on the **Variable One-to-One Class Indicator Info** tab or add the descriptor to the **Implementor** tab.

**516: The chosen reference descriptor is not an interface descriptor.**

**Cause:** This variable one-to-one mapping has a reference descriptor selected which is not an interface descriptor. The reference descriptor for a variable one-to-one mapping must be an interface descriptor for the mapping to be valid.

**Action:** You must either choose a reference descriptor that is an interface descriptor, or change the mapping to no longer be variable.

**520: No attribute transformer is specified.**

**Cause:** No attribute transformer is specified for this transformation mapping.

**Action:** Select an attribute transformer for this transformation mapping.

**521: The attribute transformer class is missing.**

**Cause:** No class has been specified for the attribute transformer for this transformation mapping.

**Action:** Select a class for the attribute transformer.

**522: The attribute transformer class *[class name]* is not a valid transformer class.**

**Cause:** The attribute transformer class that is selected is not a valid attribute transformer class.

**Action:** Select a valid attribute transformer class for the transformation mapping.

**523: The attribute transformer method is missing.**

**Cause:** No method has been selected for the attribute transformer for the transformation mapping.

**Action:** Select a method for the attribute transformer.

**524: The attribute transformer method *[method name]* is not visible to the parent descriptor's class.**

**Cause:** The selected attribute transformer method is not visible to the descriptor class for this mapping.

**Action:** You must either select a different method that is visible, or change the method in the class to make it visible.

**525: The attribute transformer method *[method name]* is not a valid transformer method.**

**Cause:** The selected attribute transformer method either has the wrong return type or accepts the wrong parameters to be a valid transformer method for this transformation mapping.

**Action:** You must either select a method with the correct return type and parameters, or change the selected method so that it meets these criteria.

**526: No field transformer associations are specified.**

**Cause:** No field transformer association has been specified for this transformation mapping.

**Action:** Specify at least one field transformer association.

**527: No transformer is specified for the field [field name].**

**Cause:** No transformer specified for the given field.

**Action:** Specify a transformer for this field.

**528: There is a missing field in the field transformer association.**

**Cause:** There is no field specified for a field transformer association for this transformation mapping.

**Action:** Specify a field for all the field transformer associations for this transformation mapping.

**529: There is a missing transformer class for the field [field name].**

**Cause:** The `Transformer` class is specified for this field transformer association, but the `Transformer` class is unspecified.

**Action:** Specify a `Transformer` class for the field transformer association for this field.

**530: The transformer class [class name] for the field [field name] is not a valid transformer class.**

**Cause:** The specified `Transformer` class for the field of this field transformer association is invalid.

**Action:** Specify a valid `Transformer` class for the field transformer association for this transformation mapping.

**531: There is a missing transformer method for the field [field name].**

**Cause:** A transformer method is specified for this field transformer association, but the transformer method is unspecified.

**Action:** Specify a transformer method for the field transformer association for this field.

**532: The transformer method [method name] for the field [field name] is not visible to the parent descriptor's class.**

**Cause:** The specified transformer method for the field transformer association for this field is not visible to the descriptor or the class of this mapping.

**Action:** You must either choose a method that is visible to the class, or change the method so that it is visible.

**533: The field transformer method [method name] for the field [field name] is not a valid transformer method.**

**Cause:** The specified method for the field transformer association for this field either has the incorrect return type, or accepts the wrong parameters.

**Action:** You must either select a method that has the correct return type and parameters, or change the currently selected method so that it has the correct return type and parameters.

**540: No object type is selected.**

**Cause:** You created an object type mapping, but did not select the type.

**Action:** You must select the **Object Type** and **Database Type** on the **General** tab of the mapping.

**542: No object-type mappings have been specified.**

**Cause:** You created an object type mapping, but did not create an object-to-database mapping.

**Action:** You must specify at least one mapping (**Database Value** and **Object Value**) on the **General** tab of the mapping.

**545: NCharacter, NString, and NClob database types are currently supported only on the Oracle9i platform.**

**Cause:** You attempted to map a database type that is not supported by your database.

**Action:** The database type for a type conversion mapping or direct-to-field mapping can be `NCharacter`, `NString`, or `NCLOB` only if you are using an Oracle9i database.

**550: Attribute is typed as a ValueHolderInterface, but the mapping does not use Value Holder Indirection.**

**Cause:** You did not specify indirection or transparent indirection for the mapping.

**Action:** If the class attribute is of type `ValueHolderInterface`, you must use `ValueHolder` indirection for the mapping.

**551: Mapping uses ValueHolder Indirection, but its associated attribute is not a ValueHolderInterface.**

**Cause:** You selected indirection without a `ValueHolderInterface`.

**Action:** If you select the **Use Indirection (ValueHolder)** option for a one-to-many, many-to-many, or direct collection mapping, the associated class attribute must be `ValueHolderInterface`.

**560: The container class for this mapping must implement oracle.toplink.indirection.IndirectContainer.**

**Cause:** This mapping uses transparent indirection, but the `Container` class selected for its container policy is not an `IndirectContainer`.

**Action:** You must either select a `Container` class that is an `IndirectContainer`, or remove transparent indirection from the mapping.

**570: The chosen reference descriptor is not an aggregate descriptor.**

**Cause:** This is an aggregate mapping, but the selected reference descriptor is not an aggregate descriptor.

**Action:** You must either select a reference descriptor for this mapping that is an aggregate descriptor, or change this mapping to no longer be an aggregate mapping.

**571: Aggregate fields are not specified.**

**Cause:** You created an aggregate mapping without specifying specific fields.

**Action:** Every **Field Description** on the **Fields** tab must contain a unique **Field** for aggregate mappings.

**572: Aggregate mapping fields must be unique.**

**Cause:** You created an aggregate mapping without specifying unique fields.

**Action:** Every **Field Description** on the **Fields** tab must contain a unique **Field** for aggregate mappings.



**573: The selected field does not exist on this descriptor's associated tables.**

**Cause:** The field selected for one of the aggregate-path-to-fields for this aggregate mapping does not exist on any of the descriptor's associated tables.

**Action:** You must either select a different field for the path-to-field, or add the field to the appropriate table.

**580: No XML field specified.**

**Cause:** You mapped an attribute in an XML or EIS descriptor, but did not select an XML field.

**Action:** You must complete the **XML Field** field on the **General** tab of the mapping.

**581: The specified XPath is not valid within the current schema.**

**Cause:** The XPath specified for this mapping does not resolve in the schema.

**Action:** You must either select a different XPath, or alter the schema so that this XPath will resolve.

**582: The specified XPath does not represent text data.**

**Cause:** The XPath specified for this direct mapping does not resolve to a direct field in the schema.

**Action:** You must either select a different XPath, alter the schema so that this XPath will resolve to a direct field, or change the mapping type.

**583: The specified XPath does not represent a single xml field.**

**Cause:** The XPath specified for this mapping resolves to a field which is a collection, but this is not a collection mapping.

**Action:** You must either select a different XPath, alter the schema so that this XPath will resolve to a singular field, or change the mapping type.

**590: The chosen reference descriptor is not a root eis descriptor.**

**Cause:** The reference descriptor selected for this EIS reference mapping is not a root descriptor. Reference mappings in EIS descriptors must be root descriptors.

**Action:** You must either select a different reference descriptor for this mapping which is a `root` descriptor, or change the mapping type.

**591: No relationship partner is specified.**

**Cause:** This mapping has **Maintains Bidirectional Relationship** selected, but no relationship partner is specified.

**Action:** You must either deselect **Maintains Bidirectional Relationship**, or select a relationship partner.

**592: The relationship partner must be an EIS One-to-One or EIS One-to-Many mapping.**

**Cause:** The relationship partner selected for this mapping is not of the type EIS one-to-one or EIS one-to-many.

**Action:** You must select an EIS one-to-one or EIS one-to-many mapping as the relationship partner for this mapping, or deselect **Maintains Bidirectional Relationship**.

**593: The specified relationship partner mapping does not specify this mapping as its own relationship partner.**

**Cause:** The mapping selected as the relationship partner for this mapping does not have this mapping selected as its relationship partner. For these relationships to be bidirectional, you must select the relationship partner for both mappings.

**Action:** You must either go to the mapping selected as the relationship partner for this mapping and select this mapping as its relationship partner, or select a different relationship partner mapping for this mapping to maintain this mapping as its relationship partner.

**594: There is a missing source XML field.**

**Cause:** No field has been specified as the source XML field for this mapping.

**Action:** You must specify a source XML field.

**595: There is a missing target XML field.**

**Cause:** No field has been specified as the target XML field for this mapping.

**Action:** You must specify a target XML field.

**600: A foreign key grouping element is required if there are multiple field pairs.**

**Cause:** No foreign key grouping element is specified for this mapping and multiple field pairs.

**Action:** You must specify a foreign key grouping element.

**601: The foreign key grouping element does not contain all foreign keys fields.**

**Cause:** The specified foreign key grouping element does not contain all the foreign key fields.

**Action:** You must either remove the foreign key fields not contained in this foreign key grouping element, or pick a foreign key grouping element that contains all the foreign key fields.

**602: A delete all interaction is specified, but the mapping is not private owned.**

**Cause:** A `deleteall` interaction is specified for this mapping, but the mapping is not private owned.

**Action:** You must either make the mapping private owned, or remove the `deleteall` interaction.

**610: At least one field pair must be specified, unless the mapping has no selection interaction and is read-only.**

**Cause:** No field pairs are specified, and this mapping has a `selection` interaction specified and/or is not read-only.

**Action:** You must either specify a field pair for the mapping, or make the mapping read-only and remove the `selection` interaction.

### A.3.6 XML Schema Errors (700 – 706)

This section lists TopLink Workbench XML schema errors.

**701: A database table can only have one IDENTITY column defined.**

**Cause:** You defined more than one identity column for this table.

**Action:** On the database table's **Columns** tab, leave only one identity (**Identity**) column. See [Section 5.5.2.1, "Working with Column Properties"](#).

**702: A size is required for the column [column].**

**Cause:** You did not specify any size for this column. The default size is 0.

**Action:** On the database table's **Columns** tab, specify the size (**Size**) for the column (field). See [Section 5.5.2.1, "Working with Column Properties"](#).

**703: The reference [table reference] does not have any field pairs.**

**Cause:** You added a reference for a table, but the reference does not include a key pair.

**Action:** On the database table's **References** tab, specify source and target field pairs for the table reference. See [Section 5.5.2.3, "Creating Table References"](#).

**704: A key pair has not been completely specified for a reference.**

**Cause:** A reference table is missing a complete key pair (source and target fields).

**Action:** You must specify a foreign key reference for the database table. On the database table's **References** tab, add a complete key pair. [Section 5.5.2.3, "Creating Table References"](#).

**705: A development login has not been specified.**

**Cause:** You created a relational TopLink Workbench project, but did not specify a development login.

**Action:** On the Database property sheet, select a **Development Login** from the available defined logins, or add a new login. See [Section 20.5, "Configuring Development and Deployment Logins"](#).

**706: A deployment login has not been specified.**

**Cause:** You created a relational TopLink Workbench project, but did not specify a deployment login.

**Action:** On the Database property sheet, select a **Deployment Login** from the available defined logins, or add a new login. See [Section 20.5, "Configuring Development and Deployment Logins"](#).

### A.3.7 Session Errors (800 – 812)

This section lists the TopLink sessions XML errors.

**801: [session name] Login - The connection URL has to be specified.**

**Cause:** You have not specified a connection URL for the session (when using a database driver manager). Each session must have at least one login connection.

**Action:** On the session's **Login – Connection** tab, complete the **Driver URL** field. See [Section 89.3, "Configuring a Session Login"](#).

**802: [session name] Login - The driver class has to be specified.**

**Cause:** You have not specified a driver class for the session (when using a data source database driver).

**Action:** On the session's **Login – Connection** tab, complete the **Driver Class** field. See [Section 89.3, "Configuring a Session Login"](#).

**803: [session or connection pool name]Login - Login - The data source name has to be specified.**

**Cause:** You have not specified a driver class for the session login (when using a Java EE data source database driver).

**Action:** On the session's or connection pool's **Login – Connection** tab, complete the **Data Source** field. See [Section 89.3, "Configuring a Session Login"](#).

**804: Login - Session Broker - It has to have at least one session, either a server or a database session.**

**Cause:** You created a session broker but did not add any sessions. Each session broker must contain a session.

**Action:** On the session broker's **General – Sessions** tab, select a session to add to this broker. See [Chapter 94, "Configuring Session Broker and Client Sessions"](#).

**805: [session name] Database Session - It has to have at least one XML file or a class specified.**

**Cause:** Your database session does not have a primary project (an associated deployment XML file or Java class file).

**Action:** On the session's **Project – General** tab, complete the **Primary Project** field. See [Section 89.2, "Configuring a Primary Mapping Project"](#).

**806: Login - The transport class has to be specified.**

**Cause:** You selected a custom (user-defined) cache coordination type, but did not specify the transport class for cache coordination.

**Action:** On the session's **Cache Coordination** tab, complete the **Transport Class** field, or select a different cache coordination type. See [Chapter 103, "Configuring a Coordinated Cache"](#).

**807: [session name] Login - The location of the log file has to be specified.**

**Cause:** You are using standard logging and selected to have the log saved to a file, but did not select a file name and location.

**Action:** On the session's **Logging** tab, complete the **Log Location** field. See [Section 89.4, "Configuring Logging"](#).

**811: [session or broker name] - An external transaction controller (JTA) has to be specified.**

**Cause:** You selected a custom server platform, but did not specify the JTA for the platform.

**Action:** On the session or session broker's **General – Server Platform** tab, complete the **External Transaction Controller (JTA)** field. See [Section 89.9, "Configuring the Server Platform"](#).

**812: [session or broker name] - A server class has to be specified.**

**Cause:** You selected a custom server platform, but did not specify the server class for the platform.

**Action:** On the session or session broker's **General – Server Platform** tab, complete the **Server Class** field. See [Section 89.9, "Configuring the Server Platform"](#).

### A.3.8 Common Classpath Problems

The following are some common TopLink Workbench error messages that may result from invalid classpath information. See [Section 117.3, "Configuring Project Classpath"](#) for more information.

**The TopLink Workbench does not display the class(es) to import.**

**Cause:** Your classes are not available for import on the **Select Classes** dialog box.

**Action:** Ensure that the class is in your project's classpath (on the project's **General** properties tab). Ensure that the class is in the `.zip` or `.jar` file. You cannot import compressed classes.

**The TopLink Workbench generates an exception error when importing classes.**

**Cause:** TopLink class import utility did not start correctly. One of the classes includes a static initialization method, which may cause the import utility to fail.

**Action:** Ensure that your project's classpath points to the root directory of your package hierarchy. For example, to import the `com.company.class` package in the `C:\classes\com\company` directory, your project classpath should be `C:\classes\.`

**The TopLink Workbench fails to import the class, but does not generate an exception error.**

**Cause:** The classpath containing your JDBC drivers should still be on your system CLASSPATH. TopLink Workbench classpath is for domain classes only.

**Action:** Ensure that you have properly indicated the directories that contain your domain class(es) to map on the project's **General** tab.

### A.3.9 Database Connection Problems

This section describes common errors and problems you may encounter when communicating with or logging in to the database.

**The class *[class]* was not found.**

**Cause:** You attempted to log in to the database, but TopLink could not find the JDBC driver for the database.

**Action:** Ensure that the `JDBC_CLASSPATH` in the `setenv.cmd` file points to your JDBC driver JAR files. Verify that your `PATH` includes all files (for example, native `.dll` files) required by the driver. If the path to your JDBC driver JAR files contains spaces, then the path must be enclosed in double-quotes in the `setenv.cmd` file. For example:

```
set JDBC_CLASSPATH="C:\Program Files\some directory\driver.jar\"
```

For more information, see [Section 5.2, "Configuring the TopLink Workbench Environment"](#).

**Username or password could be invalid.**

**Cause:** TopLink was unable to log in to the database.

**Action:** Ensure that the **Username** and **Password** for the database are correct. Verify with your DBA that the database is set up and operating correctly.

**You must define a development login.**

**Cause:** You attempted to log in to the database from TopLink Workbench, but you did not define a development login.

**Action:** On the database property sheet, select a **Development Login**, or create a new **Defined Login**. See [Section 20.4, "Configuring Login Information at the Project Level"](#).

**No database driver has been specified.**

**Cause:** You attempted to log in to the database from TopLink Workbench, but you did not complete the login information.

**Action:** Complete all the required fields on the database property sheet for the selected development login. See [Section 20.4, "Configuring Login Information at the Project Level"](#).

**Invalid URL specified.**

**Cause:** You attempted to log in to the database from TopLink Workbench, but the URL is incorrect.

**Action:** Complete the **URL** field on the database property sheet for the selected development login. See [Section 20.4, "Configuring Login Information at the Project Level"](#).

---

---

# Glossary

This glossary contains terms and abbreviations that you should be familiar with when using Oracle TopLink.

## **attribute**

A variable of a class or object. In TopLink, an *attribute* describes all instance variables of a class. Every attribute contains a single mapping. Attributes store primitive data such as integers, and simple Java types such as `String` or `Date`.

## **authentication**

The means by which a data source validates a user's identity and determines whether or not the user has sufficient privileges to perform a given action.

## **bean class**

The implementation of the bean. The bean is accessed from the client using the home and remote interfaces.

## **bean-managed persistence (BMP)**

A scheme for persisting entity beans that requires the developer to manually code the methods that perform the persistence.

Compare to [container-managed persistence \(CMP\)](#).

## **branch class**

Has a persistent superclass and also has subclasses. By default, queries performed on the branch class return instances of the branch class and any of its subclasses. However, the branch class can be configured so that queries on it return only instances of itself without instances of its subclasses.

Compare to [leaf class](#).

## **class**

A category of objects. Classes allow data and method to be grouped together.

## **class indicator field**

A field in the table of the root class that indicates which subclass should be instantiated

## **client session broker**

A collection of client sessions, one from each server session associated with the session broker.

**connection pool**

A collection of reusable connections to a single data source.

**container-managed persistence (CMP)**

A scheme for persisting entity beans that uses information supplied by the developer or deployer to perform the persistence

Compare to [bean-managed persistence \(BMP\)](#).

**custom SQL**

Refers to any non-TopLink-generated SQL used through TopLink. This includes hard-coded SQL and stored procedure calls.

**data definition language (DDL)**

The data definition part of the structured query language (SQL). TopLink Workbench can generate DDL creation scripts that can be used to create tables on the desired database.

**database session**

A database session provides a client application with a single data store connection, for simple, standalone applications in which a single connection services all data store requests for one user.

**default mapping**

A relational persistence framework term that refers to making the framework automatically generate the object descriptor metadata (including such things as mappings, login data, database platform, locking, and foreign keys). Default mapping is available for TopLink projects using EJB 2.0 CMP applications with OC4J.

**dependent class path (IBM WebSphere)**

Location where nonbean classes are specified. TopLink requires that the bean classes be included here since they are referenced by the project.

**deployment descriptor**

A set of XML files that provide the additional required information to install an EJB within its server. Typically, this includes security, transaction, relationship, and persistence information.

Compare with TopLink [descriptors](#).

**descriptors**

An TopLink object that describes how an object's attributes and relationships are to be represented in relational database table(s). An "TopLink descriptor" is not the same as a [deployment descriptor](#), although it plays a similar role.

**direct access**

By default, TopLink accesses public attributes directly when writing the attributes of the object to the database or reading the attributes of the object from the database.

Compare to .

**direct mapping**

There are two basic ways of storing object attributes directly in a table:



- The information can be stored directly if the attribute type is comparable to a database type.
- If there is no database primitive type that is logically comparable to the attribute's type, it must be transformed on its way to and from the database

TopLink provides five classes of direct mappings.

Compare to [relationship mapping](#).

### **Enterprise Java Beans (EJB)**

EJB are server-side domain objects that fit into a standard component-based architecture for building enterprise applications with Java. They are objects that become distributed, transactional, and secure components. TopLink Workbench uses three types of EJB: [session beans](#), [entity beans](#), and [message-driven beans](#).

### **expressions**

The TopLink equivalent of an SQL conditional clause. TopLink expressions are specified using the `Expression` and `ExpressionBuilder` classes.

### **entity beans**

EJB that represent a persistent data object. TopLink uses two schemes for persisting entity beans: [bean-managed persistence \(BMP\)](#) and [container-managed persistence \(CMP\)](#).

### **fetch group**

A performance enhancement that defines a subset of object attributes to be loaded initially and ensures that all other attributes are loaded on demand.

### **hub**

A common connection point for devices in a network.

### **identity map**

Used to cache objects for performance and to maintain object identity.

See also [object identity](#).

### **independent relationship**

A relationship in which the source and target are public objects that exist independently; the destruction of one object does not necessarily imply the destruction of the other.

Compare to [private relationship](#).

### **indirection**

The TopLink term for [lazy loading](#).

By default, when TopLink retrieves a persistent object, it retrieves all of the dependent objects to which it refers. When you configure indirection (also known as [lazy loading](#), [lazy reading](#), and [just-in-time reading](#)) for an attribute mapped with a relationship mapping, TopLink uses an indirection object as a place holder for the referenced object: TopLink defers reading the dependent object until you access that specific attribute. This can result in a significant performance improvement, especially if the application is interested only in the contents of the retrieved object, rather than the objects to which it is related.

TopLink supports a variety of types of indirection, including: **value holder indirection**, **transparent indirect container indirection**, and **proxy indirection**.

**inheritance**

Describes how a child class inherits the characteristics of its parent class. TopLink supports multiple approaches to database implementations that preserve the inheritance relationship.

**in-memory query**

A query that is run against the shared session cache.

**instantiate**

Create an instance of a Java class.

**JCA**

The Java EE Connector architecture (JCA) adapter is a way to persist Java objects to a nonrelational data source, such as XML.

**Java SE**

The Java Platform, Standard Edition (Java SE) is the core Java technology platform. It provides software compilers, tools, runtimes, and APIs for writing, deploying, and running applets and applications in Java.

**Java EE**

The Java Platform, Enterprise Edition (Java EE) is an environment for developing and deploying enterprise applications. Java EE includes a set of services, APIs, and protocols for developing multitiered web-based applications.

**Java EE Containers**

A Java EE container is a run-time environment for Enterprise Java Beans (EJB) that includes such basic functions as security, life cycle management, transaction management, and deployment services. Java EE containers are usually provided by a Java EE server, such as Oracle Containers for Java EE.

**Java Messaging Service (JMS)**

The JMS API is a protocol for communication that provides asynchronous communication between components in a distributed computing environment.

**Java Naming and Directory Interface (JNDI)**

The JDBC specification recommends using a JNDI naming service to acquire a connection to a database. TopLink supports acquiring a database connection in this fashion. To take advantage of this feature, construct and configure an instance of `oracle.toplink.jndi.JNDIConnector` and pass it to the project login object using the `setConnector` method.

**Java Persistence API (JPA)**

The Java Persistence API (JPA) provides a **POJO** persistence model for object-relational mapping in both Java EE and Java SE applications.

TopLink is the default JPA persistence provider in OC4J.

**Java Transaction API (JTA)**

The Java Transaction API (JTA) specifies the interfaces between a transaction manager, a resource manager, an application server, and transactional applications involved in a distributed transaction system.

**just-in-time reading**

A synonym for [indirection](#).

**lazy loading**

A synonym for [indirection](#).

This is the term used for [indirection](#) in the [Java Persistence API \(JPA\)](#).

**lazy reading**

A synonym for [indirection](#).

**leaf class**

Has a persistent superclass in the hierarchy but does not have subclasses; queries performed on the leaf class can return only instances of the leaf class.

Compare to [branch class](#).

**locking policy**

A mechanism that ensures one user does not overwrite another users' work. TopLink descriptors support optimistic and pessimistic locking policies.

**mappings**

Describe how individual Java objects and attributes relate to a data source.

**message-driven beans**

An EJB that processes asynchronous Java Messaging Service (JMS) messages. For TopLink clients, a message-driven bean is simply a JMS consumer with no conversational state and no home or remote interfaces.

**method access**

The application registers accessor methods for the attribute.

Compare to [direct access](#).

**named query**

A TopLink query that is created and stored, by name, in a session for later retrieval and execution

**object identity**

Ensures that each object is represented by one and only one instance in the application; that is, multiple retrievals of the same object return references to the same object instance, not multiple copies of the same object. Violating object identity can corrupt the object model.

See also [identity map](#).

**object-relational data type**

The object-relational data type paradigm extends traditional relational databases to include object-oriented functions. Oracle, IBM DB2, Informix, and other DBMS

databases allow users to store, access, and use complex data in more sophisticated ways.

The object-relational data type standard is an evolving standard concerned mainly with extending the database data structures and SQL (SQL 3).

Object-relational data type descriptors describe Java objects that you map to special relational database types that correspond more closely to object types. Using these special object-relational data type database types can simplify mapping objects to relational database tables. Not all relational databases support these special object-relational data type database types.

### **optimistic locking**

Also known as write locking; allows unlimited read access to objects. A client can write an object to the database only if the object has not changed since it was last read.

Compare to [pessimistic locking](#).

### **packet**

A piece of a message transmitted over a packet-switching network. One of the key features of a packet is that it contains the destination address in addition to the data.

### **packet time-to-live**

A number of hops that session data packets can take before expiring. The default is 2.

See also [packet](#).

### **persist**

In object technology, the storage of an Java object by a data source.

### **pessimistic locking**

Objects are locked before they are edited, which ensures that only one client is editing the object at any given time.

Compare to [optimistic locking](#).

### **POJO**

Plain Old Java Object.

In TopLink, POJO means just a regular Java object model class and is used to refer to using the TopLink API directly rather than using TopLink API indirectly by way of CMP or JPA.

### **primary key**

A field (or combination of fields) that uniquely identifies a record in the data source.

### **private relationship**

A relationship in which the target object is considered to be a private component of the source object; the target object cannot exist without the source and is accessible only through the source object; furthermore, if the source object is destroyed, the target object is destroyed as well.

Compare to [independent relationship](#).

### **proxy indirection**

A type of TopLink [indirection](#).

Introduced in JDK 1.3, the Java class `Proxy` lets you to use dynamic proxy objects as place-holders for a defined interface. Certain TopLink mappings can be configured to use proxy indirection, which gives you the benefits of TopLink indirection without the need to include TopLink classes in your domain model or use weaving.

**query manager**

An object, owned by a descriptor, that controls the way the descriptor accesses the database. The query manager generates its own default SQL to access the database in a transparent manner.

**query optimization**

TopLink supports two forms of query optimization: joining and batch reading. Their purpose is to optimize database access through reducing the number of database calls required to read a group of objects.

**relationship**

In TopLink, a reference between two TopLink-enabled objects.

**relationship mapping**

Persistent objects use relationship mappings to store references to instances of other persistent classes. The appropriate mapping class is chosen primarily by the cardinality of the relationship. TopLink provides five classes of relationship mappings.

Compare to [direct mapping](#).

**Remote Method Invocation (RMI)**

A set of protocols that enable Java objects to communicate remotely with other Java objects.

**remote session**

A remote session is a client-side session that communicates over RMI with a corresponding client session and server session on the server side. Remote sessions handle object identity and marshalling and unmarshalling between client side and server side.

**service channel**

A name of the TopLink coordinated cache channel to which sessions subscribe in order to participate in the same coordinated cache.

**session beans**

EJB that represent a business operation, task, or process. TopLink can use session beans to make the regular Java objects they access persistent, or to wrap other legacy applications.

**stale data**

An artifact of caching, in which an object in the cache is not the most recent version committed to the data source.

**TopLink session broker**

A mechanism that enables client applications to transparently access multiple databases through a single TopLink session.

**transparent indirect container indirection**

A type of TopLink [indirection](#).

Using this type of TopLink indirection, you can configure indirection for any relationship attribute of a persistent class that holds a collection of related objects as any of the following:

- `java.util.Collection`
- `java.util.Hastable`
- `java.util.List`
- `java.util.Map`
- `java.util.Set`
- `java.util.Vector`

TopLink will use an indirection object that implements the appropriate interface and also performs just-in-time reading of the related objects. When using transparent indirection, you do not have to declare the attributes as `ValueHolderInterface`.

Newly created collection mappings use transparent indirection by default if their attribute *is not* a `ValueHolderInterface`.

For JPA entities or POJO classes that you configure for weaving, TopLink weaves **value holder indirection** for one-to-one mappings and **transparent indirect container indirection** for collection mappings

### **unit of work**

A transactional TopLink session that allows for a transaction to occur at the object level not only the database level. Changes to objects are not visible globally until the unit of work is committed.

### **value holder indirection**

A type of TopLink **indirection**.

Persistent classes that use indirection must replace relationship attributes with value holder attributes. A value holder is an instance of a class that implements the `ValueHolderInterface` interface, such as `ValueHolder`. This object stores the information necessary to retrieve the object it is replacing from the database. If the application does not access the value holder, the replaced object is never read from the database. To obtain the object that the value holder replaces, use the `getValue` and `setValue` methods of the `ValueHolderInterface`. A convenient way of using these methods is to hide the `getValue` and `setValue` methods of the `ValueHolderInterface` inside `get` and `set` methods, as shown in the following illustrations.

For JPA entities or POJO classes that you configure for weaving, TopLink weaves **value holder indirection** for one-to-one mappings and **transparent indirect container indirection** for collection mappings

## A

---

### access

- data access, 96-1
- direct, 121-37
- modifiers, 5-46
- optimizing data access, 12-17
- remote sessions, 87-33

### access method

- direct, 121-16
- generating, 5-33
- mappings, 121-18
- method, 121-16
- specifying, 121-37

### access modifiers, classes, 5-46

### acquiring

- client sessions, 90-6
- sessions, at runtime, 87-5
- unit of work, 114-1

### activating descriptors, 5-10

### Add Named Query dialog, 119-14, 119-21, 119-23

### Add New Class dialog, 5-45, 5-58

### Add New Table button, 5-23

### Add or Refresh Class button, 5-56

### addConstraingDependencies(), 115-16

### address

- multicast group, 103-5
- multicast port, 103-7

### Add/Update Existing Tables from Database

- button, 5-24

### advanced properties for descriptors, 117-8

### After Load tab, 119-91

### aggregate collection relational mappings

- and EJB, 27-11
- configuring, 35-1
- understanding, 27-10

### aggregate descriptors

- about, 16-5
- aggregate object mapping, 21-1
- EIS projects, 74-2
- EJB 3.0, 16-5, 21-2
- inheritance, 16-12
- relational projects, 21-1
- XML projects, 50-1

### aggregate object relational mappings

- aggregate descriptors, and, 21-1

### configuring, 37-1

### understanding, 27-13

### aggregation, isolated client sessions, 87-25

### AllFieldsLockingPolicy, 16-16

### allows none, 110-14, 119-21

### allows null, 110-14, 119-21

### amending descriptors, 2-24, 16-6, 119-90

### *see also* after load

### announcement delay, 103-15

### Ant, integrating with Oracle TopLink

### Workbench, 5-58

### any attribute XML mappings

### configuring, 70-1

### any collection XML mappings

### configuring, 60-1

### understanding, 53-33

### any object XML mappings

### configuring, 59-1

### understanding, 53-31

### application development

### deploying, 11-1

### mapping, 2-24

### querying, 2-15, 86-2

### application layer, remote sessions, 87-32

### Application Navigator

### TopLink elements, 4-2

### application servers

### EJB support, customizing, 8-21

### integrating with Oracle TopLink, 8-1

### logging, 87-10

### MaD support, 8-16

### optimization, 12-36

### setter parameter type checking, 8-21, 8-22

### single-object finder return type checking, 8-21, 8-22

### software requirements, 8-2

### target platforms, 2-5

### unknown primary key class support, 8-21, 8-22

### architectures

### application, 1-5

### BMP, 1-6, 2-38

### cache, 102-1

### choosing, 2-3

### CMP, 1-6, 2-34

### EIS, 2-4, 77-5

### EJB entity beans, 1-6, 2-34, 2-38

- EJB session bean facade, 1-5, 2-31, 2-32
- locking, 2-11
- optimistic locking, 2-11
- Oracle TopLink, 1-1
- pessimistic locking, 2-11
- selecting, 2-5
- session brokers, 87-27
- sessions, 87-3
- three-tier, 1-5, 2-28
- two-tier, 1-7, 2-30, 2-31
- unit of work, 113-1
- web services, 1-6, 2-42
- arguments, binding in query, 23-7
- array
  - dimensionality, 5-49, 5-54
  - object-relational data type mappings, 40-3
- AsOfClause, 111-2
- asynchronous change propagation, 103-2
- AttributeChangeTrackingPolicy
  - about, 113-8
  - JPA entities, 113-8
  - OC4J CMP integration, 113-9
  - POJO classes, 113-8
- attributes
  - adding to descriptors, 5-48
  - array dimensionality, 5-49, 5-54
  - changes, tracking, 119-83
  - final, 5-49, 5-53
  - in TopLink Workbench, Navigator window, 5-9
  - static, 5-49, 5-53
  - transforming, 27-17, 121-34, 121-36
  - transient, 5-49, 5-53
  - unmapping, 120-5
  - volatile, 5-49, 5-53
- Attributes tab, 5-48, 5-49, 5-52
- Attunity Connect platform, 96-3
- auditing
  - authentication, 96-6
  - unit of work, 115-20
- authentication
  - about, 96-5
  - auditing, 96-6
  - proxy authentication, 96-5
  - simple JDBC authentication, 96-5
  - three-tier architecture, 96-5
  - two-tier architecture, 96-5
- Automap, 120-3
- automapping descriptors
  - about, 120-3
  - see also* mappings
- automatic table generation
  - about, 17-5
  - configuring, 9-12

## B

---

- Base64 encoded strings, 121-29
- batch
  - writing
    - JDBC, 12-19, 98-11

## Index-2

- batch options
  - mappings, 28-8
  - writing, 12-18
- batch reading
  - in query objects, 109-10
  - read optimization, and, 12-25
- batch writing
  - about, 12-18, 12-32, 98-12
  - dynamic, 12-18
  - dynamic, `setMaxBatchWritingSize()`, 12-18
  - JDBC, 12-19, 98-11
  - MySQL4 platform, 98-10
  - native, 12-19, 98-11
  - non-parameterized, 12-18
  - parameterized, 12-18
  - `setMaxBatchWritingSize()`, 12-18
  - TopLink, 12-19, 98-11
- BEA WebLogic
  - setting classpath, 8-5
  - setting shared library, 8-5
  - using a security manager, 8-6
- beans
  - session beans, 2-33
  - stateful beans, 2-33
  - stateless beans, 2-33
- bidirectional relationships
  - about, 27-2
  - generating, 5-34
  - in Java, 121-42
  - in one-to-one mappings, 121-42
  - target keys, 27-6
  - with indirection, 121-42
- binary data collection XML mappings
  - configuring, 65-1, 67-1
- binary data XML mappings
  - configuring, 64-1
- `bindAllParameters()` method, 20-8
- bindings
  - arguments, 23-7
  - JAXB, 47-2
  - see* parameter binding
- BLOB
  - mapping to, 120-3
- BMP
  - and EJB 1.1, 2-39
  - and EJB 2.0, 2-39
  - and TopLink, 2-38
  - deployment files, 9-9
  - descriptors, 16-3, 119-51
  - packaging for deployment, 10-6
- boolean logic in expressions, 110-3
- branch classes, 16-9, 16-10
- buttons. *see* toolbars
- Byte array Base64, 121-29

## C

---

- cache
  - about, 2-15, 2-20, 102-1
  - architecture, 102-1



- configuring, 102-6, 111-20
- coordination, 102-9, 103-1, 104-1, 105-1
- descriptor level, 119-39
- disabling during read query, 108-35
- distributed, 102-9
- expiration, 111-21, 117-22, 119-47
- expression limitations, 108-32
- identity maps, using, 87-30
- in-memory queries, 108-31, 108-32
- internal query object cache, 108-37, 108-38, 111-20, 111-21
- invalidation, 102-7, 102-8, 117-22, 119-47
- isolated client sessions, 87-25
- isolation, 102-6, 102-9
- object cache, 108-37
- object cascading refresh, 108-36
- object refresh, 108-36
- optimizing, 12-16
- project level, 117-14
- queries, 102-6, 108-30
- query cache, 108-37
- readObject method, and, 109-2
- refreshing, 102-8, 108-36, 119-30
- restrictions, 108-38
- service channel, 103-4
- sessions, 87-3, 87-34, 102-2
- stale data, 102-6
- storing query results, 108-37, 111-20
- type and size, 117-14, 119-39
- unit of work cache, 102-2
- cache coordination
  - about, 102-9
  - application server clustering, and, 8-5
  - avoiding stale data, 102-7
  - CMP projects, 9-12
  - EJB Entity Beans with BMP architecture, and, 2-40
  - explicit query refreshes, 102-7
  - JMS, 104-1
  - JMS, connection handling, 104-5
  - JMS, error receiving message, 104-5
  - JMS, ERROR\_RECEIVED\_JMS\_MESSAGE\_IS\_NULL, 104-5
  - JMS, ERROR\_RECEIVING\_JMS\_MESSAGE, 104-5
  - JMS, null JMS message, 104-5
  - orion-ejb-jar.xml, 9-12
  - packet time-to-live, 103-19
  - permissions, 8-20
  - RMI, 105-1
- cache invalidation, avoiding stale data, 102-7
- cache synchronization. *see* cache coordination
- cacheAllStatements(), 20-8
- cacheQueryResults(), 119-28
- cache-synchronization property, 9-12
- Caching tab, 119-40, 119-42, 119-45, 119-49
- calendar, 28-3
- call
  - call queries, 108-17
  - EIS, 109-32
  - EJBQLCall, 109-32
  - JPQLCall, 109-32
  - SQLCall, about, 109-18
  - SQLCall, JDBC arguments, 109-19
  - SQLCall, no arguments, 109-19
  - StoredProcedureCall, 109-31
  - StoredProcedureCall, about, 109-21
  - StoredProcedureCall, ErrorCodeListener, 109-29
  - StoredProcedureCall, JDBC arguments, 109-22, 109-24
  - StoredProcedureCall, no arguments, 109-22
  - StoredProcedureCall, PL/SQL arguments, 109-24
- Call object, queries, 108-3
- cascading
  - object refresh, 108-36
  - optimistic version locking, 16-14
  - write queries, compared to
    - non-cascading, 108-15, 109-13
- catalog, database, 5-23
- catchExcpetions(), 89-21
- CDATA in mappings, 121-44
- change policy
  - about, 113-7
  - attribute change tracking, configuring, 119-83
  - configuring, 119-81
  - deferred change detection, configuring, 119-82
  - empty transaction, 113-7
  - object change tracking, configuring, 119-82
  - unit of work, 113-7
- change tracking
  - attribute, configuring, 119-83
  - deferred, configuring, 119-82
  - method and direct field access limitations, 2-13, 117-5, 121-17
  - object, configuring, 119-82
- changed items, displaying in TopLink Workbench Navigator window, 5-9
- ChangedFieldsLockingPolicy, 16-16
- changing package names, 5-57
- checkDatabase(), 89-21
- checking in/out projects, 7-3
- checkInstantiationPolicy(), 89-22
- choice collection XML mappings
  - configuring, 69-1
- choice object XML mappings
  - configuring, 68-1
- Choose a Schema Context dialog box, 52-3, 76-3
- Choose Query Key dialog box, 110-14
- Choose Relationships to Generate dialog box, 5-33
- Choose Root Element dialog box, 52-6, 76-5
- class extraction method
  - about, 16-11
  - inheritance, 16-12
- class indicator
  - about, 16-10
  - class extraction method, 16-11
  - class indicator field, 16-10, 32-2
- class loader
  - host application, 2-5
  - IBM WebSphere, 8-17

- loading session, 90-4
- class modifiers, 5-46
- Class tab, 5-46, 5-47
- classes
  - access modifiers, 5-46
  - adding and refreshing, 5-56
  - branch, 16-10
  - creating, 5-44, 5-58
  - CursoredStream, optimizing, 111-18
  - Database Exception, 109-33
  - DatabaseMapping, 17-28
  - default null values, 121-43
  - DeleteObjectQuery, 109-13
  - DirectCollectionMapping, 36-3
  - ExpressionBuilder, 110-16
  - generating from database, 5-32
  - InsertObjectQuery, 109-12
  - InsertObjectQuery, 113-12
  - interfaces, 5-47
  - leaf, 16-10
  - merging files, 7-5
  - methods, adding, 5-52
  - naming, 5-46
  - non-descriptor classes, 5-57
  - object model, 2-18
  - Performance Profiler, 12-3
  - persistent requirements, 2-13
  - preferences, 5-16
  - refreshing, 5-56
  - removing, 5-56
  - root, 16-9
  - troubleshooting, A-34
  - unit of work, 113-12
  - UpdateObjectQuery, 109-12, 109-13
  - ValueHolderInterface, 2-14, 17-8, 27-10, Glossary-8
  - VariableOneToOneMapping, 27-7
  - see also* specific class name
- classpath
  - adding, 117-4
  - BEA WebLogic, 8-5
  - configuring, 5-2, 8-5, 8-18
  - connector.jar, 5-2
  - custom Collection class, 5-3
  - DRIVER\_CLASSPATH, Oracle TopLink Workbench, 5-2
  - JCA adapter, 5-2
  - JDBC driver, 5-2, 5-24
  - JDBC\_CLASSPATH, 5-2
  - Oracle TopLink Workbench DRIVER\_CLASSPATH, 5-2
  - relative, 117-4
  - SunAS, 8-18
  - troubleshooting, A-34
  - xdb.jar file, 5-3
- client session
  - logging out, 90-10
  - releasing, 90-10
- client sessions
  - about, 87-2, 87-14, 87-26
  - acquiring, 90-6
  - configuration, 94-1
  - example, 87-16
  - shared resources, 87-16
- client-controlled transactions, 115-24
- client-server architecture. *See* two-tier
- CLOB
  - mapping to, 120-3
- clones
  - copying methods, 119-80
  - merging changes, 115-12
  - post-commit, avoiding, 115-31
  - unit of work, 113-2, 113-9
- Cloudscape platform, 96-3
- clustering, integrating TopLink with, 8-5
- CMP
  - and EJB 1.1, 2-34
  - and EJB 2.x, 2-34
  - and EJB 3.0, 2-34
  - and TopLink, 2-34
  - automatic table creation, 6-7
  - CMPPolicy, 16-3, 119-51
  - deploying, 9-8, 11-2
  - descriptors, inheritance, 16-5
  - external transactions, 113-3
  - isolated client sessions, 87-25
  - local transaction, 115-25
  - non-deferred write, 115-25
  - OC4J persistence, 8-7
  - packaging for deployment, 10-5
  - schema generation, 6-7
  - schema manager, 6-7
  - setter parameter type checking, 8-22
  - single-object finder return type checking, 8-22
  - transaction attribute, 115-24
  - unit of work, 113-3, 115-24
  - unknown primary key class support, 8-22
- code generation, optimizing, 12-11
- collapsing items in Navigator window, 5-10
- collection class
  - specifying, 109-9
  - specifying in query objects, 109-9
- collection reference XML mappings
  - configuring, 63-1
- collections
  - persistent requirements for mappings, 2-14
  - query results, 108-8
- comments
  - descriptors, 119-10, 119-11
  - mappings, 121-21
  - projects, 117-23
- commit
  - and Java Transaction API, 113-10, 113-11
  - failure, resuming after, 115-14
  - resuming unit of work after, 115-13
- Communication Exceptions, A-7
- composite collection EIS mappings
  - configuring, 82-1
  - example, 77-8
  - understanding, 77-8

- composite collection XML mappings
  - configuration, 58-1
  - configuring, 58-1
  - understanding, 53-29
- composite descriptors
  - about, 16-5
  - composite object mapping, 50-1, 74-2
  - EIS projects, 74-2
  - XML projects, 50-1
- composite direct collection EIS mappings
  - configuring, 80-1
  - understanding, 77-7
- composite direct collection XML mappings
  - configuring, 56-1
  - understanding, 53-18
- composite EIS descriptors, 75-2
- composite object EIS mappings
  - composite descriptors, and, 74-2
  - configuring, 81-1
  - understanding, 77-7
- composite object mappings
  - composite descriptors, and, 50-1
- composite object XML mappings
  - configuration, 57-1
  - configuring, 57-1
  - understanding, 53-24
- composite primary key, 27-10, 121-42
- concrete class. *see* container policy
- concurrency
  - about, 2-20
  - exceptions, A-4
  - server session, 87-19
- Concurrency Exceptions, A-4
- configurations
  - about, 88-2
  - creating, 88-2
  - development environment, 7-1
  - new, 88-2
  - Oracle JDeveloper, 7-1
  - session, 88-2, 88-3, 88-4, 88-7, 88-9
- conforming
  - about, 115-7
  - alternatives, UnitOfWork method
    - writeChanges, 115-11
  - alternatives, UnitOfWork properties, 115-12
  - descriptors, and, 115-11
  - queries, alternatives to, 115-11
- ConnectBy, 111-7
- connection policy
  - configuring, 89-22
  - exclusive connections, 89-22
  - lazy connection acquisition, 89-23
- connection pool
  - about, 96-7
  - connection count, 101-2
  - ConnectionPolicy, 87-21
  - external, 96-8
  - internal, 96-7, 96-8, 96-9
  - lazy connection allocation, 87-19, 90-8
  - named, 96-9
  - parameter binding, 12-20
  - prepared statement caching, 12-20
  - read, 96-8
  - sequence, 15-5, 96-8
  - server session, 87-19
  - sessions, and, 87-4
  - size, 101-2
  - write, 96-8
- Connection Specifications tab, 73-3
- Connection tab, 73-3
- connections
  - about, 96-7
  - connection pool, 96-7
  - exclusive write connection, 89-22
  - lazy acquisition, 89-23
  - reading through the write connection, 115-27
- connector.jar, 5-2, 71-1
- container policy
  - about, 121-30
  - custom Collection class, 5-3
  - sorting, in memory, 121-31
- container-controlled transactions, 115-24
- context
  - JAXB context, 47-8
  - menus, 5-6
  - schema, 52-2, 76-3
- Context.SECURITY\_CREDENTIALS, 103-12
- Context.SECURITY\_PRINCIPAL, 103-11
- Conversion Exception, A-4
- Converter tab
  - object type mappings, 121-25, 121-26
- converters
  - custom, 27-3
  - object type, 121-25, 121-26
- coordinated announcement delay, 103-15
- coordinated cache, 9-12
  - configuring, 103-1
  - naming service, 103-9
  - service channel, 103-4
- copy policy
  - about, 119-79
  - method, 119-80
  - setting, 119-80
- copying project objects, 7-6
- Copying tab, 119-80
- CORBA
  - Oracle TopLink transport layer support, 87-32
  - Transaction Service *see* OTS
- Create New Project button, 116-2
- Create New Project dialog box, 116-2
- Create New Session dialog, 88-4, 88-7, 88-9
- Create Project from JAXB dialog, 48-2
- creating
  - configurations, 88-2
  - expressions, 110-13
  - sessions, 88-4, 88-7, 88-9, 100-1
- Crimson XML parser, 8-4
- cursor streams
  - example, 111-18
  - optimizing, 111-18

- remote sessions, 87-34
- usage example, 87-34
- cursors
  - as query results, 108-9
  - traversing scrollable, 111-16
- Custom Calls tab, 76-7
- custom SQL
  - Custom SQL tab, 23-8
  - unit of work, 115-15
- customization
  - about, 13-1
  - data types, 13-1
  - EIS, 13-1
  - mapping extensions, 13-1
  - overview, 2-17
  - XML, 13-1

## D

---

- data access
  - about, 96-1
  - authentication, 96-5
  - connection pool, 96-7
  - connections, 96-7
  - optimizing, 12-17, 12-18
  - platforms, 96-3
- data level queries
  - example, 110-12
  - in expressions, 110-11
- data source platform
  - about, 96-3
  - Attunity Connect database, 96-3
  - Cloudscape database, 96-3
  - databases, 96-3
  - DB2 database, 96-3
  - EIS, 96-4
  - HSQL database, 96-3
  - Informix database, 96-3
  - JCA adapter, 96-4
  - JDBC drivers, 96-3
  - Microsoft Access database, 96-3
  - MySQL4 database, 96-3
  - Oracle database, 96-3
  - Oracle8 database, 96-3
  - Oracle9 database, 96-3
  - PointBase database, 96-3
  - SQLAnywhere database, 96-3
  - SQLServer database, 96-3
  - SybasePlatform database, 96-3
- data sources
  - configuring, 97-1
  - nontransactional, 96-1
  - transactional, 96-1
- Database Exceptions, 109-33, A-4
- database fields, configuring, 28-3
- database functions, in expressions, 110-3
- database login
  - parameter binding, 98-10
  - prepared statement caching, 98-10
- Database Preferences, 5-18

- database queries
  - about, 108-10, 109-4
  - fetch groups, 108-14
  - join reading, 108-12
  - object level modify query, 108-14, 108-16
  - object level read query, 108-11, 108-14
  - partial object query, 108-11
  - read all query, 108-11
  - read object query, 108-11
  - report query, 108-16
- database schema
  - tables, 5-23
- database session
  - logging out, 90-10
- database sessions
  - about, 87-2, 87-29
  - cache, 87-34
  - configuration, 95-1
  - creating, 88-8
- database tables
  - about, 5-22
  - adding to database, 5-23
  - creating, 5-23
  - descriptors and classes, generating, 5-32
  - EJB entity generation, 5-34
  - fields, 5-27
  - generating, 5-32, 5-35
  - importing, 5-24
  - Java source generation, 19-4
  - JDBC driver classpath, 5-24
  - properties, 5-27
  - references, 5-29, 5-30
  - removing, 5-26
  - renaming, 5-26
  - schema, 5-23
  - SQL generation, 5-32
  - TopLink Workbench, Navigator window, 5-10
- DatabaseException class, 109-33
- DatabaseLogin, 96-2
- DatabaseMapping class, 17-28
- DatabaseQuery, 108-3
- DatabaseRow, 27-17
- databases
  - catalog, 5-23
  - common problems, A-35
  - connect to, 5-23
  - creating reference tables on, 5-29
  - custom drivers, 5-19
  - disconnect from, 5-23
  - drivers, 5-19
  - exceptions, 109-33
  - fields, configuring, 28-3
  - for project, 116-3
  - Java type conversion, 6-3
  - linking, 87-29
  - log out of, 5-23
  - logging into, 5-23, 20-8
  - logins, 98-1
  - mapping. *See* mappings
  - platform, 20-3, 73-2, 98-2, 99-1, 116-3

- preferences, 5-18
- schema, 5-23
- schema manager, 6-1
- tables, 5-22
- TopLink Workbench, Navigator window, 5-10
- troubleshooting, A-35
- type conversion, schema manager, 6-3
- using with Oracle TopLink Workbench, 5-22
- DatabaseSession class
  - logging SQL and messages, 87-10
- DB2
  - platform, 96-3
  - schema manager type conversion, 6-3
- DBase platform, 96-3
- default mapping
  - about, 17-5
  - automatic table generation, and, 17-5
  - configuring, 9-12
  - default table generator, 6-6
- Default Mapping Exception, A-7
- default table generator
  - default mapping, 6-6
  - table creator, creating, 6-4
- defaults
  - login level null values, 97-8
  - mapping level null values, 121-14
  - null values, 97-8, 121-14, 121-43
  - optimization, 12-11
  - root, 52-5
  - see also* preferences
- DefaultSequence, 98-7
- deferred change detection
  - configuring, 119-82
- DeferredChangeDetectionPolicy, 113-7
- Delete All Interaction tab, 84-4
- deleteObject(), 23-9
- DeleteObjectQuery, 109-13
- deletes
  - controlling order, 115-16
  - delete operation, 109-4
  - queries, EIS mappings, 84-4
- demarcation of unit of work transactions, 113-2
- dependent objects
  - non-deferred write, 16-4
- deploying
  - about, 2-16, 2-17, 11-1
  - application server requirements, 8-1
  - CMP applications, 11-2, 11-3
  - database login, 20-8
  - entity beans overview, 2-17
  - generating XML for, 9-3
  - hot deployment, 11-3
  - Java applications, 11-1
  - JSP and Servlet applications, 11-1
  - non-CMP applications, 11-3
  - packaging, 10-1
  - Session Bean applications, 11-2
- deployment descriptors, 16-3
- deployment files
  - BMP applications, 9-9
  - CMP applications, 9-8
  - creating, 9-2
  - descriptors, 16-3
  - EJB 3.0, 9-2, 10-1
  - Java applications, 9-7
  - JSP and Servlet applications, 9-7
  - Session Bean applications, 9-7
  - XML, generating, 9-3
- deployment XML, exporting, 116-9
- DeploymentXMLGenerator, 9-3
- descriptor customizer, 16-5
- Descriptor Event Listener, 119-70
- Descriptor Event Manager
  - about, 16-6
  - Descriptor Event Listener, 119-69
  - domain object methods, 119-66
  - event types, 119-66
  - handlers, 16-6, 119-66, 119-69
  - handlers, Descriptor Event Listener, 119-70
  - understanding, 16-6
- descriptor events
  - about, 16-6
  - Descriptor Event Listener, 119-69
  - domain object methods, 119-66
  - handlers, 16-6, 119-66, 119-69, 119-70
  - types of, 119-66
  - understanding, 16-6
- Descriptor Exceptions, A-3, A-5
- Descriptor Info tab, 23-3, 23-4, 52-2, 52-5, 52-7, 76-3, 76-4, 119-5, 119-7, 119-8
- DescriptorEventListener, 119-69
- descriptors
  - about, 14-2, 16-1
  - advanced properties, default, 117-8
  - aggregate, 16-5, 22-2
  - aggregate, EJB 3.0 and, 16-5, 21-2
  - aggregate, relational projects and, 21-1
  - amending, 2-24, 16-6, 119-90
  - API, 16-18
  - architecture, 16-2
  - attributes, adding, 5-48
  - automapping, 120-3
  - automatically mapping, 120-3
  - BMP, 16-3, 119-51
  - cache refreshing, 119-30
  - change policy, 119-81
  - child inheritance, 119-57
  - class, 22-2
  - CMP, 16-3, 119-51
  - CMPPolicy, 16-3, 119-51
  - comments, 119-10, 119-11
  - composite, 16-5, 50-1, 74-2, 75-2
  - composite EIS, 75-2
  - configuring, 119-1, 119-2
  - conforming, 115-7
  - creating, 22-1, 22-2, 25-1, 75-1, 75-2, 118-1
  - custom EIS interactions for basic persistence, 76-6
  - custom SQL queries for basic persistence, 23-7
  - customization, 16-5
  - deactivating, 5-10

- default mappings, 17-5
- default root, 76-4
- deployment information, 16-3
- Descriptor Event Listener, 119-69, 119-70
- domain object methods, 119-66
- EIS, 74-1, 75-1, 75-2, 76-1
- EIS projects, 74-2
- EJB, 16-3, 119-51
- EJB information, 16-3, 119-51
- errors, 5-10, 5-11, A-12
- event handlers, 16-6, 119-66, 119-69, 119-70
- events, 16-6, 119-66
- existence checking, 12-16, 119-49
- fetch groups, 119-88
- files, merging, 7-5
- generating from database, 5-32
- hierarchy, inheritance, 16-19
- history policy, 119-84
- identity maps, 117-15, 119-40, 119-42, 119-45
- in Java, 16-18
- inactive, 5-10
- inheritance, 16-3, 16-9, 21-4, 119-57, 119-59
- instantiation, 12-16
- interface, 22-2, 119-35, 119-37
- mapping, 17-5, 23-3, 119-5, 120-2, 120-3
- merging, 7-5
- named queries, 119-12
- nondesoriptor classes, 5-57
- object-relational data type, 24-1, 26-1
- optimizing, 12-16
- parent inheritance, 119-59
- projects, 14-2, 16-1
- query key interfaces, 119-35
- query timeout, 119-27, 119-29
- read-only, 119-6
- registering with sessions, 87-13, 89-3, 89-4, 89-12
- relational, 21-1, 22-2, 23-1
- removing, 5-56
- returning policy, 119-75
- root EIS, 75-1
- root element, 52-5
- schema context, 52-2, 52-3, 76-3
- sequencing, 16-7, 23-4
- TopLink Workbench, Navigator window, 5-9
- types of, 16-1
- validating, 118-1
- XML, 50-1, 51-1, 52-1
- XML projects, 50-1
- developing applications with Oracle TopLink, 2-1
- development environments
  - about, 3-2
  - configuring, 7-1
  - database logins, 20-8
- development process
  - about, 2-1
  - additional support, 2-3
  - stages of, 2-2
  - with Oracle TopLink, 2-1
- development tools
  - about, 3-1
  - profiler, 12-2
  - schema manager, 6-1
- dimensionality, array, 5-49, 5-54
- direct access
  - about, 117-5, 121-16
  - specifying, 121-37
- direct collection relational mappings
  - configuring, 36-1
  - example, 27-12, 36-3
  - in Java code, 36-3
  - understanding, 27-12
- direct collections
  - session broker limitations, 87-28
- direct EIS mappings
  - configuring, 79-1
  - understanding, 77-6
- direct field
  - in direct collection mappings, 36-4
- direct field access
  - change tracking limitations, 2-13, 117-5, 121-17
- direct key fields, 38-3
- direct map relational mappings
  - configuring, 38-1
  - direct keys, 38-3
  - direct value, 38-2
  - understanding, 27-13
- direct mappings
  - generating deprecated, 20-13
  - with EJB, 27-4
- direct value fields, 38-2
- direct XML mappings
  - configuring, 55-1
  - understanding, 53-7
- DirectCollectionMapping class, 36-3
- directionality in mappings, 27-2
- direct-to-field mappings
  - ObjectTypeMapping deprecated, 27-3
  - SerializedObjectMapping deprecated, 27-3
  - type conversions, 28-3
  - TypeConversionMapping deprecated, 27-3
- direct-to-field relational mappings
  - configuring, 29-1
  - options, 29-1
  - timestamp support, 28-4
  - understanding, 27-4
- direct-to-XMLType relational mappings
  - configuring, 30-1
  - understanding, 27-5
- Discovery Exception, A-8
- DMS profiler
  - about, 12-4, 87-12
  - accessing with JMX, 12-9
  - and JMX
  - nouns, 12-4, 87-12
  - selecting, 89-13, 89-16, 89-20, 89-23
  - sensors, 12-4, 87-12
- document information in XML schemas, 5-37, 5-40, 5-43
- documentation
  - hosted, 5-14

- See also Help*
- does exist write object, 12-32
- dontOptimizeDataConversion(), 12-18
- doPrivileged(), 8-4
- DRIVER\_CLASSPATH
  - Oracle TopLink Workbench environment, 5-2
- drivers, custom database, 5-19
- dynamic batch writing
  - about, 12-18
  - setMaxBatchWritingSize(), 12-18
- dynamic fetch groups, querying with, 111-4

## E

---

- editor, 4-1
- editor window
  - TopLink, 4-2
- Editor window, about, 5-5, 5-11
- EIS
  - about, 71-2
  - architecture, 2-4
  - call, 109-32
  - custom interactions for basic persistence, per
    - descriptor, 76-6
  - indexed records, configuring, 76-6
  - interactions, 76-6, 108-6, 109-32
  - mapped records, configuring, 76-6
  - mappings, 71-2, 77-2
  - projects, 116-3
  - queries, 108-6
  - record format, configuring, 76-6
  - XML records, configuring, 76-6
- EIS descriptors
  - composite, 75-2
  - configuring, 76-1
  - default root, 76-4
  - locking policy, 119-71
  - root descriptor, 75-2
  - schema context, 76-3
  - setDataTypeName, 76-5
  - understanding, 74-1
- EIS mappings
  - about, 77-1, 77-2
  - architecture, 77-5
  - composite collection, 77-8, 82-1
  - composite direct collection, 77-7, 80-1
  - composite object, 77-7, 81-1
  - configuring, 78-1
  - direct, 77-6, 79-1
  - jaxb:class support, 77-3
  - list support, 77-3
  - one-to-many, 77-13, 84-1
  - one-to-many, key on source, 77-14
  - one-to-many, key on target, 77-16
  - one-to-one, 77-9, 83-1
  - one-to-one, key on source, 77-10
  - one-to-one, key on target, 77-11
  - transformation, 77-18, 85-1
  - types of, 77-1
  - union support, 77-3

- xsd:list, 77-3
- xsd:union, 77-3
- EIS projects
  - configuring, 73-1
  - connector.jar, 71-1
  - indexed records, 72-2
  - mapped records, 72-2
  - sequencing, 15-5
  - understanding, 71-1
  - XML records, 72-1
- EIS queries, 108-6
- EIS record types, supported, 77-2
- EISLogin, 96-2
- EJB
  - descriptors, 116-5
  - isolated client sessions, 87-25
  - setter parameter type checking, 8-21
  - single-object finder return type checking, 8-21
  - unknown primary key class support, 8-21
- EJB 1.1
  - and BMP, 2-39
  - and CMP, 2-34
- EJB 2.x
  - and BMP, 2-39
  - and CMP, 2-34
  - default mapping, 17-5
- EJB 3.0
  - <Java EE-Container>-jar.xml file, 9-6
  - and CMP, 2-34
  - default mapping, 17-5
  - deployment files, 9-2, 10-1
  - Embedded annotation, 16-5, 21-2
  - packaging for deployment, 10-1
  - projects.xml file, 9-4, 9-5
  - sessions.xml file, 9-5
- EJB descriptors, opening projects with, 116-5
- EJB entities
  - CMP hot deployment, 11-3
  - deployment overview, 2-17
  - EJB 2.x indirection, 17-11
  - generating, 5-34
  - hot deployment, 11-3
  - inheritance, 16-5, 16-13
  - inserting after ejbCreate, 16-4
  - inserting after ejbPostCreate, 16-4
  - mapping, 117-7
  - non-CMP hot deployment, 11-3
  - non-deferred write, understanding, 16-3, 115-25
  - sequencing, 18-10
- EJB entity beans
  - and EJB 1.1, 2-34, 2-39
  - and EJB 2.x, 2-34, 2-39
  - and EJB 3.0, 2-34
  - with BMP architecture, 1-6, 2-38
  - with CMP architecture, 1-6, 2-34
- EJB Exception Factory Exceptions, A-6
- EJB finders
  - about, 108-25
  - Call finders, 108-27
  - creating, 111-8

- DatabaseQuery finders, about, 108-27
- default finders, about, 108-27
- default finders, creating, 111-8
- EJB QL finders, about, 108-28
- ejb-jar.xml options, 111-9
- ejbSelect method, 108-29
- ejbSelect, creating, 111-15
- ejbSelect, using, 111-15
- expression finders, about, 108-28
- named query finders, about, 108-27
- predefined, about, 108-25
- primary key finders, about, 108-28
- redirect finders, about, 108-29
- redirect finders, using, 111-12
- single-object finder return type checking, 8-21
- SQL finders, about, 108-28
- EJB Info tab, 119-52
- EJB JAR XML Exception, A-10
- EJB Preferences, 5-17
- EJB QL
  - exceptions, A-6
  - queries, 23-7, 108-5
  - query language, 108-5
- EJB session bean facade architecture
  - about, 1-5, 2-31
  - understanding, 2-32
- EJB Session Beans, 87-32
- ejbCreate, 16-4
- ejb-jar.xml file
  - about, 9-5, 19-4
  - corresponding to Oracle TopLink Workbench functions, 19-4
- EJB finder options, 111-9
- location, 117-8
- managing, 7-6
- preferences, 5-17
- synchronization under EJB 2.0, 9-5
- updating from, 19-6
- writing, 19-5
- ejbPostCreate, 16-4
- @Embeddable, 16-5, 21-2
- @Embedded, 16-5, 21-2
- empty unit of work transactions, 113-7
- encrypting login passwords, 116-9
- encryption
  - password encryption, customizing, 97-3
  - Securable interface, 97-4
- enhanced validation exceptions, 9-4
- Enterprise Information Systems. *see* EIS
- entity beans
  - deployment, 2-17
  - descriptor information, 16-3
  - direct mappings, 27-4
  - indirection, EJB 2.x, 17-11
  - sequencing with, 18-10
- Entity Manager Setup Exception, A-9
- EntityManagerSetupException, A-9
- environment
  - configuring, 5-2
  - JAVA\_HOME, 5-2
  - JDBC\_CLASSPATH, 5-2
  - proxy, 5-13, 5-14, 19-6
- error codes
  - 1-176, A-3
  - 1-99, A-10
  - 100-199, A-12
  - 200-399, A-12
  - 400-599, A-23
  - 500-699, A-27
  - 700-799, A-32
  - 800-899, A-33
  - 12000-12004, A-7
  - 18001-18002, A-7
  - 22001-22004, A-8
  - 22101-22105, A-8
  - 3001-3007, A-4
  - 4002-4018, A-4
  - 5001-5008, A-5
  - 6001-6098, A-5
  - 7001-7104, A-5
  - 72000-72023, A-9, A-10
  - 8001-8010, A-6
  - 9000-9009, A-6
- ErrorCodeListener
  - StoredProcedureCall, and, 109-29
- errors
  - about, A-2
  - codes and descriptions, A-2, A-10
  - descriptors, 5-10, 5-11
  - MaD support, A-1
  - migration, 8-15
  - Oracle TopLink Workbench, A-10
- Event Manager, 87-6
- events
  - about, 16-6
  - client session, 87-6
  - database access, 87-6
  - Descriptor Event Listener, 119-70
  - domain object methods, 119-66
  - handlers, 16-6, 119-66, 119-69
  - listeners, sessions, 87-8
  - session, 87-6
  - session manager, 87-6
  - server session, 87-6
  - types of, 119-66
  - unit of work, 87-6
- examples
  - composite collection EIS mapping, 77-8
  - context menu, 5-6
  - cursoried streams, 111-17
  - direct collection mappings, 27-12
  - direct-to-field mappings, 27-4
  - exception handler, 89-15
  - indirection, 17-7
  - inheritance, 16-9
  - Oracle TopLink Workbench, 5-3
  - performance optimization, 12-28, 12-30
  - proxy indirection in code, 121-10
  - READALL finders, 111-10
  - report query, 109-6



- scrollable cursors, 111-17
- serialized mapping, 17-12
- transformation mapping, 27-16
- transformation XML mapping, 53-36, 77-19
- Unit of Work, 113-6, 114-10
- write, write all, 109-3
- exception handler
  - about, 87-12
  - example, 89-15
  - selecting, 89-14
- exceptions
  - chained, 87-11
  - communication exceptions, A-7
  - conversion exceptions, A-4
  - database exceptions, 109-33, A-4
  - Default mapping exception, A-7
  - descriptor exceptions, A-3
  - discovery exceptions, A-8
  - EJB JAR XML exceptions, A-9, A-10
  - EJB QL exceptions, A-6
  - enhanced validation, 9-4
  - JMS processing exceptions, A-7
  - Migration utility exception, A-9
  - optimistic locking, A-5
  - query exceptions, A-5
  - remote command manager exceptions, A-8
  - selecting exception handler, 89-14
  - session loader exceptions, A-6
  - Transaction exception, A-8
  - validation exceptions, A-5
  - XML conversion exception, A-8
- exclusive connections
  - about, 115-28
  - internal read connection pool, 101-7
  - isolated sessions, 87-21, 89-22
  - named queries, 119-27
- existence checking, 115-3
  - descriptors, 119-49
  - projects, 117-10
- expanding items in Navigator window, 5-10
- expiration of objects in the cache, 117-22, 119-47
- explicit query refreshes, cache coordination, 102-7
- exporting
  - deployment XML, 116-9
  - Java model, 116-10
  - Java source, 19-3
  - preferences, 5-12
  - projects, 116-9
  - relational projects, 19-3
- Expression Builder, 110-13, 110-15
- Expression Builder dialog box, 110-13
- Expression class, 110-1
- ExpressionMath class, 110-1
- expressions
  - about, 108-4, 110-1
  - allows none, 110-14, 119-21
  - allows null, 110-14, 119-21
  - building, 110-13
  - comparing with SQL, 110-1
  - components, 110-2
  - creating, 110-12
  - data level queries, 110-11
  - database functions, 110-3
  - in relationships, 110-6
  - in-memory queries, limitations, 108-32
  - mathematical functions, 110-4
  - multiple, 110-10
  - one-to-one mappings, 110-5
  - parallel expressions, 110-11
  - parameterized, 110-8
  - platform functions, 110-16
  - query keys, 110-9
  - subqueries and subselects, 110-10
  - user-defined functions, 110-5, 110-16
  - using Boolean logic, 110-3
  - XML Type functions, 110-5
  - see also* queries
- external
  - applications, 115-26
  - connection pools, 96-8
  - controller, transaction, 113-1
  - JDBC pools, 2-40
  - transactions, 113-1
- external transaction controller
  - configuration, sessions, 89-17
  - session, 113-2

## F

---

- factory name, JMS connection, 104-3
- failure, resuming unit of work after commit, 115-14
- features of Oracle TopLink, 1-4
- fetch groups
  - about, 16-5, 119-88
  - configuring, 111-3, 119-88
  - default, 111-3, 119-88
  - disabling, 111-3
  - dynamic, 111-4
  - object level read query, 108-14
  - read optimization, and, 119-88
  - size, 12-22
  - static, 111-3
- field access
  - about, 2-12
- field references, 28-10
- Field uses XML Schema "type" attribute
  - option, 17-15
- fields in database tables, 5-27
- Fields tab, 37-2, 37-3
- field-to-object attribute transformation, 121-34, 121-36
- files
  - JAXB-specific, 47-4
  - see also* specific file name
- final attributes, 5-49, 5-53
- finalizers
  - sessions, 90-10
- findAll, using, 108-27
- finders
  - caching options, 108-38

- disabling cache, 108-39
- managing large result sets, 111-18
- refreshing results, 108-39
- see also* EJB finders
- flashback queries
  - about, 108-24
  - historical client sessions, 93-1
- `forceUpdateToVersionField()`, 115-17
- foreign keys
  - about, 2-19, 28-10
  - configuring in EIS mappings, 83-2
  - direct collection mappings, 36-3
  - EIS mappings, 84-2
  - multiple tables, 23-15
  - one-to-many mappings, 27-8
  - one-to-one mappings, 27-6, 121-42
  - parameterized expressions, 110-8
  - references, A-25, A-33
  - target, 27-6, 28-10
  - troubleshooting, A-25, A-33
- fragment XML mappings
  - configuring, 66-1
- full identity map, 102-3

## G

---

- garbage collection, managing, 114-11
- General Preferences dialog, 5-13
- Generate Classes and Descriptors dialog, 5-33
- Generate EJB Entity Classes and Descriptors dialog, 5-34
- generating
  - access method, 5-33
  - deprecated direct mappings, 20-13
  - see also* exporting
- `getCatalogs()`, 5-24
- `getField()`, 110-12
- `getImportedKeys()`, 5-24
- `getParameter()`, 110-8
- `getPrimaryKeys()`, 5-24
- `getTable()`, 110-12
- `getTables()`, 5-24
- `getTableTypes()`, 5-24
- `getValue()`, 17-8, Glossary-8
- `getValue()` method, 17-8, Glossary-8

## H

---

- hard cache weak identity map
  - about, 102-4
  - when to use, 102-5
- help
  - about, 5-12
  - displaying, 5-12
- Help button., 5-12
- Help Preferences, 5-14, 5-16
- hierarchical queries
  - about, 108-24
  - described, 111-7
- hints, Oracle Hints in queries, 108-24
- historical client sessions

- about, 87-2, 87-25
- cache, 87-34
- limitations of, 87-26
- historical queries, 111-2
  - about, 108-21
  - see also* AsOfClause
- history policy, configuring, 119-84
- holders, value, 17-8, Glossary-8
- host URL, JMS topic, 104-4
- hosted
  - documentation, 5-14
  - XSD files, 9-2
- hot deployment
  - about, 11-3
  - CMP applications, 11-3
  - non-CMP applications, 11-3
- HSQL platform, 96-3

## I

---

- IBM Informix Database native sequencing, 18-8
- IBM WebSphere
  - setting class loader order, 8-17
- identity
  - about, 2-19, 102-3
  - cache, and, 102-3
  - using cache to preserve, 102-3
  - see also* identity map
- identity map cache
  - disabling during a write query, 109-13
  - refresh in read query, 108-36
- identity maps
  - about, 87-30, 119-40, 119-42, 119-45
  - cascading refresh during read query, 108-37
  - descriptors, 117-15, 119-40
  - example, 108-36
  - full, 102-3
  - guidelines for choosing type, 102-4
  - hard cache weak identity map, 102-4, 102-5
  - isolated client sessions, 87-25
  - no identity map, 102-4
  - refreshing during read query, 108-36
  - soft, 102-4
  - soft cache weak identity map, 102-4, 102-5
  - soft cache weak identity map and read optimization, 12-25
  - specifying, 119-40, 119-42, 119-45
  - weak, 102-4
  - weak identity map and read optimization, 12-25
- Identity tab. *see* Caching tab
- impedance mismatch, solving, 1-3
- Implementors tab, 119-38
- Import Tables from Database dialog, 5-24
- importing
  - classes, 5-16
  - preferences, 5-12
- inactive descriptors
  - about, 5-10
  - mapping to, 28-7, 78-4
- independent relationships, 121-18

- indexed records, 77-3
- indirection
  - about, 2-21, 17-6
  - bidirectional relationships, 121-42
  - choosing the correct type, 121-5
  - configuring, 121-4
  - EJB, 17-11
  - EJB 2.x CMP, 17-11
  - example, 17-7
  - many-to-many mappings, 27-10
  - nontransparent, 2-14
  - one-to-many mappings, 121-42
  - proxy indirection, 17-10
  - remote sessions, 87-33
  - serialization, 17-11
  - serialization and JPA, 17-12
  - transparent, 2-14, 17-9, Glossary-8
  - value holder, 17-8, Glossary-8
  - ValueHolderInterface, 2-14
  - see also* proxy indirection, transparent indirection
- Informix platform, 96-3
- inheritance
  - about, 2-19, 16-3, 16-9, 21-4
  - aggregate classes, 16-12
  - aggregate collection mappings, 27-11
  - branch classes, 16-9, 16-10
  - child descriptors, 119-57
  - class extraction, 16-11, 16-12
  - class indicator, 16-10, 16-11
  - descriptors, 16-3, 16-9, 21-4, 119-57, 119-59
  - finding subclasses, 16-10
  - instantiating subclasses, 16-10
  - isolated client sessions, 87-25
  - leaf classes, 16-9, 16-10, 111-5
  - outer-join subclasses, 12-19
  - primary keys, 16-12, 21-4, 74-2
  - queries, 108-22
  - querying on hierarchy, 111-5
  - relational parent, 119-59
  - root class, 16-9, 119-58, 119-59
  - root class subclasses, finding in inheritance, 16-10
  - supporting with multiple tables, 21-5
  - supporting with one table, 21-4
  - transformed to relational model, 12-14
  - using with EJB, 16-5, 16-13
- inheritance hierarchies
  - descriptors, 16-19
  - querying on, 111-5
- Inheritance tab, 119-58, 119-59
- inherited subclasses, mapping, 119-65
- in-memory query
  - about, 102-6
  - check cache using exact primary key, 108-31
  - check cache using primary key, 108-31
  - check database if not in cache, 108-31
  - conform results in unit of work, 108-32
  - expression limitations, 108-32
  - supported, 108-32
  - using, 108-31
- inner join, 110-6
- insert operation, 109-3, 109-4
- insertObject(), 23-8
- instantiation policy
  - about, 119-78
  - setting, 119-78
- Instantiation tab, 119-78
- integrity checker, 118-1
  - about, 87-12
  - configuring, 89-21
- interactions
  - about, 109-32
  - creating, 76-6
- interface alias
  - about, 23-11
  - creating, 23-12
- Interface Alias tab, 23-12
- interfaces
  - classes, implementing, 5-47
  - customizing, 5-13
  - descriptors, 22-2, 119-35, 119-37
  - queries, 108-22
  - query keys, 119-35
  - querying on, 111-4
- internal connection pool
  - about, 96-7
  - named, 96-9
  - read, 96-8
  - sequence, 96-8
  - write, 96-8
- internal query object cache
  - about, 108-37
  - configuring, 111-20, 111-21
  - expiration, 111-21
  - restrictions, 108-38
- internal transactions, 113-1
- invalidation of objects in the cache, 102-8, 117-22, 119-47
- IP address for multicast group, 103-5
- isolated client sessions
  - about, 87-2, 87-20, 92-1
  - configuration, 92-1
  - life cycle, 87-23
  - limitations of, 87-24
  - session event handlers, 87-22
  - with Oracle Virtual Private Database (VPD), 87-22
- isolated session
  - cache, 87-34
  - ConnectionPolicy, 87-21
  - exclusive connections, 87-21
  - supported databases, 87-21
- isolation
  - cache, 102-6
  - transaction levels, 113-2
  - unit of work transactions, 113-4
- Iterator interface, 111-16

**J**

---

Java

- database tables, 19-4
- exporting to, 19-3
- integration with any datasource, 1-3
- iterators, 111-16
- object model, 2-11
- Java applications
  - deploying, 11-1
  - deployment files, 9-7
  - packaging for deployment, 10-1
- Java Cryptography Extension, 116-9
- <Java EE-Container>-jar.xml file
  - EJB 3.0, 9-6
- Java EE
  - parameter binding, 12-20
  - prepared statement caching, 12-20
  - web applications, 1-5
- Java Management Extensions. *see* JMX
- Java Naming and Directory Interface. *See* JNDI
- Java Object Builder, 87-4
- Java streams
  - described, 111-17
  - optimizing, 111-18
  - support for, 111-17
- Java Transaction API
  - and unit of work commit, 113-10, 113-11
  - and unit of work rollback, 113-11
  - see also* JTA
- Java Transaction Service *see* JTA
- JAVA\_HOME, 5-2
- javaagent, 2-26
- JAVA-EE-CONTAINER-*ejb-jar.xml* file, 9-6
- java.util.Collection interface, 121-30
- java.util.Map interface, 121-30
- java.util.Set interface, 121-30
- javax.ejb.EntityBean interface, 5-34
- JAXB
  - creating projects from, 48-2
  - files, 47-4
  - generating project from the command line, 48-3
  - jaxb:class, and EIS mappings, 77-3
  - jaxb:class, and XML mappings, 53-6
  - proxy configuration, 48-3
  - tljxb.cmd<Default Para>, 48-3
  - tljxb.sh<Default Para>, 48-3
  - typesafe enumeration converter, 121-29
  - understanding, 47-2
  - validation, 47-9
  - XML projects, 47-2
- JAXB typesafe enumeration converter
  - configuring, 121-29
  - understanding, 17-27
- JAXBContext, 47-8
- JCA adapters
  - about, 96-4
  - configuring for Oracle TopLink Workbench, 5-2
  - EISLogin, 96-2
  - selecting, 96-4
  - with EIS, 116-3
- JCE. *see* Java Cryptography Extension
- JConnect, 12-18
- JDBC
  - adaptor for EIS, 77-5
  - database gateway for EIS, 77-5
  - driver classpath, 5-24
  - JConnect, 12-18
  - Sybase JConnect, 12-18
- JDBC drivers
  - about, 96-3
  - configuring for Oracle TopLink Workbench, 5-2
  - fetch size, 12-17, 12-22
  - general properties, 12-17
  - mapping LOBs, 120-3
  - selecting, 96-3
- JDBC pools
  - external with EJB Entity Beans with BMP
    - architecture, 2-40
- JDBC\_CLASSPATH, 5-2
- JMS
  - connection factory name, 104-3
  - coordinated cache, 104-1
  - coordinated cache, connection handling, 104-5
  - coordinated cache, error receiving message, 104-5
  - coordinated cache, ERROR\_RECEIVED\_JMS\_MESSAGE\_IS\_NULL, 104-5
  - coordinated cache, ERROR\_RECEIVING\_JMS\_MESSAGE, 104-5
  - coordinated cache, null JMS message, 104-5
  - Processing Exceptions, A-7
  - topic host URL, 104-4
  - topic name, 104-2
- JMX
  - about
  - and DMS profiler
  - DMS profiler, 12-9
- JNDI naming service, 103-9
- joining
  - about, 108-12
  - addJoinedAttribute, 109-11
  - expressions, and, 110-6
  - inner joins, 110-6
  - mappings, and, 28-13
  - ObjectLevelReadQuery, 109-11
  - ObjectLevelReadQuery, and, 109-11
  - one-to-many, 109-11
  - one-to-many, filtering duplicates, 109-11
  - one-to-many, when not to use, 108-13
  - one-to-one, 109-11
  - optimizing reads, 12-25
  - outer-join inherited subclasses, 12-19
  - QueryManager expressions, 111-5
  - read queries, 12-25, 109-11
  - reading and object-level read queries, 108-12
  - reference mappings, 28-13
  - setAdditionalJoinExpression(), 111-5
  - setMultipleTableJoinExpression(), 111-5
  - use inner join, 108-13, 109-11
  - use outer join, 108-13, 109-11
- JPA
  - indirection and serialization, 17-12

- schema generation, 6-6
- schema manager, 6-6
- serialization and indirection, 17-12
- JPQL, 109-16
- JSP and Servlet applications
  - deploying, 11-1
  - deployment files, 9-7
  - packaging for deployment, 10-2
- JTA
  - about, 113-3
  - and unit of work, 113-3
  - isolated client sessions, 87-25
  - unit of work, 113-1
- JTA/JTS
  - using with EJB Entity Beans with BMP architecture, 2-40
- just-in-time reading. *See* indirection

## K

---

- Key Converter tab, 38-4
- key pairs
  - database table reference, 5-30
  - troubleshooting, A-25, A-33
- keys
  - about, 2-19
  - composite primary, 121-42
  - foreign, 2-19, 27-6, 121-42
  - foreign, target, 27-6
  - inheritance, 16-12, 21-4, 74-2
  - multiple tables, 23-14
  - primary, 16-12, 21-4, 23-14, 27-7, 27-10, 32-5, 74-2, 121-3, 121-42
  - read-only settings, 121-3
  - reference key field, 27-12, 27-13
  - variable class relationships, 27-7, 32-5

## L

---

- language, specifying, 5-3
- large result sets, managing in finders, 111-18
- lazy attribute loading and read optimization, 119-88
- lazy connection
  - acquisition, 89-23
  - allocation, 87-19, 90-8
- lazy loading
  - See* fetch groups
  - See* indirection
- lazy reading. *See* indirection
- leaf classes, 16-9, 16-10, 111-5
- life cycle of unit of work, 113-5
- LOB
  - mapping to, 120-3
- local
  - documentation, 5-14
  - transactions, 115-25
- locked files, 7-6
- locking policy
  - AllFieldsLockingPolicy, 16-16
  - ChangedFieldsLockingPolicy, 16-16
  - configuring, 119-71

- field locking, 16-16
- optimistic, 16-7, 16-14, 16-16
- optimistic version locking, 16-13
- optimistic version locking, stored function restrictions, 16-14
- optimistic version locking, stored procedure restrictions, 16-14, 109-27
- OptimisticLockException, 16-14, 16-17
- pessimistic locking policy, 16-7, 16-17
- SelectedFieldsLockingPolicy, 16-17
- stale data, and, 102-7
- three-tier architectures, optimistic locking and, 16-18
- three-tier architectures, pessimistic locking and, 16-18
- TimestampLockingPolicy, 16-14
- understanding, 16-7, 16-13
- version locking, 16-14
- VersionLockingPolicy, 16-13
- Locking tab, 119-72
- log into database, 5-23
- Log Out of Database, 5-23
- log out of database, 5-23
- logging
  - application server, 87-10
  - chained exceptions, 87-11
  - configuring, non-Oracle Java EE container, 89-9
  - configuring, TopLink Workbench, 89-6
  - configuring, using java.util.logging, 89-9
  - configuring, using Oracle Enterprise Manager, 89-9
  - configuring, using Session API in Java, 89-8
  - java.util.logging, 87-9
  - log level, 87-10
  - logging.properties file, 87-11
  - MaD, configuration with Oracle Enterprise Manager, A-2
  - MaD, in-container defaults, A-2
  - MaD, out of container defaults, A-2
  - non-Oracle, in-container defaults, 87-11
  - out of container defaults, 87-11
  - output, 87-10
  - permissions, 8-21
  - sessions, 87-8, 89-5, 89-6
  - TopLink native logging, 87-8
  - types, 87-8
- Logging tab, 89-6
- logging.properties file, 87-11
- login
  - CMP deployment, 15-3
  - database, 20-6, 20-8, 98-1
  - deployment, 15-3
  - development, 15-4
  - platforms, and, 15-4
  - projects, and, 15-3, 86-2, 96-2
  - role in project, 15-3, 96-2
  - session, 15-3, 89-5
  - session role, non-CMP, 15-3
- logMessages method, 87-8
- look and feel, specifying, 5-13

## M

---

### MaD

- about, A-1
- DMS, about, A-2
- DMS, TopLink instrumentation, 12-4
- integrating with, 8-16
- JMX, about, A-2
- logging, about, A-1
- logging, configuration with Oracle Enterprise Manager, A-2
- logging, in-container defaults, A-2
- logging, out of container defaults, A-2
- Manage Non-Descriptor Classes dialog, 5-57
- Manageability and Diagnosability. *see* MaD
- management, source control, 7-3
- manager, session events, 87-6
- many-to-many mappings
  - relation table, 27-10
  - relation tables, 34-2
  - session broker limitations, 87-28
- many-to-many relational mappings
  - configuring, 34-1
  - EJB, 27-10
  - understanding, 27-9
- mapped records, 77-3
- mapping extensions
  - custom data types, 13-1
  - JAXB typesafe enumeration converter, 17-27, 121-29
  - object type converter, 17-14, 121-25
  - serialized object converter, 17-12, 121-21
  - simple type translator, 17-15, 121-28
  - transformation mappings, 17-17
  - type conversion converter, 17-13, 121-23
- mappings
  - about, 2-18, 2-19, 14-1, 14-2, 17-1, 120-1
  - access types, 121-18
  - aggregate collection mappings and EJB, 27-11
  - anyType mapping, 54-5
  - as part of the application development process, 2-24
  - automatic, 120-3
  - batch options, 28-8
  - bidirectional relationships in Java, 121-42
  - CDATA, 121-44
  - class hierarchy, 17-28
  - comments, 121-21
  - configuring, 120-2, 121-1, 121-2
  - database field, 28-4
  - default mapping, 17-5
  - deprecated, generating, 20-13
  - direct access, 12-16, 121-16
  - directionality, 27-2
  - EIS mappings, 17-29, 71-2, 77-1, 78-1
  - EJB 2.0 entities, 117-7
  - errors, A-23
  - example, 17-4
  - extensions, about, 17-12
  - hierarchy, 17-28
  - inactive descriptors, 28-7, 78-4

- indirection, 12-16, 17-6, 121-4
- isolated client sessions, 87-24
- manually configuring, 120-2
- many-to-many, 27-10
- many-to-many, with EJB, 27-10
- method access, 121-16, 121-37
- null values, 121-14, 121-43
- object-relational data type, 40-1, 41-1
- one-to-many object, with EJB, 27-9
- one-to-one with EJB, 27-6
- optimizing, 12-16
- OX mappings, 17-29
- projects, and, 14-2
- read-only, 121-3, 121-4
- relation tables, 34-2
- relational, 27-1, 28-1
- removing, 120-5
- to tables, 23-3, 119-5
- TopLink Workbench, Navigator window, 5-9
- types of, 17-1
- use inner join, 108-13, 109-11
- use outer join, 108-13, 109-11
- XML mappings, 53-1, 54-1

mathematical functions, in TopLink expressions, 110-4

menu bar, 5-5

menus

- about, 5-4, 5-5
- context menus, 5-6
- menu bar, 5-5

merging

- changes in clones, 115-12
- Oracle TopLink Workbench project files, 7-4
- project files, 7-4

messages, error, A-2, A-10

metadata

- about, 2-14, 2-22
- advantages, 2-23
- creating, 2-24, 2-25
- mapping and configuration, 14-1
- project metadata, 2-24
- session metadata, 2-25

Metalink, 2-3

method access

- about, 2-12, 117-5, 121-16
- change tracking limitations, 2-13, 117-5, 121-17
- setting, 121-37

methods

- adding, 5-52
- getValue(), 17-8, Glossary-8
- setValue(), 17-8, Glossary-8
- wrapper policy, 119-87
- see also* specific method name

Microsoft Access

- platform, 96-3
- schema manager type conversion, 6-3

Microsoft SQL Database native sequencing, 18-8

migrating

- error messages, 8-15
- OC4J persistence to TopLink, 8-7

- Oracle TopLink Workbench projects, 116-5
- troubleshooting, 8-15
- Migration Utility Exception, A-9
- model source, exporting, 116-10
- modifiers, class, 5-46
- multicast group address, coordinated cache, 103-5
- multicast port, coordinated cache, 103-7
- multiple sessions, 87-29, 90-2
- multiple tables
  - about, 23-14
  - specifying for descriptors, 23-14
- multiplicity in relationships, resolving, 8-16
- multi-processing, 12-35
- Multitable Info tab, 23-14, 119-55
- mutability
  - about, 2-21
  - change policy, 113-9, 119-81
  - change tracking performance, 2-22
  - defaults, 2-21
  - EIS transformation mapping, and change policy, 113-9
  - transformation mapping, and change policy, 113-9
  - transformation mappings, 121-39
  - transformation mappings, and copy policy, 2-22, 113-9, 121-39
  - weaving, 2-22
- mutable mappings, 121-39
- .mwp file, 5-1, 116-2
- MySQL4
  - batch writing, 98-10
  - platform, 96-3
  - primary key restrictions, 18-4
  - schema manager type conversion, 6-3

## N

---

- named connection pools, 96-9
- named queries
  - about, 89-24, 108-17, 119-12
  - configuring, 89-24, 119-12
  - descriptor level, 119-12
  - exclusive connections, 119-27
  - options, advanced, 119-27
  - parameter binding, 119-25
  - prepared statement caching, 119-25
  - redirect query, 108-17
  - session level, 89-24
  - using, 109-18
  - when not to use, 108-17
  - when to use, 108-17
- namespaces
  - about, 15-5, 15-6
  - configuring, 5-42
- naming service
  - coordinated cache, 103-9
  - JNDI, 103-9
  - RMI, 103-13
- native batch writing, 12-19, 98-11
- native sequencing

- IBM Informix Database, 18-8
- Microsoft SQL Database, 18-8
- Microsoft SQL Server, 6-6
- non-Oracle database, 18-8
- Oracle Database, 18-7, 18-9
- Oracle Database SEQUENCE object, 18-8
- Sybase Database, 6-6, 18-8
- native SQL, 98-11
- Navigator window
  - about, 5-4, 5-9
  - attribute and mapping, 5-9
  - database, 5-10
  - database tables in, 5-22
  - descriptor, 5-9
  - example, 5-9
  - package, 5-9
  - project, 5-9
  - unsaved or changed item, 5-9
- NCHAR, 17-14
- NCLOB, 17-14
- neediness warnings. *See* troubleshooting
- nested table object-relational data type mappings
  - configuring, 46-1
  - understanding, 40-3
- nested unit of work, 113-10, 115-15
- new projects, 116-2
- New Reference dialog box, 5-30
- New Session button, 88-4, 88-7, 88-9
- New Sessions Configuration, 88-2
- New Table dialog box, 5-23
- newInstance method, 115-2
- no identity map, 102-4
- non-cascading write queries
  - compared to cascading, 108-15, 109-13
  - creating using dontCascadeParts () method, 108-15, 109-13
- non-deferred write
  - configuring, 115-11
  - dependent objects, 16-4
  - understanding, 16-3, 115-6, 115-25
- nonintrusive persistence, 2-20
- nonpersistent projects, 15-2
- nonrelational projects, 15-2
- nontransactional data sources, 96-1
- nontransparent indirection, 2-14
- nouns
  - DMS profiler, 12-4, 87-12
- null values
  - default, 97-8, 121-14, 121-43
  - in expressions, 110-14
  - login level, 97-8
  - mapping level, 121-14
- NVARCHAR2, 17-14

## O

---

- object array object-relational data type mappings
  - configuring, 45-1
  - understanding, 40-3
- object cache, 108-37

- object cache, sessions, 87-3
- object change tracking
  - configuring, 119-82
- object identity, 87-30
  - about, 2-19, 102-3
  - cache, and, 102-3
  - using cache to preserve, 102-3
  - see also* identity map
- object indirection
  - read optimization, as, 12-25
- object level modify query
  - about, 108-14, 108-16
- object level read query
  - about, 108-11, 108-14
  - addJoinedAttribute, one-to-many mappings, 109-11
  - addJoinedAttribute, one-to-one mappings, 109-11
  - fetch groups, 108-14
  - join reading, 108-12
  - joining one-to-many mappings, 109-11
  - joining one-to-many mappings, filtering duplicates, 109-11
  - joining one-to-one mappings, 109-11
  - partial object query, 108-11
  - read all query, 108-11
  - read object query, 108-11
  - setShouldFilterDuplicates, one-to-many mappings, 109-11
- object model
  - about, 2-18
  - generating with `tljxsb.cmd`, 48-4
  - optimization, 12-11
  - Oracle TopLink requirements, 2-11
- object reference XML mappings
  - configuring, 62-1
- object type converter
  - about, 13-2, 17-14
  - configuring, 121-25
- object type mappings
  - configuring, 121-25, 121-26
- ObjectLevelChangeTrackingPolicy, 113-7
- object-relational data type descriptors
  - configuring, 26-1
  - locking policy, 119-71
  - understanding, 24-1
- object-relational data type mappings
  - about, 40-1
  - array, understanding, 40-3
  - configuring, 41-1
  - nested table, 40-3, 46-1
  - object array, 40-3, 45-1
  - overview, 2-18
  - reference, 40-2, 43-1, 44-1
  - structure, 40-2, 42-1
- object-relational data type projects
  - about, 18-2
  - sequencing, 15-5
- objects
  - cascading refresh in cache, 108-36
  - creating and registering, 115-2
  - query, 109-6
  - refreshing in cache, 108-36
  - registering and unregistering, 115-1
- ObjectTypeMapping
  - see* ObjectTypeConverter
- one-to-many EIS mappings
  - configuring, 84-1
  - key on source, 77-14
  - key on target, 77-16
  - understanding, 77-13
- one-to-many relational mappings
  - configuring, 33-1
  - understanding, 27-8
- one-to-one EIS mappings
  - configuring, 83-1
  - key on source, 77-10
  - key on target, 77-11
  - understanding, 77-9
- one-to-one relational mappings
  - configuring, 31-1
  - expressions, 110-5
  - understanding, 27-5
  - with EJB, 27-6
- online help, 5-14
- Open Project button, 116-5
- opening projects, 116-5
- operators
  - boolean logic, 110-3
- optimistic locking
  - about, 16-7
  - application architecture, 2-11
  - cascading locking policy, 16-14, 119-75
  - database exception, 109-33
  - exceptions, A-5
  - field locking policy, about, 16-16
  - rollbacks, 16-16
  - version locking policy, 16-13, 16-14, 119-75
  - version locking policy, stored function restrictions, 16-14
  - version locking policy, stored procedure restrictions, 16-14, 109-27
  - with `forceUpdateToVersionField()` method, 115-17
- optimistic locking policy
  - field locking, about, 16-16
  - version locking, 16-13, 16-14, 119-75
  - version locking, stored function restrictions, 16-14
  - version locking, stored procedure restrictions, 16-14, 109-27
- OptimisticLockException, 16-14, 16-17
- optimization
  - about, 12-1
  - application bottlenecks, 12-2
  - application server, 12-36
  - batch reading, 12-21
  - batch writing, 12-18
  - CMP partial object queries, 12-22
  - code generation, 12-11
  - cursor streams, 12-23



- data access, 12-17, 12-18
- data format, 12-18
- database, 12-36
- descriptors, 12-16
- DMS profiler, 12-4, 87-12
- existence checking, 12-16
- fetch groups, 12-22
- fetch size, JDBC, 12-22
- general, 12-11
- inheritance, 21-5
- instantiation, 12-16
- JDBC driver, 12-17, 12-22
- join reading, 12-21
- mappings, 12-16
- named queries, 12-21
- object model, 12-11
- outer-join inherited subclasses, 12-19
- overview, 2-17
- pagination, 12-24
- parameter binding, 12-19
- parameterized SQL, 12-21
- partial object queries, 12-22
- prepared statement, 12-21
- prepared statement caching, 12-19
- profiler, 12-4, 87-12
  - optimization
    - TopLink Profiler, 12-2
- queries, 12-21
- reading, 12-24
- read-only queries, 12-22
- read-only query, 12-22
- ReadQuery method setMaxRows, 12-23
- result set pagination, 12-24
- schema, 12-11
- scrollable cursors, 12-23
- setMaxRows, 12-23
- understanding, 12-1
- unit of work, 12-35
- weaving, 12-36
- writing, 12-32
- Oracle
  - development support, 2-3
  - remote session support, 87-32
- Oracle Containers for J2EE
  - migrating to TopLink, 8-7
- Oracle Database
  - date and timestamp mappings, 28-4
  - native sequencing, 18-9
    - Oracle Database
      - SEQUENCE object, 18-8
  - platform, 96-3
  - schema manager type conversion, 6-3
- Oracle extensions
  - hierarchical queries, 111-7
  - Oracle Hints, 111-6
- Oracle Hints, using with TopLink queries, 111-6
- Oracle JDeveloper
  - configuring with Oracle TopLink, 7-1
- Oracle TopLink
  - about, 1-1, 5-1
  - application architectures, 1-5
  - architectures, 1-1
  - development, 2-1, 3-1
  - features, 1-4
  - integrating with application server, 8-1, 8-2
  - mapping types, 17-1
  - optimization, 12-1
  - packaging your application, 10-1
  - public source, 13-2
  - runtime components, 3-3
  - understanding, 1-1
- Oracle TopLink Sessions Editor. *see* sessions
- Oracle TopLink Workbench
  - about, 5-1
  - Ant integration, 5-58
  - classpath, 5-2
  - creating projects, 19-1, 48-1, 72-1, 116-1
  - development process, 5-1
  - DRIVER\_CLASSPATH, 5-2
  - environment, 5-2
  - error messages, A-10, A-12
  - JDBC\_CLASSPATH, 5-2
  - parts of, 5-3
  - preferences, 5-12
  - project, 5-1, 116-2
  - proxy, 5-13, 5-14, 19-6
  - sample, 5-3
  - table creator, creating, 6-4
  - upgrading projects, 116-5
- Oracle Virtual Private Database (VPD)
  - isolated client sessions, 87-22
  - proxy authentication, 87-22, 96-6, 98-14
- oracle.sql.TimeStamp, 28-4
- order
  - query keys, 28-8
  - relational mappings, 28-8
- OrderSibling, 111-7
- orion-ejb-jar.xml file
  - about, 9-10
  - entity-deployment attribute pm-name, 9-10
  - modifying for Oracle TopLink, 9-10
  - persistence-manager attribute
    - class-name, 9-10
  - persistence-manager attribute
    - descriptor, 9-10
  - persistence-manager attribute name, 9-10
  - persistence-manager subentry
    - forpm-properties
      - cache-synchronizations, 9-11
      - customization-class, 9-11
      - db-platform-class, 9-11
      - default-mapping, 9-11
      - project-class, 9-11
      - remote-relationships, 9-11
      - session-name, 9-11
- orion-ejb-jar.xml file, 9-6
- OTN (Oracle Technology Network), 1-4, 2-3
- OTS (Object Transaction Service)
  - about, 113-3
  - unit of work, 113-3

- outer joins
  - inheritance, 119-55
- OX mappings
  - about, 17-29
  - extensions, simple type translator, 17-16
  - read conversions, 17-16
  - write conversions, 17-16

## P

---

- package names
  - generating, 5-33
  - renaming, 5-57
  - TopLink Workbench, Navigator window, 5-9
  - see also* classes
- packaging
  - JPA application for weaving, 10-5
  - POJO application for weaving, 2-27, 10-5
- packaging for deployment
  - about, 10-1
  - BMP applications, 10-6
  - CMP applications, 10-5
  - EJB 3.0, 10-1
  - Java applications, 10-1
  - JSP and Servlet applications, 10-2
  - Session Bean applications, 10-3
- packet time-to-live cache coordination, 103-19
- pagination, 111-20
- parallel
  - expressions, 110-11
  - unit of work, 113-10
- parameter binding
  - about, 12-19
  - byte arrays, 12-20
  - configuring, 20-8, 98-10, 109-17, 119-25
  - database login level, 98-10
  - descriptor level, 119-25
  - external connection pools, 12-20
  - internal connection pools, 12-20
  - Java EE, 12-20
  - named queries, 119-25
  - optimizing, 12-19
  - project level, 20-8
  - queries, 109-17
  - streams, 12-20
  - strings, 12-20
  - trouble shooting, 12-20
- parameterized batch writing
  - about, 12-18
  - `setMaxBatchWritingSize()`, 12-18
- parameterized expressions
  - about, 110-8
  - example, 110-9
- parameterized SQL
  - enabling on queries, 109-17
  - Oracle TopLink optimization features, 12-32
  - See also* parameter binding
- parser conflicts, XML, 8-3
- partial object reading optimization, 12-25
- password encryption

- customizing, 97-3
- passwords, encryption, 116-9
- performance optimization
  - about, 12-1
  - application bottleneck, 12-2
  - examples, 12-26
  - JConnect method `isClosed`, 12-18
  - using Performance Profiler, 12-2
- Performance Profiler
  - about, 12-2
  - class, 12-3
- persistence
  - about, 2-20
  - by reachability, 114-8
  - components of, 2-14
  - descriptor, 9-9
  - implementation options, about, 2-11
  - implementation options, JPA annotations, 2-12
  - implementation options, JPA metadata, 2-12
  - implementation options, JPA XML, 2-12
  - implementation options, method and direct field access, 2-12
  - implementation options, TopLink metadata Java API, 2-12
  - implementation options, TopLink metadata XML, 2-12
  - implementation options, weaving, 2-13
  - manager, 8-4
  - OC4J, 8-7
  - projects, 15-2
  - types, 117-6
  - using a persistence layer, 2-16
- persistence manager
  - default, 8-4
  - migration, 8-5
  - restrictions, 8-4
- persistent classes
  - project, 5-35
  - requirements, 2-11, 2-14
  - types, 117-8
- pessimistic locking
  - about, 16-7, 16-17
  - application architecture, 2-11
  - policy, 16-17
- phantom reads, preventing, 115-30
- platforms
  - Attunity Connect database, 96-3
  - Cloudscape database, 96-3
  - data source, 96-3
  - database, 20-3, 73-2, 96-3, 98-2, 99-1, 116-3
  - DB2 database, 96-3
  - EIS, 96-4
  - functions in expressions, 110-16
  - HSQL database, 96-3
  - Informix database, 96-3
  - JCA adapter, 96-4
  - JDBC drivers, 96-3
  - Microsoft Access database, 96-3
  - MySQL4 database, 96-3
  - Oracle database, 96-3

- Oracle8 database, 96-3
- Oracle9 database, 96-3
- parser, XML, 8-3
- PointBase database, 96-3
- projects, and, 15-4
- server, 89-18
- session configuration, 89-18
- SQLAnywhere database, 96-3
- SQLServer database, 96-3
- SybasePlatform database, 96-3
- XML parser, 8-3
- see also* target platforms
- PointBase platform, 96-3
- pop-up menus. *see* context menus
- ports
  - multicast group, 103-7
  - permissions, 8-20
- post-commit clones, avoiding, 115-31
- Potential EJB Descriptors dialog box, 116-5
- pre-allocating sequence numbers, 18-9, 20-4, 98-6, 99-2
- preferences
  - class import, 5-16
  - database, 5-18
  - EJB, 5-17
  - general, 5-13
  - help, 5-14
  - importing and exporting, 5-13
  - Oracle TopLink Workbench, 5-12
  - sessions, 5-19
- Preferences - Class dialog, 5-16
- Preferences - EJB dialog, 5-17
- Preferences - General dialog, 5-13
- Preferences - Help dialog, 5-14
- Preferences - Mappings dialog, 5-15
- Preferences dialog, 5-13
- prepared statement caching
  - about, 12-19
  - configuring, 20-8, 98-10, 109-17, 119-25
  - database login level, 98-10
  - descriptor level, 119-25
  - external connection pools, 12-20
  - internal connection pools, 12-20
  - Java EE, 12-20
  - named queries, 119-25
  - optimizing, 12-19
  - project level, 20-8
  - queries, 109-17
  - query level, 109-17
- preserving XML documents, 52-6
- primary key
  - about, 2-19
  - cache, 108-35
  - composite, 27-10, 121-42
  - inheritance, 16-12, 21-4, 74-2
  - multiple tables, 23-14
  - primkey in ejb-jar.xml file, 19-4
  - queries with compound, 108-35
  - read-only settings, 121-3
  - restrictions, 18-4
  - setting, 5-28, 23-3, 119-5
  - unit of work, 113-11
  - unknown, 119-53
  - variable class relationships, 27-7, 32-5
- private relationships, 121-18
- Problems window
  - about, 5-5, 5-11
  - sample, 5-11
  - see also* error messages
- profiler
  - about, 89-12
  - development tool, 12-2
  - DMS, 12-4, 87-12
  - Oracle TopLink, 12-2, 12-3, 87-12
  - selecting, 89-13, 89-16, 89-20, 89-23
- Project - Multiple Projects tab, 89-11
- Project Status Report dialog box, 116-8
- projects
  - about, 14-1, 15-1, 15-2, 116-2
  - architecture, 15-2
  - cache type and size, 117-14, 119-39
  - comments, 117-23
  - configuring, 117-1
  - copying objects, 7-6
  - creating, 19-1, 48-1, 48-2, 72-1, 116-1, 116-2, 116-3
  - deployment login, and, 15-3
  - deployment overview, 2-17
  - descriptors, 14-2
  - development login, and, 15-4
  - direct access to mapped fields, 117-5
  - EIS, 71-1, 72-1, 72-2, 73-1
  - errors, A-12
  - existence checking, 117-10
  - exporting, 116-9, 116-10
  - for sessions, 89-11
  - indexed records, 72-2
  - Java, 116-3
  - JAXB, 48-2
  - locked, 7-6
  - login, 15-3, 86-2, 96-2
  - login, and, 15-3
  - mapped field access, default, 117-5
  - mapped records, 72-2
  - mapping projects, creating, 116-2
  - mappings, 14-2, 116-2
  - merging, 7-4
  - metadata, 2-24
  - method access to mapped fields, 117-5
  - model, exporting, 116-10
  - .mwp file, 5-1
  - nondesoriptor classes, 5-57
  - non-persistent, 15-2
  - nonrelational, 15-2
  - object-relational data type, types supported, 18-2
  - open, 116-5
  - Oracle TopLink Workbench, 116-2
  - packages, renaming, 5-57
  - persistence type, 15-2, 117-8
  - platforms, and, 15-4
  - prior TopLink versions, 116-5

- recently opened, 116-5
- relational, 15-2, 18-1, 18-2, 20-1
- renaming, 116-8
- reopening, 116-5
- saving, 116-6
- sequencing, 15-4, 15-5, 20-3, 98-5
- session login, and, 15-3
- sharing, 7-6
- status report, 116-8
- team development, 7-3
- TopLink Workbench, Navigator window, 5-9
- types of, 15-1, 16-1, 116-2
- updating from `ejb-jar.xml`, 19-6
- upgrading from 2.x or 3.x, 116-5
- writing `ejb-jar.xml` file, 19-5
- XML, 47-1, 49-1
- XML records, 72-1
- `projects.xml` file
  - about, 9-2
  - EJB 3.0, 9-4, 9-5
- propagation mode, cache, 103-2
- proxies. *see* wrapper policy
- proxy authentication
  - about, 96-5
  - applications, 96-5
  - Oracle Virtual Private Database, 87-22, 96-6, 98-14
  - session events, 87-6
  - use cases, 96-5
- proxy indirection
  - about, 17-10, Glossary-7
  - example, 121-10
  - restrictions, 17-10
- proxy settings, preferences, 5-13
- public source code, 13-2

## Q

- qualified names, database tables, 5-24, 5-25
- queries
  - about, 108-1
  - application development process, 2-15, 86-2
  - building, 108-6
  - cache, 108-30
  - Call queries, 108-3, 108-17
  - cascading, 108-15, 109-13
  - concepts, 108-2
  - conforming, 115-7
  - database queries, 108-10, 109-4
  - DatabaseQuery, 108-3
  - descriptor query manager, 108-3
  - EIS interactions, 108-6
  - EJB finders, 108-25
  - EJB QL query language, 108-5
  - `ejb-jar.xml` file, 23-7, 76-6, 119-12
  - executing, 108-7
  - expressions, 108-4
  - fetch groups, 108-14
  - flashback queries, 93-1, 108-24
  - hierarchical queries, Oracle extensions, 108-24

- hints, Oracle extensions, 108-24
- historical, 108-21, 111-2
- interface and inheritance queries, 108-22
- joining, 108-12
- languages, about, 108-4
- named queries, 108-17
- object level modify query, 108-14, 108-16
- object level read query, 108-11, 108-14
- on inheritance hierarchies, 111-5
- on interfaces, 111-4
- optimizing, 12-21
- Oracle database features, 111-6
- Oracle extensions, 108-24
- pagination, 111-20
- parameter binding, 109-17
- partial object query, 108-11
- performance, 12-21
- prepared statement caching, 109-17
- query keys, 108-4
- read all query, 108-11
- read object query, 108-11
- read-only, about, 108-12
- read-only, configuring, 111-4
- read-only, optimization, 12-26
- redirect queries, 108-20
- remote sessions, 87-33
- report, 12-27, 108-8
- report query, 108-16
- result set pagination, 111-20
- results, 108-8
- returning policy, 108-18
- session queries, 108-9, 109-1
- SQL query language, 108-5
- stored functions, Oracle extensions, 108-24
- subqueries, 110-10
- summary queries, 108-3
- timeout, 109-10, 119-27, 119-29
- types, 108-1
- UpdateAll, 108-15
- XML query language, 108-6
- Queries tab, 119-30, 119-31
- query by example, 109-6
- query cache, 108-37
- Query Exception, A-5
- query keys
  - about, 108-4, 119-33, 119-37
  - adding, 119-35, 119-38
  - and expressions, 110-9
  - creating, 119-35, 119-38
  - direct mappings, 119-34
  - generating, 119-34
  - in expressions, 110-9
  - interface descriptors, 119-35, 119-37
  - Java implementation, 119-35
  - modifying, 119-34
  - order, 28-8
  - relationship mappings, 119-34
  - specifying, 119-35, 119-38
  - unmapped attributes, 119-34, 119-35
- Query Keys tab, 119-35

- query object query. *See* DatabaseQuery
- query objects
  - batch reading, 109-10
  - cache expiration, 111-21
  - caching results, 108-37, 111-20
  - examples, 109-5
  - ordering for ReadAll queries, 109-8
  - report query, 12-27
  - specifying collection class, 109-9
- query results
  - about, 108-8
  - caching, 108-37
  - collections, 108-8
  - cursors, 108-9
  - reports, 108-8
  - streams, 108-9
- query timeout example, 109-10
- QueryManager
  - about, 16-6
  - joining expressions, 111-5
- QuerySequence, 98-8

**R**

---

- read access
  - providing in sessions, 87-16
- read all operation, 109-2
- read conversions
  - simple type translator, 17-16
- read operation, 109-2
- read optimization
  - about, 12-24
  - batch reading, 12-25
  - fetch groups, 119-88
  - joining, 12-25
  - lazy attribute loading, 119-88
  - object indirection, 12-25
  - partial object reading, 12-25
  - read-only query, 12-26
  - report query, 12-25
  - soft cache weak identity map, 12-25
  - unit of work, 12-25
  - weak identity map, 12-25
- read queries
  - cascading refresh of identity maps, 108-37
  - identity map cache refresh, 108-36
  - refreshing identity maps, 108-36
- ReadAll finders
  - executing, 111-10
- ReadAll queries
  - ordering in query objects, 109-8
- readAllObjects()
  - about, 23-9
  - example, 109-2
- reading
  - ejb-jar.xml, 19-6
  - just-in-time reading, 17-9, Glossary-8
  - whole XML documents, 30-2
- reading through the write connection, 115-27
- read-locking, 16-7

- readObject()
  - example, 109-2
- read-only
  - descriptors, 119-6
  - files, 7-6
  - mappings, 121-3, 121-4
  - queries, configuring, 111-4
  - queries, optimization, 12-26
- read-only queries
  - about, 108-12
- read-only query
  - read optimization, and, 12-26
- recently opened projects, 116-5
- redirect queries
  - about, 108-20, 111-1
  - creating, 111-1
  - finders, 111-13
  - using, 111-12
- reference key field, 27-12, 27-13
- reference object-relational data type mappings
  - configuring, 43-1, 44-1
  - understanding, 40-2
- reference relational mappings
  - joining, 28-13
- ReferenceMapping class, 40-2
- references
  - database tables, 5-29, 5-30
  - foreign keys, A-25, A-33
- Refresh from Database button, 5-26
- refresh policy
  - cache, 102-8
  - EJB finders, 102-8
- refreshing
  - cache, 119-30
  - classes, 5-56
  - refreshObject(), 109-2
  - remote sessions, 87-33
  - sessions, 90-5
- registering objects, 115-1
- registerNewObject method, 114-8, 115-5
- registerObject method, 114-5, 114-6, 114-8, 115-5
- reimporting schemas, 5-39
- relation tables
  - about, 27-10
  - many-to-many mappings, 34-2
  - mappings, 34-2
- relational descriptors
  - associated table, 23-3
  - configuring, 23-1
  - locking policy, 119-71
  - understanding, 21-1
- relational mappings
  - about, 27-1
  - aggregate collection, 27-10, 35-1
  - aggregate object, 27-13, 37-1
  - configuring, 28-1
  - converters, 27-3
  - direct collection, 27-12, 36-1
  - direct map, 27-13, 38-1

- direct-to-field, 27-4, 29-1
- direct-to-XMLType, 27-5, 30-1
- many-to-many, 27-9, 34-1
- one-to-many, 27-8, 33-1
- one-to-one, 27-5, 31-1
- options, 28-1
- order, 28-8
- transformation, 27-3, 27-16, 39-1
- variable one-to-one, 27-7, 32-1
- relational projects
  - about, 15-2
  - configuring, 20-1
  - exporting, 19-3
  - introduction, 18-1
  - object-relational data type databases, 18-2
  - relational databases, 18-1
  - understanding, 18-1
- relationships
  - about, 2-19
  - bidirectional, 5-34, 27-6, 121-42
  - expressions, 110-6
  - in `ejb-jar.xml` file, 19-5
  - unexpected multiplicity, 8-16
- relative locations
  - about, 117-2
  - classpath, 117-4
- Remote Command Manager Exception, A-8
- remote connection using RMI, example, 88-11
- remote sessions
  - about, 87-2, 87-30
  - application layer, 87-32
  - creating, 88-11
  - limitations of, 87-32
  - securing access, 87-33
  - server layer, 87-32
  - transport layer, 87-32
  - unit of work, 87-34
- Remove Class button, 5-56
- Remove Table button, 5-26
- removing sessions from brokers, 94-2
- Rename dialog, 5-26
- renaming
  - packages, 5-57
  - projects, 116-5, 116-8
- reopening projects, 116-5
- report query
  - about, 108-16
  - query objects, 12-27
  - read optimization, and, 12-25
  - using, 12-27, 108-8
- reports
  - project status, 116-8
  - query results, 108-8
  - see also* status reports
- result set pagination, 111-20
- resuming unit of work
  - after commit, 115-13
  - after commit failure, 115-14
- returning policy
  - configuring, 119-75

- SQLCall, 108-18
- Returning tab, 119-76
- RMI
  - coordinated cache, 105-1
  - naming service, 103-13
  - remote session support, 87-32
- rollback
  - and Java Transaction API, 113-11
  - overview, 113-11
  - with optimistic locking, 16-16
- root class
  - about, 16-9
  - inheritance, 16-9
- root EIS descriptor, 75-2
- root element, descriptor, 52-5
- runtime
  - acquiring sessions, 87-5
  - components, 3-3
  - services, configuring sessions, 89-17

## S

---

- Save All button, 116-7
- Save button, 116-7
- saving projects, 117-2
- schema context
  - descriptors without, 52-3
  - EIS descriptors, 76-3
  - XML descriptors, 52-2
- Schema Document Info tab, 5-40, 5-43
- schema manager
  - about, 6-1
  - automatic table creation, 6-6
  - automatic table creation, CMP, 6-7
  - automatic table creation, JPA, 6-6
  - CMP, 6-7
  - creating a table creator, 6-4
  - DB2, 6-3
  - default table generator, 6-4
  - Java table creator, 6-5
  - JPA, 6-6
  - MS Access, 6-3
  - MySQL, 6-3
  - Oracle, 6-3
  - Oracle TopLink Workbench table creator, 6-4
  - sequencing, 6-3, 6-6
  - Sybase, 6-3
  - table creator, 6-2, 6-4, 6-5, 6-6
  - table definition, 6-2
  - type conversion, 6-3
  - usage, 6-1
- Schema Structure tab, 5-37
- schemas
  - about, 2-18
  - context for EIS descriptors, 76-3
  - context for XML descriptor, 52-2
  - data storage, 2-18
  - database, 5-23
  - default root for EIS descriptors, 76-4
  - details, 5-37

- document information, 5-40, 5-43
- errors, A-32
- importing, 5-38
- optimizing, 12-11
- properties, 5-39
- reimporting, 5-39
- schema manager, 6-1
- structure, 5-37
- XML schemas, 5-37
- SCM. *see* source control management
- scripts
  - SQL, generating, 5-32
  - see also* SQL
- scrollable cursor
  - traversing, 111-16
  - using for ReadAllQuery, 111-16
- Securable interface, 97-4
- security
  - cache coordination, permissions, 8-20
  - data source access, permissions, 8-20
  - disabling doPrivileged(), 8-21
  - doPrivileged(), 8-4
  - EJB, permissions, 8-21
  - enabling doPrivileged(), 8-21
  - Java EE application, permissions, 8-21
  - java.util.logging, permissions, 8-21
  - loading project.xml, permissions, 8-20
  - loading sessions.xml, permissions, 8-20
  - logging, permissions, 8-21
  - password encryption, 116-9
  - permissions by feature, 8-19
  - permissions when doPrivileged()
    - disabled, 8-21
  - port permissions, 8-20
  - SecurityManager integration, 8-4
  - system properties, permissions, 8-20
  - understanding permissions, 8-19
  - with BEA WebLogic, 8-6
- Select Classes dialog box, 5-56
- SelectedFieldsLockingPolicy, 16-17
- sensors, DMS profiler, 12-4, 87-12
- SEQ\_COUNT column in sequence table, 18-9
- sequence connection pools, 96-8
- sequence numbers, write optimization
  - features, 12-32
- SEQUENCE objects in Oracle native
  - sequencing, 18-8
- sequencing
  - and relational projects, 18-3
  - configuring, 18-4
  - connection pools, 96-8
  - default, 98-6, 98-7
  - DefaultSequence class, 98-7
  - descriptors, 16-7, 23-4
  - entity beans, 18-10
  - isolated client sessions, 87-25
  - Java configuration, 18-4
  - Microsoft SQL Server, 6-6
  - native, 18-7, 18-8, 18-9
  - non-Oracle database native, 18-8
  - Oracle Database native, 18-7, 18-9
  - Oracle TopLink Workbench configuration, 18-4
  - overriding default, 98-7
  - platform default, 98-6
  - preallocation, 12-34, 18-9
  - QuerySequence class, 98-8
  - schema manager, 6-6
  - SEQ\_COUNT, 18-9
  - sequence type, configuring, 20-3, 98-5
  - sessions, and, 87-13
  - stored procedures, 18-6, 98-8
  - Sybase Database, 6-6
  - table, 18-5
  - table, default column and table names, 18-5
  - unary table, 18-6
  - with stored procedures, 98-8
- serialization
  - descriptor exceptions, 115-12
  - indirection, 17-11
  - indirection and JPA, 17-12
  - merging into session cache with unit of work, 115-12
- serialized object converter
  - about, 13-2, 17-12
  - configuring, 121-21
- serialized object mappings, 121-22
- SerializedObjectMapping. *see* SerializedObjectConverter
- server layer, 87-32
- server platform
  - BEA WebLogic, integrating, 8-5
  - external transaction controller, 89-17
  - IBM WebSphere, integrating, 8-16
  - JBoss, integrating, 8-18
  - OC4J, integrating, 8-7
  - runtime services, 89-17
  - session configuration, 89-17
  - session event listener, 89-16, 89-19, 119-90
  - SunAS, integrating, 8-17
- Server Platform tab, 89-18
- server session
  - logging out, 90-10
- server sessions
  - about, 87-2, 87-14, 87-16
  - cache, 87-34
  - connection options, 87-19
- service channel, coordinated cache, 103-4
- session beans
  - about, 2-33
  - deploying, 9-7, 10-3, 11-2
  - model, 2-30, 2-33
  - remote session support for, 87-32
- session broker session
  - logging out, 90-10
- session brokers
  - about, 87-2, 87-26
  - adding sessions to, 94-2
  - alternatives, 87-29
  - architecture, 87-27
  - configuration, 94-1

- limitations of, 87-28
- renaming, 94-2
- two-phase commits, 87-28
- two-stage commits, 87-28
- session configuration file
  - about, 88-1
  - loading alternative, 90-4
  - preferences, 5-19
  - see also* `sessions.xml` file
- session customizer, 87-5
- Session Event Manager, 87-6
- session events
  - example, 87-6
  - isolated sessions, and, 87-6
  - manager, 87-6
  - proxy authentication, and, 87-6
- Session Loader Exceptions, A-6
- Session Manager
  - about, 90-1, 90-2
  - acquiring, 90-2
  - defaults, 90-3
  - destroying sessions, 90-10
  - Java EE defaults, 90-3
  - sessions, acquiring, 87-5
  - storing sessions, 90-10
- sessions
  - about, 2-14, 86-1, 87-1, 87-2, 88-1
  - acquiring at runtime, 87-5, 90-1
  - adding to session brokers, 94-2
  - additional mapping projects, 89-11
  - API, 87-34
  - application server logging, 87-10
  - architecture, 87-3
  - cache, 87-34, 102-2
  - client, 87-14
  - configuring, 89-1, 89-2, 89-16, 89-19, 119-90
  - connection policy, 89-22
  - creating, 2-25, 88-1, 88-4, 88-7, 88-9, 100-1
  - customization, 87-5
  - database sessions, 87-29, 88-8
  - destroying in Session Manager, 90-10
  - errors, A-33
  - event listeners, 87-8, 89-16, 89-19, 119-90
  - events, 87-6
  - external transaction controller, 89-17, 113-2
  - finalizers, 90-10
  - historical client sessions, 87-25
  - in `sessions.xml` file, 87-5
  - isolated client sessions, 87-20, 92-1
  - loading with alternative class loader, 90-4
  - logging, 87-8, 87-10, 89-5, 89-6
  - logging into, 90-9
  - logging out of a client session, 90-10
  - logging out of a database session, 90-10
  - logging out of a server session, 90-10
  - logging out of a session broker session, 90-10
  - logins, 89-5
  - management of, 90-1
  - metadata, about, 2-25
  - multiple sessions, 87-29, 90-2
  - named queries, 89-24
  - object cache, 87-3
  - optimizing, 12-16
  - preferences, 5-19
  - queries, 108-7, 108-9, 109-1
  - refreshing, 90-5
  - registering descriptors, 87-13, 89-3, 89-4, 89-12
  - releasing a client session, 90-10
  - remote sessions, 87-30, 88-11
  - sequencing, about, 87-13
  - server, 87-14
  - server platform, 89-17, 89-18
  - SQL and messages, 87-10
  - storing in Session Manager, 90-10
  - three-tier architecture, 87-15
  - transformation mappings, 27-17
  - types, 87-1
  - unit of work, 87-2, 87-20
- `sessions.xml` file
  - about, 9-4
  - acquiring, 87-5
  - CMP applications, 9-5, 87-5
  - creating, 88-1
  - default location, 87-5
  - EJB 3.0, 9-5
  - loading alternative configuration file, 90-4
  - non-CMP applications, 9-4
  - schema, 9-4
  - sessions, 87-5
  - XSD file, 9-4
- `setAdditionalJoinExpression()`, 111-5
- `setenv.cmd` file, 5-2
- `setenv.sh` file, 5-2
- `setFirstResult`, 111-20
- `setMaxBatchWritingSize()`, 12-18
- `setMaxRows`, 111-20
- `setMultipleTableJoinExpression()`, 111-5
- `setShouldPerformDeletesFirst()`, 115-16
- Settings tab, 119-30, 119-31
- `setValue()` method, 17-8, Glossary-8
- shared library for BEA WebLogic, 8-5
- simple type translators
  - about, 13-2, 17-15
  - configuring, 121-28
  - in Java, 121-29
  - read conversions, 17-16
  - write conversions, 17-16
- soft cache weak identity map
  - about, 102-4
  - when to use, 102-5
- soft identity map, 102-4
- sorting, in memory, 121-31
- source code, public, 13-2
- source control management
  - projects, 7-3
  - with Oracle TopLink Workbench, 7-3
  - see also* team development
- source table, reference, 5-30
- splash screen, 5-13
- SQL



- comparing with expressions, 110-1
- custom queries for basic persistence, per
  - descriptor, 23-7
- EJBQLCall, 109-32
- generating from database tables, 5-32
- JPQLCall, 109-32
- parameter binding, 12-19
- parameterized, 12-32, 109-17
- prepared statement caching, 12-19
- queries, 108-5
- scripts with binding arguments, 23-7
- SQLCall, about, 109-18
- SQLCall, JDBC arguments, 109-19
- SQLCall, no arguments, 109-19
- unit of work, 115-15
- SQL Creation Script dialog box, 5-32
- SQL DISTINCT, 12-30
- SQL Exception, A-4
- SQLAnywhere platform, 96-3
- SQLCall
  - about, 108-18
  - Returning Policy, 108-18
- SQLServer platform, 96-3
- stages of development with Oracle TopLink, 2-2
- stale data
  - cache, 102-6, 102-7
  - coordination, cache, 102-7
  - invalidating the cache, 102-7
  - locking policy, and, 102-7
  - per-class cache configuration, 102-7
  - per-query cache refresh, 102-7
- startWith, 111-7
- stateful
  - beans, 2-33
  - comparing with stateless, 2-33
- stateless
  - comparing with stateful, 2-33
- static attributes, 5-49, 5-53
- static fetch groups, querying with, 111-3
- status report, generating, 116-8
- stored functions
  - about, 108-24
- stored procedure
  - StoredProcedureCall, 109-31
  - StoredProcedureCall, about, 109-21
  - StoredProcedureCall, ErrorCodeListener, 109-29
  - StoredProcedureCall, JDBC arguments, 109-22, 109-24
  - StoredProcedureCall, no arguments, 109-22
  - StoredProcedureCall, PL/SQL arguments, 109-24
- stored procedures
  - sequencing, and, 18-6, 98-8
- streams
  - as query results, 108-9
  - cursor, 87-34, 111-18
- structure object-relational data type mappings
  - configuring, 42-1
  - understanding, 40-2
- Structure window
  - TopLink, 4-3

- StructureMapping class, 40-2
- subqueries
  - multiple expressions, 110-10
  - subselects in expressions, 110-10
- subselects, in expressions, 110-10
- summary queries, 108-3
- SunAS
  - setting classpath, 8-18
- superclass, 5-46
- Sybase
  - database schema manager type conversion, 6-3
  - native sequencing, 18-8
  - platform, 96-3
- synchronous change propagation, 103-2
- system properties
  - oracle.j2ee.security.usedoprivileged, 8-21
  - oracle.j2ee.toplink.security.usedoprivileged, 8-21
  - toplink.cts.collection.checkParameters, 8-22
  - toplink.xml.platform, 8-4

## T

---

- table creator
  - about, 6-2
  - creating, 6-4
  - using, 6-6
- Table Creator dialog, 6-4
- table generation. *see* automatic table generation
- table sequence
  - about, 18-5
  - default column and table names, 18-5
- tables
  - adding database, 5-23
  - associating with relational descriptors, 23-3
  - database, 5-22
  - defining schema, 6-2
  - errors, A-27
  - generating, automatic, 17-5
  - import filter, 5-24
  - mapping to descriptors, 23-3, 119-5
  - merging files, 7-5
  - multiple, 23-14
  - primary key, 5-28
  - references, 28-10
  - relation tables for mappings, 34-2
  - TableDefinition class, 6-2
  - see also* database tables
- target foreign keys
  - about, 27-6, 28-10
  - configuring, 28-10
- target platforms
  - about, 2-5
  - choosing, 2-3
- target tables
  - in direct collection mappings, 36-2
  - reference, 5-30
- team development, 7-3
- technical support, 2-3
- three-tier architecture

- about, 1-5, 2-28
- authentication, 96-5
- migrating to scalable architecture, 87-30
- overview, 2-28
- sessions, 87-15
- timestamp support
  - about, 28-4
  - direct to field mappings, 28-4
  - Oracle Database, 28-4
  - TIMESTAMP timezone, 28-4
- TimestampLockingPolicy, 16-14
- timezone, with TIMESTAMP, 28-4
- tljxb.cmd file, 48-4
- toolbars, 5-4, 5-6
- topic name, 104-2
- TopLink, 4-1
  - Application Navigator, 4-2
  - Structure window, 4-3
- TopLink. *see* Oracle TopLink
- TopLink Editor, 4-2
- TopLink editor, 4-2
- TopLink Editor, parts of, 4-1
- TopLink expressions. *see* expressions
- TopLink profiler
  - about, 12-2
  - selecting, 89-13, 89-16, 89-20, 89-23
- TopLink Workbench. *see* Oracle TopLink Workbench
- toplink-*ejb-jar.xml* file, 9-7, 14-2
- toplink-*ejb-jar.xml** file, 9-7
- toplink-*src.zip*, 13-2
- Transaction Exception, A-8
- transactional data sources, 96-1
- transactions
  - client-controlled, 115-24
  - CMP, 113-3, 115-24, 115-25
  - container-controlled, 115-24
  - demarcation, 113-2
  - external transaction controller, 89-17, 113-2
  - external, integrating, 113-2, 115-20
  - isolated client sessions, 87-25
  - isolation, 113-2, 113-4
  - JTA, 113-3
  - JTS, 113-3
  - local, 115-25
  - local, CMP, 115-25
  - OTS, 113-3
  - overview, 2-15, 86-3, 113-1
  - see also* unit of work
- transformation EIS mappings
  - configuring, 85-1
  - understanding, 77-18
- transformation mappings
  - about, 17-17
  - and copy policy, 2-22, 113-9, 121-39
  - attribute transformation, 121-34, 121-36
  - mutability, 121-39
- transformation relational mappings
  - configuring, 39-1
  - understanding, 27-16
- transformation XML mappings
  - configuring, 61-1
  - understanding, 53-35
- transient attributes, 5-49, 5-53
- transparent indirection
  - about, 17-9, Glossary-8
  - persistent class requirements, 2-14
- transport layer, 87-32
- troubleshooting
  - MaD support, A-1
  - migration from OC4J persistence, 8-15
  - Oracle TopLink Workbench, A-10
  - unit of work, 115-31, 115-34
- two-phase commits, 87-28
- two-stage commits, 87-28
- two-tier architecture
  - about, 1-7, 2-30
  - authentication, 96-5
  - understanding, 2-31
- type conversion
  - automatic, 28-3
  - NCHAR, 17-14
  - NCLOB, 17-14
  - NVARCHAR2, 17-14
  - oracle.sql.TimeStamp, 28-4
  - schema manager, 6-3
  - String to TIMESTAMP, 17-14
  - TIMESTAMP to String, 17-14
- type conversion converter
  - about, 13-2, 17-13
  - configuring, 121-23
  - provided by direct-to-field mappings, 28-3
- TypeConversionMapping
  - see* TypeConversionConverter
- types of mappings, 17-1
- typesafe enumeration, in EIS mappings, 77-4

## U

---

- unary table sequence
  - about, 18-6
- undeployment, 11-3
- unexpected relationship multiplicity, 8-16
- unidirectional relationships, 27-2
- unit of work
  - about, 113-1, 113-2, 113-4
  - acquiring, 114-1
  - API, 113-12
  - architecture, 113-1
  - auditing, 115-20
  - benefits of, 113-4
  - cache, 102-2
  - change policy, 113-7
  - clones, 113-9
  - CMP integration, 113-3
  - commit and Java Transaction API, 113-10, 113-11
  - commit, writing changes before, 115-6, 115-11, 115-23
  - conform results of in-memory query, 108-32
  - creating objects, 114-2
  - deleting objects, 114-10

- example, 113-6
- external transaction controller, 89-17, 113-2
- external transactions, 113-2, 115-20
- integrating with CMP, 115-24
- isolation, 113-4
- JTA integration, 113-3
- JTS integration, 113-3
- life cycle, 113-5
- modifying objects, 114-2
- mutable mappings, 121-39
- nested, 113-10, 115-14, 115-15
- newInstance method, 115-2
- optimization, 12-35, 113-11
- OTS integration, 113-3
- parallel, 113-10, 115-14
- pre-commit validation, 115-34
- primary keys, 113-11
- proxy indirection, 17-10
- queries, 113-12
- read optimization, 12-25
- read-only classes, 115-5, 115-6
- registerNewObject method, 114-8, 115-5
- registerObject method, 114-5, 114-6, 114-8, 115-5
- remote sessions, 87-34
- resuming, 115-13
- reverting, 115-14
- rollback, 113-11
- sessions, 87-2, 87-20
- transaction demarcation, 113-2
- transactions, 113-4
  - unit of work
    - demarcation, 113-2
- troubleshooting, 115-31
- validating objects, 115-34
- with custom SQL, 115-15
- write optimization, 12-32
- writing changes before commit, about, 115-6
- writing changes before commit, and external transaction exceptions, 115-23
- writing changes before commit, and external transaction timeouts, 115-23
- writing changes before commit, as alternative to conforming, 115-11
- unmapping, 120-5
- unregistering objects, 115-1
- unsaved items, displaying in TopLink Workbench Navigator window, 5-9
- update
  - operation, 109-3, 109-4
  - projects from prior versions, 116-5
- UpdateAll query, 108-15
- updateObject(), 23-9
- Use XML Schema "type" attribute, configuring, 121-28
- useBatchWriting(), 98-12
- user-defined functions, in expressions, 110-16

## V

---

- validating
  - descriptors, 118-1
  - JAXB, 47-9
  - projects, 116-8
  - unit of work, 115-34
- Validation Exceptions, A-5
- Value Converter tab, 38-6
- value holders
  - about, 17-8, Glossary-8
  - ValueHolder class, 17-8, Glossary-8
- ValueHolderInterface class, 2-14, 17-8, 27-10, Glossary-8
- variable one-to-one relational mappings
  - class indicator, 32-2
  - configuring, 32-1
  - primary key, unique, 32-4
  - understanding, 27-7
- VariableOneToOneMapping class, 27-7
- Varray in Oracle database. *see* array mappings
- version control, 7-6
- Version Control Assistance dialog box, 7-6
- VersionLockingPolicy, 16-13
- volatile attributes, 5-49, 5-53
- VPD. *see* Oracle Virtual Private Database

## W

---

- warning icon, 5-10, 5-11
- weak identity map, 102-4
- weaving
  - about, 2-13, 2-25
  - application servers, about, 2-28
  - change policy and lazy loading,
    - configuring, 17-9, 17-11, 119-81, 121-5
  - change tracking, 2-27, 113-8
  - change tracking and lazy loading, about, 113-8
  - change tracking and lazy loading,
    - configuring, 17-9, 17-11, 119-81, 121-5
  - disabling, 2-26
  - dynamic, javaagent, 2-26
  - fetch groups, configuring, 119-88
  - indirection, 2-27
  - indirection, configuring, 17-9, 17-11, 119-81, 121-5
  - Java EE, about, 2-28
  - JPA enties, change tracking, 113-8
  - lazy loading, 2-27
  - lazy loading, configuring, 17-9, 17-11, 119-81, 121-5
  - mutability, 2-22
  - OC4J, about, 2-28
  - optimization, 12-36
  - packaging JPA application, 10-5
  - packaging POJO application, 2-27, 10-5
  - POJO classes, about, 2-27
  - POJO classes, change tracking, 2-27, 113-8
  - POJO classes, indirection, 2-27
  - POJO classes, lazy loading, 2-27
  - static, about, 2-26
- web browser, specifying, 5-14

- web services architecture, 1-6, 2-42
- weblogic-ejb-jar.xml file
  - modifying for Oracle TopLink, 9-9
  - unsupported tags, 9-9
- weblogic-ejb-jar.xml file, 9-6
- welcome screen, 5-14
- wildcard, 54-5
- wrapper policy
  - about, 119-86
  - implementing in Java, 119-87
- write
  - conversions, simple type translator, 17-16
  - write all operation, 109-3
- write query
  - disabling identity map cache, 109-13
  - non-cascading, 109-13
  - objects, 109-12
  - overview, 109-12
- write-locking, 16-7
- writing
  - batch, 12-32, 98-12
  - ejb-jar.xml file, 19-5
  - optimization, 12-32
  - sessions write access, 87-17

## X

---

- xdb.jar file, 5-3
- XML
  - descriptor, schema context, 52-2
  - generating deployment, 9-3
  - mappings, concepts, 53-3
  - preserving data, 52-6
  - projects, 116-3
  - query language, 108-6
  - reading whole documents, 30-2
  - records, 77-3
  - schemas, 5-37
- XML Conversion Exception, A-8
- XML descriptors
  - configuring, 52-1
  - creating, 51-1
  - schema context, 52-2
  - understanding, 50-1
- XML mappings
  - about, 53-1
  - any attribute, 70-1
  - any collection, 53-33, 60-1
  - any object, 53-31, 59-1
  - any type support, 53-5
  - binary data, 64-1
  - binary data collection, 65-1, 67-1
  - CDATA, 121-44
  - choice collection, 69-1
  - choice object, 68-1
  - collection reference, 63-1
  - composite collection, 53-29, 58-1
  - composite direct collection, 53-18, 56-1
  - composite object, 53-24, 57-1
  - concepts, 53-3

- configuring, 54-1
- default conversion pairs, customizing, 53-15
- direct, 53-7, 55-1
- extensions, 53-6
- fragment, 66-1
- jaxb:class support, 53-6
- list support, 53-5
- object reference, 62-1
- reference descriptor, configuring, 54-3
- transformation, 53-35, 61-1
- types of, 53-1
- union support, 53-5
- xsd:list, 53-5
- xsd:union, 53-5
- XML parser platform
  - about, 8-3
  - configuring, 8-3
  - creating, 8-4
  - Crimson, 8-4
  - default, 8-3
  - limitations, 8-4
  - parser conflicts, 8-3
  - toplink.xml.platform system property, 8-4
  - used by, application server, 8-3
  - used by, Oracle TopLink, 8-3
- XML Platform Exception, A-9
- XML projects
  - configuring, 49-1
  - introduction, 47-1
  - JAXB support, 47-2
  - sequencing, 15-5
- XML queries, 108-6
- XML schema
  - jaxb:class, and EIS mappings, 77-3
  - jaxb:class, and XML mappings, 53-6
  - jaxb:class, understanding, 17-23
  - type, 121-28
  - xs:any, understanding, 53-5
  - xs:anyType, understanding, 53-5
  - xsd:list, understanding, 17-20
  - xsd:union, understanding, 17-20
  - see also* schemas
- XML Type functions, 110-5
- XMLMarshalException, A-9
- XMLPlatformException, A-9
- XPath
  - by name, 17-19
  - by position, 17-18
  - mapping Java attributes, 121-12
  - support in OX mappings, 17-17
  - support in XML mappings, 53-5, 77-3
- XSD file
  - sessions.xml file, 9-4

## Z

---

- zero-argument constructors
  - editing, 5-16, 5-17