Oracle® Application Server Portal

Developer's Guide

10*g* Release 2 (10.1.2)

Part No. B14134-01

November 2004

Oracle Application Server Portal Developer's Guide, 10*g* Release 2 (10.1.2)

Part No. B14134-01

Contributing Authors: Joan Carter, Tugdual Grall, Helen Grembowicz, Peter Moskovits, Frank Rovitto, Ingrid Snedecor, Sue Vickers, Vanessa Wang

Contributors: Gareth Bryan, Abhinav Gattani, Paul Lin, Pankaj Mittal, Lei Oh, Alistair Wilson

Contents

Send Us Your Comments xiii

Preface xv

Intended Audience xv
Documentation Accessibility xv
Structure xvi
Related Documents xvii
Conventions xvii
Browser Recommendations xviii

Part I Portlet Overview

1 Understanding Portlets

1.1 Introduction to Portal Development 1-1
1.2 Understanding Portlets 1-1
1.3 Portlet Anatomy 1-3
1.4 Portlet Resources 1-4
1.4.1 Out-of-the-Box Portlets 1-5
1.4.2 Other Sources of Pre-Built Portlets 1-6
1.4.3 Web Clipping 1-6
1.4.4 OmniPortlet 1-9
1.4.5 Portlet Builder 1-10
1.4.6 Programmatic Portlets 1-11
1.4.7 Deciding Which Tool to Use 1-12
1.5 Summary 1-12

Part II Portlet Technologies

2 Portlet Technologies Matrix

2.1 The Portlet Technologies Matrix 2-1
2.2 General Suitability 2-4
2.2.1 Web Clipping 2-4
2.2.1.1 Examples of portlets you can build using Web Clipping 2-4
2.2.2 OmniPortlet 2-4
2.2.2.1 Examples of portlets you can create with OmniPortlet 2-4

2.2.3 Java Portlets 2-4
2.2.3.1 Examples of portlets you can build using Java 2-5
2.2.4 Portlet Builder 2-5
2.2.4.1 Examples of portlets you can build using the Portlet Builder 2-5
2.2.5 PL/SQL Portlets 2-5
2.2.5.1 Examples of portlets you can build using PL/SQL 2-5
2.3 Expertise Required 2-5
2.3.1 Web Clipping 2-5
2.3.2 OmniPortlet 2-6
2.3.3 Java Portlets 2-6
2.3.4 Portlet Builder 2-6
2.3.5 PL/SQL Portlets 2-6
2.4 Deployment Type 2-6
2.4.1 Web Providers 2-7
2.4.2 WSRP Providers 2-8
2.4.3 Database Providers 2-8
2.4.4 Provider Architecture 2-10
2.5 Caching Style 2-11
2.5.1 Web Clipping, OmniPortlet, and Portlet Builder 2-12
2.5.2 Java Portlets 2-12
2.5.3 PL/SQL Portlets 2-12
2.6 Development Tool 2-12
2.6.1 Web Clipping, OmniPortlet, and Portlet Builder 2-12
2.6.2 Java Portlets 2-12
2.6.3 PL/SQL Portlets 2-13
2.7 Portlet Creation Style 2-13
2.7.1 OmniPortlet and Web Clipping 2-14
2.7.2 Java Portlets 2-14
2.7.3 Portlet Builder 2-14
2.7.4 PL/SQL Portlets 2-15
2.8 User Interface Flexibility 2-15
2.8.1 Web Clipping 2-15
2.8.2 OmniPortlet and Portlet Builder 2-15
2.8.3 Java Portlets and PL/SQL Portlets 2-15
2.9 Ability to Capture Content from Web Sites 2-15
2.9.1 Web Clipping 2-15
2.9.2 OmniPortlet 2-16
2.9.3 Java Portlets 2-16
2.9.4 PL/SQL Portlets 2-16
2.10 Ability to Render Content Inline 2-16
2.10.1 Web Clipping 2-16
2.10.2 OmniPortlet 2-17
2.10.3 Java Portlets 2-17
2.10.4 Portlet Builder 2-17
2.10.5 PL/SQL Portlets 2-17
2.11 Charting Capability 2-17
2.11.1 Web Clipping 2-17

2.11.2 OmniPortlet 2-17
2.11.3 Java Portlets 2-17
2.11.4 Portlet Builder 2-18
2.11.5 PL/SQL Portlets 2-18
2.12 Public Portlet Parameters Support 2-18
2.13 Private Portlet Parameter Support 2-19
2.13.1 OmniPortlet, Web Clipping, and Portlet Builder 2-19
2.13.2 Java Portlets and PL/SQL Portlets 2-19
2.14 Event Support 2-19
2.14.1 Web Clipping, OmniPortlet, and Java Portlets 2-19
2.14.2 Portlet Builder and PL/SQL Portlets 2-19
2.15 Ability to Hide and Show Portlets Based on User Privileges 2-19
2.15.1 Web Clipping and OmniPortlet 2-19
2.15.2 Java Portlets 2-19
2.15.3 Portlet Builder 2-20
2.15.4 PL/SQL Portlets 2-20
2.16 Multi-lingual Support 2-20
2.16.1 Web Clipping, OmniPortlet, Java Portlets, and PL/SQL Portlets 2-20
2.16.2 Portlet Builder 2-20
2.17 Pagination Support 2-20
2.17.1 OmniPortlet 2-20
2.17.2 Java Portlets and PL/SQL Portlets 2-20
2.17.3 Portlet Builder 2-20
2.18 Single Sign-On and External Application Integration 2-20
2.18.1 Web Clipping 2-21
2.18.2 OmniPortlet 2-21
2.18.3 Java Portlets 2-21
2.18.4 PL/SQL Portlets 2-21

Part III Building Portlets

3 Building Portlets with OmniPortlet

3.1 What is OmniPortlet? 3-2
3.1.1 Type 3-3
3.1.2 Source 3-3
3.1.2.1 Spreadsheet 3-5
3.1.2.2 SQL 3-5
3.1.2.3 XML 3-6
3.1.2.4 Web Service 3-7
3.1.2.5 Web Page 3-8
3.1.3 Filter 3-9
3.1.4 View 3-10
3.1.5 Layout 3-11
3.1.5.1 Tabular Layout 3-12
3.1.5.2 Chart Layout 3-13
3.1.5.3 News Layout 3-15

3.1.5.4 Bullet Layout 3-17
3.1.5.5 Form Layout 3-18
3.1.6 Edit Defaults mode 3-19
3.1.7 Events 3-20
3.2 Parameters and Events 3-20
3.2.1 Portlet Parameters and Events 3-21
3.2.2 Page Parameters and Events 3-22
3.3 Using OmniPortlet 3-22
3.3.1 Adding an OmniPortlet to a Portal Page 3-22
3.3.2 Defining a Portlet Based on a SQL Data Source 3-23
3.3.3 Defining a Portlet Based on a Web Service 3-26
3.3.4 Defining a Portlet Based on an Existing Web Page 3-28
3.3.5 Modifying the Layout of an Existing OmniPortlet 3-35
3.3.6 Using Parameters and Events 3-37
3.3.6.1 Adding Parameters to an Existing OmniPortlet 3-38
3.3.6.2 Adding Events to an Existing OmniPortlet 3-40
3.3.6.3 Relating Portlet Parameters and Events on a Page 3-40
3.4 Summary 3-43

4 Building Content-Based Portlets with Web Clipping

4.1 What Is Web Clipping? 4-1
4.2 Adding Web Page Content to a Portal Page 4-2
4.2.1 Adding a Web Clipping Portlet to a Page 4-2
4.2.2 Selecting a Section of a Web Page to Display in the Web Clipping Portlet 4-4
4.2.3 Setting Web Clipping Portlet Properties 4-8
4.3 Integrating Authenticated Web Content Using Single Sign-On 4-9
4.4 Example: Adding a Web Clipping That Users Can Customize 4-15
4.4.1 Exercise: Adding a Web Clipping Portlet to a Personal Page 4-15
4.4.2 Exercise: Selecting a Clipping in OTN 4-16
4.4.3 Exercise: Customizing a Web Clipping Portlet 4-18
4.5 Current Limitations for Web Clipping 4-21

5 Building Java Portlets

5.1 Guidelines for Creating Java Portlets 5-2
5.1.1 Shared Screen Mode (View Mode for JPS) 5-2
5.1.1.1 HTML Guidelines for Rendering Portlets 5-2
5.1.1.2 Cascading Style Sheet Guidelines for Rendering Portlets 5-3
5.1.2 Edit Mode (JPS and OracleAS Portal) 5-4
5.1.2.1 Guidelines for Edit Mode Options 5-4
5.1.2.2 Guidelines for Buttons in Edit Mode 5-4
5.1.2.3 Guidelines for Rendering Customization Values 5-4
5.1.3 Edit Defaults Mode (JPS and OracleAS Portal) 5-5
5.1.3.1 Guidelines for Edit Defaults Mode Options 5-5
5.1.3.2 Guidelines for Buttons in Edit Defaults Mode 5-5
5.1.3.3 Guidelines for Rendering Customization Values 5-5
5.1.4 Preview Mode (JPS and OracleAS Portal) 5-6
5.1.4.1 Guidelines for Preview Mode 5-6

5.1.5 Full Screen Mode (OracleAS Portal) 5-6
5.1.5.1 Guidelines for Full Screen Mode 5-7
5.1.6 Help Mode (JPS and OracleAS Portal) 5-7
5.1.6.1 Guidelines for Help Mode 5-7
5.1.7 About Mode (JPS and OracleAS Portal) 5-7
5.1.7.1 Guidelines for About Mode 5-7
5.1.8 Link Mode (OracleAS Portal) 5-7
5.1.8.1 Guidelines for Link Mode 5-8
5.2 Introduction to Java Portlet Specification and WSRP 5-8
5.2.1 The Relationship Between WSRP and JPS 5-9
5.3 Building JPS-Compliant Portlets with Oracle JDeveloper 5-10
5.3.1 Installing the Oracle JDeveloper Portal Add-In 5-11
5.3.2 Building JPS-compliant Portlets 5-11
5.3.2.1 Creating a Portlet 5-11
5.3.2.2 Adding Portlet Logic 5-20
5.3.2.3 Deploying Your Portlet to an Application Server 5-20
5.3.2.3.1 Creating a Connection to Oracle Application Server Containers for J2EE 5-21
5.3.2.3.2 Deploying the WAR File 5-22
5.3.2.4 Registering and Viewing Your Portlet 5-22
5.3.2.4.1 Registering on a Local OracleAS Portal Instance 5-23
5.3.2.4.2 Registering on portal.standards.com 5-28
5.3.2.4.3 Adding Your Portlet 5-28
5.4 Building PDK-Java Portlets with Oracle JDeveloper 5-28
5.4.1 Installing the Oracle JDeveloper Portal Add-In 5-29
5.4.2 Building PDK-Java Portlets 5-29
5.4.2.1 Creating a Portlet and Provider 5-29
5.4.2.2 Adding Portlet Logic 5-36
5.4.2.3 Validating Your Portlet and Provider 5-36
5.4.2.4 Deploying to an Application Server 5-38
5.4.2.4.1 Creating a Connection to Oracle Application Server Containers for J2EE 5-38
5.4.2.4.2 Deploying the WAR File 5-39
5.4.2.5 Registering and Viewing Your Portlet 5-41
5.4.3 Adding Render Modes 5-45
5.4.3.1 Assumptions 5-46
5.4.3.2 Implementing Extra Show Modes 5-46
5.4.3.3 Updating the XML Provider Definition 5-46
5.4.3.4 Viewing the Portlet 5-46
5.4.4 Customizing Portlets 5-48
5.4.4.1 Assumptions 5-50
5.4.4.2 Implementing Customization for Edit and Edit Defaults Pages 5-50
5.4.4.2.1 Reviewing the Generated Code 5-50
5.4.4.2.2 Modifying the Generated Code 5-51
5.4.4.3 Implementing Customization for Show Pages 5-52
5.4.4.4 Preference Information Within the XML Provider Definition 5-52
5.4.4.5 Viewing the Portlet 5-53
5.4.5 Passing Parameters and Submitting Events 5-53
5.4.5.1 Assumptions 5-53

5.4.5.2 Adding Parameters to Your Portlets 5-54
5.4.5.3 Submitting Events 5-56
5.4.5.3.1 Creating an Events Portlet 5-56
5.4.6 Accessing Session Information 5-59
5.4.6.1 Assumptions 5-60
5.4.6.2 Implementing Session Storage 5-60
5.4.6.3 Viewing the Portlet 5-62
5.4.7 Implementing Portlet Security 5-62
5.4.7.1 Assumptions 5-62
5.4.7.2 Portlet Security Features 5-62
5.4.7.2.1 Authentication 5-62
5.4.7.2.2 Authorization 5-63
5.4.7.2.3 Communication Security 5-63
5.4.7.3 Single Sign-On 5-63
5.4.7.3.1 Partner Application 5-64
5.4.7.3.2 External Application 5-64
5.4.7.3.3 No Application Authentication 5-65
5.4.7.4 OracleAS Portal Access Control Lists (ACLs) 5-65
5.4.7.5 Portlet Security Managers 5-66
5.4.7.5.1 Viewing the Portlet 5-67
5.4.7.5.2 Implementing Your Own Security Manager 5-68
5.4.7.6 OracleAS Portal Server Security 5-68
5.4.7.7 Message Authentication 5-68
5.4.7.8 HTTPS Communication 5-69
5.4.7.8.1 Configuration of SSL 5-70
5.4.7.9 LDAP (Oracle Internet Directory) Security 5-70
5.4.7.9.1 Implementing Oracle Internet Directory Security 5-71
5.4.7.9.2 Viewing Your Portlets 5-73
5.4.8 Controlling the Export/Import of Portlet Customizations 5-74
5.4.8.1 Import/Export Programming Interface 5-75
5.4.8.1.1 Logging Interface 5-76
5.4.8.2 Exporting Customizations Example 5-77
5.4.8.3 Implementing Security for Export/Import 5-82
5.4.8.3.1 Securing Provider Communications 5-82
5.4.8.3.2 Disabling Export/Import of Customizations 5-83
5.4.8.3.3 Obfuscating Data for Transport (Automatic) 5-83
5.4.8.3.4 Encrypting Customization Data for Transport 5-83
5.4.8.3.5 Exporting by Reference 5-83
5.4.8.3.6 Encrypting Customization Data Example 5-84
5.4.8.3.7 Exporting by Reference Example 5-86
5.4.9 Enhancing Portlet Performance with Caching 5-88
5.4.9.1 Assumptions 5-89
5.4.9.2 Activating Caching 5-90
5.4.9.3 Adding Expiry-Based Caching 5-90
5.4.9.4 Adding Invalidation Based Caching 5-91
5.4.9.4.1 Configuring the Provider Servlet 5-91
5.4.9.4.2 Defining the OracleAS Web Cache Invalidation Port 5-91

5.4.9.4.3 Configuring the XML Provider Definition 5-93
5.4.9.4.4 Manually Invalidating the Cache 5-94
5.4.9.5 Adding Validation-Based Caching 5-95
5.4.10 Writing Multi-Lingual Portlets 5-95
5.4.10.1 Assumptions 5-96
5.4.10.2 Internationalizing Your Portlet 5-96
5.4.10.2.1 Providing Translations for Portlet Content 5-96
5.4.10.2.2 Providing Translation for Portlet Attributes 5-98
5.4.10.3 Viewing the Portlet 5-101
5.5 Building Struts Portlets with Oracle JDeveloper 5-101
5.5.1 OracleAS Portal and the Apache Struts Framework 5-101
5.5.1.1 Model View Controller Overview 5-101
5.5.1.2 Apache Struts Overview 5-103
5.5.1.3 OracleAS Portal Integration with Struts 5-103
5.5.1.3.1 Oracle Struts Portlet 5-104
5.5.1.4 Summary 5-105
5.5.2 Creating a Struts Portlet 5-105
5.5.2.1 Creating a Struts Portlet 5-106
5.5.2.1.1 Create a new flow and view to host the Portlet actions 5-107
5.5.2.1.2 Creating the new JSPs 5-107
5.5.2.1.3 Creating a Portlet 5-108
5.5.2.1.4 Extending the portlet to add Portal Business Logic 5-109
5.5.2.2 Registering the Provider 5-109
5.5.2.3 Summary 5-109

6 Building PL/SQL Portlets

6.1 Guidelines for Creating PL/SQL Portlets 6-1
6.1.1 Portlet Show Modes 6-2
6.1.2 Recommended Portlet Procedures and Functions 6-2
6.2 Building PL/SQL Portlets with the PL/SQL Generator 6-3
6.2.1 Creating the Input XML File 6-4
6.2.2 Running the PL/SQL Generator 6-6
6.2.3 Publishing the Generated PL/SQL Portlet 6-7
6.2.3.1 Installing the Packages in the Database 6-8
6.2.3.2 Registering the Database Provider 6-8
6.2.3.3 Adding Your Portlet to a Page 6-8
6.3 Building PL/SQL Portlets Manually 6-8
6.3.1 Implementing the Portlet Package 6-9
6.3.2 Implementing the Provider Package 6-10
6.3.3 Adding Your Portlet to a Page 6-14
6.4 Implementing Information Storage 6-14
6.4.1 Implementing a Preference Store 6-15
6.4.1.1 Using a Preference Store 6-15
6.4.1.2 Creating and Accessing a Preference Store 6-15
6.4.2 Implementing a Session Store 6-19
6.4.2.1 Creating and Accessing a Session Store 6-19
6.5 Using Parameters 6-21

6.5.1 Passing Private Parameters 6-22
6.5.2 Passing Page Parameters and Mapping Public Portlet Parameters 6-22
6.5.3 Retrieving Parameter Values 6-23
6.6 Accessing Context Information 6-24
6.6.1 Using Context Information 6-25
6.6.2 Using wwctx_api to Obtain Context Information 6-25
6.7 Implementing Portlet Security 6-27
6.7.1 Using Security 6-27
6.7.1.1 Guidelines for Using the Security APIs 6-27
6.7.2 Coding Security 6-28
6.8 Improving Portlet Performance with Caching 6-30
6.8.1 Using Caching 6-31
6.8.1.1 Validation-Based Caching 6-31
6.8.1.2 Expiry-Based Caching 6-32
6.8.1.3 Invalidation-Based Caching 6-32
6.8.2 Configuring and Monitoring the Cache 6-32
6.8.3 Implementing Validation-Based Caching 6-32
6.8.4 Implementing Expiry-Based Caching 6-34
6.8.5 Implementing Invalidation-Based Caching 6-35
6.9 Implementing Error Handling 6-36
6.9.1 Using Error Handling 6-37
6.9.1.1 Guidelines for Error Handling 6-38
6.9.2 Adding Error Handling 6-38
6.10 Implementing Event Logging 6-40
6.10.1 Using Event Logging 6-41
6.10.1.1 Guidelines for Event Logging 6-41
6.10.2 Adding Event Logging 6-41
6.11 Writing Multi-Lingual Portlets 6-43
6.11.1 Using Multi-Lingual Support 6-43
6.11.2 Adding Multi-Lingual Support 6-44
6.11.2.1 Loading Language Strings 6-44
6.11.2.2 Retrieving Language Strings 6-45

Part IV Appendixes

A Building Portlets with the Portlet Builder

A.1 Using a Wizard to Build a Portlet A-1
A.1.1 Creating a Schema in OracleAS Portal A-2
A.1.1.1 Creating a Schema A-2
A.1.1.2 Granting and Revoking Privileges on Database Objects A-4
A.1.1.3 Enrolling the Schema in One or More Roles A-7
A.1.2 Creating a Provider for Locally Built Portlets A-8
A.1.3 Exposing a Provider A-9
A.1.4 Creating Portlets Using OracleAS Portal Wizards A-11
A.1.4.1 Building Portlets Declaratively A-13
A.1.4.2 Building Forms Declaratively A-15
A.1.4.3 Building Reports Declaratively A-31

A.1.4.4 Building Forms and Reports against *inter*Media Rich Content A-54
A.1.4.5 Building Charts Declaratively A-56
A.1.4.6 Building Lists of Values Declaratively A-77
A.2 Editing a Portlet Builder Component A-81
A.3 Managing Portlets A-82
A.3.1 Navigating to the Component Management Page A-83
A.3.2 Renaming a Portlet A-83
A.3.3 Deleting a Portlet A-83
A.3.4 Copying a Portlet A-84
A.3.5 Generating the PL/SQL Package for a Portlet A-84
A.3.6 Viewing Source Code A-84
A.3.6.1 Viewing the Package Spec and Body for a Portlet A-85
A.3.6.2 Viewing the Call Interface for a Portlet A-85
A.3.7 Managing Locks on Portlets A-85
A.4 Managing Versions A-86
A.5 Managing Portlet Security A-87
A.5.1 Granting Portlet Access Privileges A-88
A.5.1.1 Inheriting Portlet Access Privileges from a Provider A-88
A.5.1.2 Granting Access Privileges to Individual Users A-89
A.6 Performing Test Runs on a Portlet A-90
A.6.1 Running a Component as a Full Page A-91
A.6.2 Running a Component as a Portlet A-91
A.6.3 Running a Component through the Customization Form A-91
A.6.4 Running the Component as a Portlet through the Portlet Customization Form A-91
A.6.5 Running in Batch Mode A-92
A.6.5.1 Setting init.ora Parameters for Batch Jobs A-93
A.6.5.2 Adding a Batch Button to an Existing Component A-93
A.7 Referencing the OracleAS Portal Schema A-94
A.8 Coding Additional Functionality A-95
A.8.1 Using Bind Variables A-96
A.8.2 Writing Event Handlers for Items on Forms A-96
A.8.2.1 Writing a JavaScript Event Handler for an Item on a Form A-96
A.8.2.2 Writing a PL/SQL Event Handler for a Button on a Form A-97
A.8.3 Using PL/SQL to Get and Set Values in a Form A-98
A.8.4 Using PL/SQL to Get or Set Cookies in a Form or Report A-99
A.8.5 Defining Values through Page Parameters A-99
A.9 Using Shared Components to Create a Look and Feel A-104
A.9.1 Granting Access to Shared Components A-105
A.9.2 Using JavaScript to Create Field- and Form-Level Validation A-106
A.9.2.1 Guidelines for Writing Field- or Form-Level Validation JavaScript A-106
A.9.2.2 Creating JavaScript under the Shared Components Provider A-107
A.9.2.3 Adding JavaScript to a Form A-107
A.9.3 Creating Color Definitions A-107
A.9.4 Creating Image Definitions A-108
A.9.5 Creating Font Definitions A-109
A.9.6 Using User Interface Templates A-110
A.9.6.1 Building a Structured User Interface Template A-110

A.9.6.2 Building an Unstructured User Interface Template ... A-111
A.9.6.3 Configuring a Page Group to Allow Use of UI Templates ... A-114
A.9.6.4 Applying a UI Template to a Page ... A-116
A.10 Example: Building Charts and Reports ... A-118
A.10.1 Exercise: Building the Team Details Report ... A-118
A.10.2 Exercise: Building the Average Salaries Chart ... A-120
A.10.3 Exercise: Building the Team Bonuses Report ... A-121

B Troubleshooting OracleAS Portal

B.1 Problems and Solutions ... B-1
B.1.1 Java Portlet Wizard Not Available ... B-1
B.1.2 Portlet Code Does Not Compile ... B-2
B.1.3 Application Server Connection Test Fails ... B-2
B.1.4 Provider Test Page Shows Error ... B-2
B.1.5 Provider Registration Fails ... B-3
B.1.6 Portlet Does Not Display on Page ... B-3
B.1.7 After Initial Successful Display, Portlet Does Not Display on Page ... B-4
B.1.8 Other Portlet Problems ... B-4
B.1.9 Provider Group Not Created ... B-5
B.1.10 URL Portlet Does Not Work ... B-5
B.2 Diagnosing OmniPortlet Problems ... B-5
B.2.1 Chart Not Rendered on UNIX ... B-6
B.2.2 Unable to Access HTTPS Site ... B-6
B.2.3 OmniPortlet Cannot Access the Specified URL ... B-6
B.2.4 Portlet Content Is Not Refreshed ... B-8
B.2.5 Edit Defaults Changes are Not Reflected in the Customized Portlet ... B-8
B.3 Diagnosing Web Clipping Problems ... B-9
B.3.1 Checking the Status of the Provider with the Test Page ... B-9
B.3.2 Solving Problems with Connections ... B-9
B.3.2.1 Configuring Proxy Servers ... B-10
B.3.2.2 Proxy Authentication ... B-10
B.3.3 Setting Logging Levels ... B-11
B.4 Need More Help? ... B-11

Glossary

Index

Send Us Your Comments

Oracle Application Server Portal Developer's Guide, 10*g* Release 2 (10.1.2)

Part No. B14134-01

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: appserverdocs_us@oracle.com
- FAX: 650-506-7375 Attn: Oracle Application Server Portal Documentation Manager
- Postal service:

  Oracle Corporation
  OracleAS Portal Documentation
  500 Oracle Parkway, 2OP8
  Redwood Shores, CA 94065
  U.S.A.

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

If you have problems with the software, please contact your local Oracle Support Services.

Preface

This manual describes how to build portlets for Oracle Application Server Portal (OracleAS Portal) using a variety of tools and technologies. This includes understanding the various technology choices open to you, choosing the technology that best meets your requirements, and using the appropriate tools to build and deploy your portlets.

Intended Audience

This manual is intended primarily for portal developers, but page designers may also find it useful.

The manual guides you through the process of first understanding and choosing a portlet technology, and then building your portlets with that technology. To find out which chapters will be of most interest to you, refer to the "Structure".

For information about the different privileges in OracleAS Portal and how these affect the tasks you can perform, see the *Oracle Application Server Portal User's Guide*.

What Is a Portal Developer? A portal developer is a user with the following global privileges: Create All Portal DB Providers and Manage All Shared Components. The main task of a portal developer is to build portlets and make them available to page designers and other users for inclusion on their pages. Since OracleAS Portal offers such a wide spectrum of tools and technologies for building portlets, a portal developer may or may not have substantial programming background.

What Is a Page Designer? A page designer (also known as a page manager) is a user with the Manage privilege on a page. A user with this privilege can perform any action on the page and can create sub-pages under the page. The page designer is often responsible for designing the layout (or region configuration) of the page and assigning privileges on the page to other users (for example, to determine who can add content to the page).

The scope of a page designer's control over a page may be limited if the page is based on a template.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. 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. 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 additional information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

JAWS, a Windows screen reader, 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, JAWS 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.

Structure

This manual is organized as follows:

Part I (Portlet Overview)

This part provides an introduction to portlets.

Chapter 1 (Understanding Portlets)

This chapter explains portlets, and how portlet development compares with traditional Web page development.

Part II (Portlet Technologies)

This part includes considerations for choosing a portlet building technology and the specific benefits of using each portlet technology to help you decide which technology is best suited for your purposes.

Chapter 2 (Portlet Technologies Matrix)

This chapter presents a summary of portlet features, characteristics, technologies, and tools. It is presented in a matrix format, and is designed to help users decide which portlet building technology suits their needs best. It lists the technologies and tools they can use with OracleAS Portal on one axis, and the features and characteristics on the other.

Part III (Building Portlets)

This part contains the chapters that explain how to use the portlet technologies discussed in this manual.

Chapter 3 (Building Portlets with OmniPortlet)

This chapter explains how to use the various tabs and controls in the OmniPortlet wizard, as well as how to extend the functionality of OmniPortlet by installing pluggable extensions.

Chapter 4 (Building Content-Based Portlets with Web Clipping)

This chapter explains how to use the Web Clipping Studio to create a portlet based on a Web clipping.

Chapter 5 (Building Java Portlets)

This chapter explains how to create Java portlets based on the JSR-168 portlet standard, how to build Java portlets using Oracle's Portal Developer Kit-Java, and how to make a portlet out of your struts application.

Chapter 6 (Building PL/SQL Portlets)

This chapter explains how to create PL/SQL portlets based on the Oracle Application Server Portal Developer Kit-PL/SQL (PDK-PL/SQL). To make effective use of this chapter, you should already know PL/SQL and have some familiarity with the PL/SQL Web Toolkit.

Part IV (Appendixes)

This part contains supporting information in appendixes.

Appendix A (Building Portlets with the Portlet Builder)

This appendix describes the process of creating a portlet through a wizard and steps you through creating, editing, managing, and running different types of portlets.

Appendix B (Troubleshooting OracleAS Portal)

This appendix explains how to troubleshoot your portlets.

Glossary

The glossary provides definitions for terms used in this and other OracleAS Portal manuals.

Related Documents

For more information, see the following manuals in the OracleAS Portal documentation set:

- Chapter 12, "Oracle Application Server Portal", in the *Oracle Application Server Release Notes*
- *Oracle Application Server Portal User's Guide*
- *Oracle Application Server Portal Configuration Guide*
- *Oracle Application Server Portal Error Messages Guide*

You may also find the following manuals in the Oracle Application Server documentation set useful:

- *Oracle Application Server Concepts*
- *Oracle HTTP Server Administrator's Guide*
- *Oracle Application Server Web Cache Administrator's Guide*

Conventions

The following conventions are also used in this manual:

Convention	Meaning
italicized text	Italicized type introduces important terms used for the first time.
boldface text	Boldface type is used for emphasis, and to represent the names of items as they appear on your screen.
CAPITALIZED text	Capitalized text indicates procedure names.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.

Browser Recommendations

When using OracleAS Portal, we recommend that you use one of the following Web browsers:

- Netscape 4.79
- Microsoft Internet Explorer 5.01 or 5.5 (with Service Pack 1) for Windows 2000
- Microsoft Internet Explorer 6.0 for Windows XP

You may encounter JavaScript errors if you use a browser older than the recommended minimum.

Cache Settings

To ensure that your browser is always displaying valid portal content, please make sure that your browser cache settings are as follows:

In Internet Explorer:

1. From the menu, choose **Tools > Internet Options**.
2. Make sure you are on the **General** tab.
3. In the Temporary Internet File section, click the **Settings** button.
4. In the **Check for newer versions of stored pages** radio group, select **Every visit to the page**.
5. Click **OK**.
6. Click **OK**.

In Netscape:

1. From the menu, choose **Edit > Preferences**.
2. Expand the **Advanced** node.
3. Click **Cache**.
4. In the **Document in cache is compared to document on network** radio group, select **Every time**.
5. Click **OK**.

Image Settings

Please make sure that images are automatically loaded as follows:

In Internet Explorer:

1. From the menu, choose **Tools > Internet Options**.
2. Click the **Advanced** tab.
3. Scroll through the list of options to the Multimedia node, and select **Show Pictures**.
4. Click **OK**.

In Netscape:

1. From the menu, choose **Edit > Preferences**.
2. Click **Advanced**.
3. Select the **Automatically load images** check box.
4. Click **OK**.

Sometimes this setting is disabled to increase performance on low bandwidth connections. However, one common problem that occurs when images are not automatically loaded is that once logged out, you cannot login again without closing and reinvoking the browser. Hence, we recommend that this setting is always enabled.

Part I

Portlet Overview

Part I contains the following chapter:

- Chapter 1, "Understanding Portlets"

1 Understanding Portlets

This chapter explains portlets, the anatomy of a portlet, and provides an overview of the various portlet-building tools available in OracleAS Portal:

- Introduction to Portal Development
- Understanding Portlets
- Portlet Anatomy
- Portlet Resources

1.1 Introduction to Portal Development

OracleAS Portal enables you to present information from multiple, unrelated data sources in one, organized view. This view, a portal page, can contain one or more components—called *portlets*—that can each get their content from different data sources.

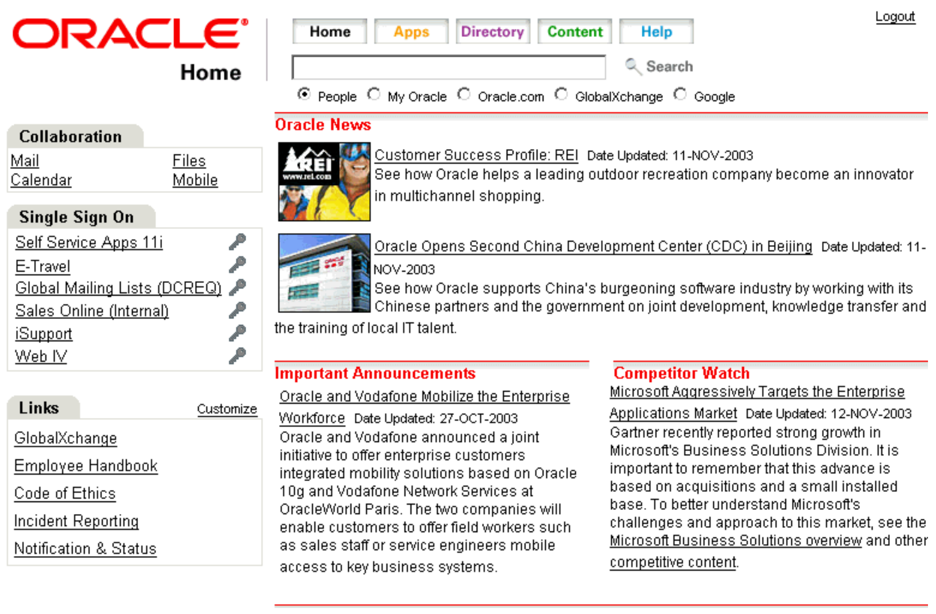
OracleAS Portal has all the tools you need for developing portlets and adding them to your portal pages. Portal's tools support a wide range of development skill: from the novice business developer to the experienced IT programmer. You can develop portlets either declaratively, through the Portal user interface, or programmatically, through Portal's collection of application programming interfaces (APIs), known as the Oracle Application Server Portal Developer Kit (PDK). Additionally, you can develop portlets through other development tools, external to OracleAS Portal, and integrate them through the PDK and a Portal entity called a *provider*. To learn more about providers, refer to Chapter 2, "Portlet Technologies Matrix".

This chapter defines portlets, lists and describes some sources for pre-built portlets and resources for building portlets, and suggests the best resource for the job.

1.2 Understanding Portlets

A portlet is a reusable, pluggable Web component that can draw content from many different sources. A typical portlet is one that displays summaries of Web content.

Figure 1–1 Portlets on the My Oracle Home Page



Collaboration

Mail Files
Calendar Mobile

Single Sign On

Self Service Apps 11i
E-Travel
Global Mailing Lists (DCREQ)
Sales Online (Internal)
iSupport
Web IV

Links Customize

GlobalXchange
Employee Handbook
Code of Ethics
Incident Reporting
Notification & Status

Oracle News

Customer Success Profile: REI Date Updated: 11-NOV-2003
See how Oracle helps a leading outdoor recreation company become an innovator in multichannel shopping.

Oracle Opens Second China Development Center (CDC) in Beijing Date Updated: 11-NOV-2003
See how Oracle supports China's burgeoning software industry by working with its Chinese partners and the government on joint development, knowledge transfer and the training of local IT talent.

Important Announcements

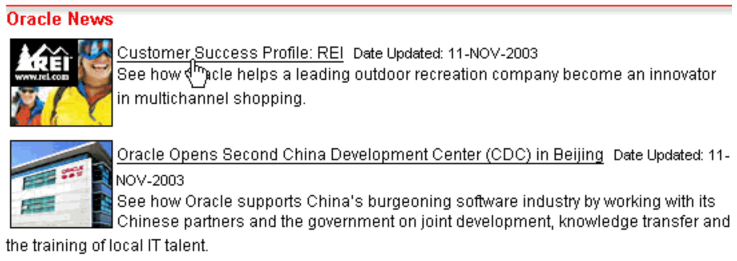
Oracle and Vodafone Mobilize the Enterprise Workforce Date Updated: 27-OCT-2003
Oracle and Vodafone announced a joint initiative to offer enterprise customers integrated mobility solutions based on Oracle 10g and Vodafone Network Services at OracleWorld Paris. The two companies will enable customers to offer field workers such as sales staff or service engineers mobile access to key business systems.

Competitor Watch

Microsoft Aggressively Targets the Enterprise Applications Market Date Updated: 12-NOV-2003
Gartner recently reported strong growth in Microsoft's Business Solutions Division. It is important to remember that this advance is based on acquisitions and a small installed base. To better understand Microsoft's challenges and approach to this market, see the Microsoft Business Solutions overview and other competitive content.

For example, in your portal you may have a news feed portlet that supplies linked news article headlines that are each accompanied by a sentence describing the content of the article (Figure 1–2).

Figure 1–2 The Oracle News Portlet on the My Oracle Home Page



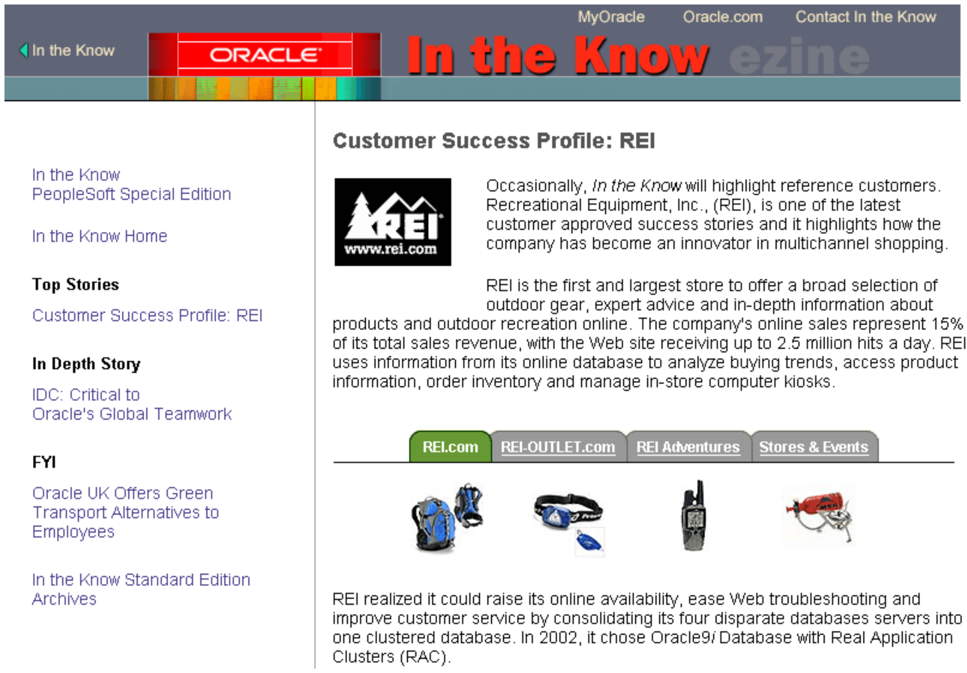
Oracle News

Customer Success Profile: REI Date Updated: 11-NOV-2003
See how Oracle helps a leading outdoor recreation company become an innovator in multichannel shopping.

Oracle Opens Second China Development Center (CDC) in Beijing Date Updated: 11-NOV-2003
See how Oracle supports China's burgeoning software industry by working with its Chinese partners and the government on joint development, knowledge transfer and the training of local IT talent.

Users click the linked headlines to get to the full text of the article, which is hosted on an external news service (Figure 1–3). The portlet has a somewhat dynamic nature in that headlines change automatically as news stories are added and removed at the source.

Figure 1–3 Content Target from a Portlet Link



In the Know
PeopleSoft Special Edition

In the Know Home

Top Stories

Customer Success Profile: REI

In Depth Story

IDC: Critical to
Oracle's Global Teamwork

FYI

Oracle UK Offers Green
Transport Alternatives to
Employees

In the Know Standard Edition
Archives

Customer Success Profile: REI

Occasionally, *In the Know* will highlight reference customers. Recreational Equipment, Inc., (REI), is one of the latest customer approved success stories and it highlights how the company has become an innovator in multichannel shopping.

REI is the first and largest store to offer a broad selection of outdoor gear, expert advice and in-depth information about products and outdoor recreation online. The company's online sales represent 15% of its total sales revenue, with the Web site receiving up to 2.5 million hits a day. REI uses information from its online database to analyze buying trends, access product information, order inventory and manage in-store computer kiosks.

REI realized it could raise its online availability, ease Web troubleshooting and improve customer service by consolidating its four disparate databases servers into one clustered database. In 2002, it chose Oracle9*i* Database with Real Application Clusters (RAC).

Portlets provide a seamless, single view of data that originates from multiple sources. Since different portlets can be placed on a common page, the user receives a single-source experience. In reality, the content is derived from multiple sources.

Portlets display excerpts of other Web sites, generate summaries of key information, perform searches, and access assembled collections of information from a variety of data sources.

1.3 Portlet Anatomy

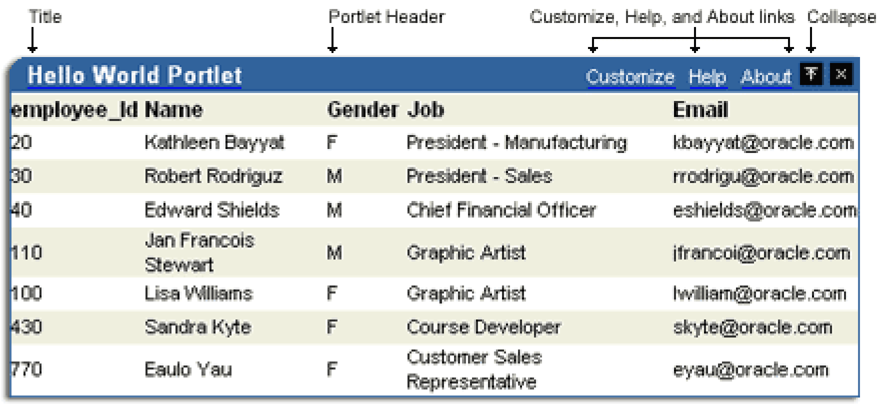
A portlet on a page is rendered in an HTML table cell. A portlet can display various types of content, such as HTML, formatted text, images, or elements of an HTML form.

Figure 1–4 shows the anatomy of a portlet, which includes a header that contains the portlet title. You can create a hyperlink in the portlet title, so that when a user clicks the title, the portlet displays in a full browser page. A portlet can also include a border, to distinguish the layout from other portlets on the page.

Every portlet includes three links: Customize, Help, and About. The portlet developer can expose these links to the page designer, who can then turn these on or off. Clicking the Customize link displays a number of options where the end user can personalize various attributes of the portlet. Clicking the Help link display a window containing help text that you can create to assist the end user with the portlet. Clicking the About link displays a window that you can create to describe the contents of the portlet.

Each portlet also contains the standard collapse icon, which the end user can click to collapse or expand the portlet on the page.

Figure 1–4 Portlet Anatomy



employee_Id	Name	Gender	Job	Email
20	Kathleen Bayyat	F	President - Manufacturing	kbayyat@oracle.com
30	Robert Rodriguz	M	President - Sales	rrodrigu@oracle.com
40	Edward Shields	M	Chief Financial Officer	eshields@oracle.com
110	Jan Francois Stewart	M	Graphic Artist	jfrancoi@oracle.com
100	Lisa Williams	F	Graphic Artist	lwilliam@oracle.com
430	Sandra Kyte	F	Course Developer	skyte@oracle.com
770	Eaulo Yau	F	Customer Sales Representative	eyau@oracle.com

1.4 Portlet Resources

Portlet resources include the many pre-built portlets available out-of-the-box from many sources, including OracleAS Portal, Oracle E-Business Suite, and third-party sources. Portlet resources also include portlet-building tools available through the Portal user interface as well as from the PDK and other Oracle tools. Each of these tools offers different product features that are targeted toward different developer roles.

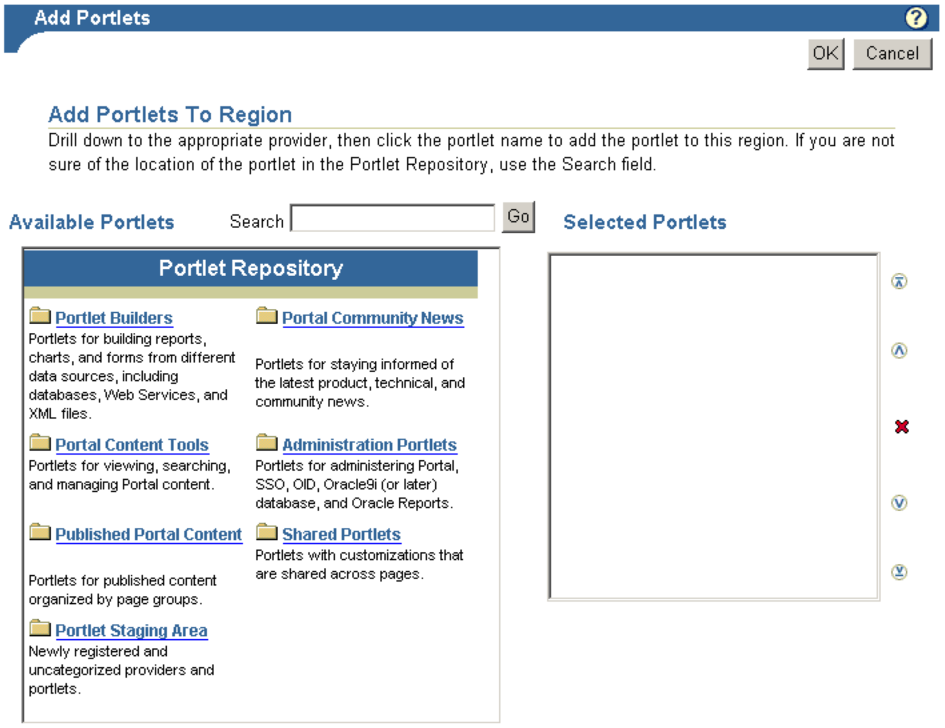
This section describes different portlet resources, suggests the level of expertise required to use them, and provides examples of when they might best be used. It includes the following subsections:

- Out-of-the-Box Portlets
- Other Sources of Pre-Built Portlets
- Web Clipping
- OmniPortlet
- Portlet Builder
- Programmatic Portlets

This section introduces you to the various portlet resources. For specific information on each tool and its benefits, refer to Chapter 2, "Portlet Technologies Matrix".

1.4.1 Out-of-the-Box Portlets

Figure 1–5 The Portlet Repository



Add Portlets To Region

Drill down to the appropriate provider, then click the portlet name to add the portlet to this region. If you are not sure of the location of the portlet in the Portlet Repository, use the Search field.

Portlet Builders
Portlets for building reports, charts, and forms from different data sources, including databases, Web Services, and XML files.

Portal Community News
Portlets for staying informed of the latest product, technical, and community news.

Portal Content Tools
Portlets for viewing, searching, and managing Portal content.

Administration Portlets
Portlets for administering Portal, SSO, OID, Oracle9i (or later) database, and Oracle Reports.

Published Portal Content
Portlets for published content organized by page groups.

Shared Portlets
Portlets with customizations that are shared across pages.

Portlet Staging Area
Newly registered and uncategorized providers and portlets.

What Is It?

Out-of-the-box portlets are fully developed, registered portlets that are immediately available from the Portlet Repository when you install OracleAS Portal (Figure 1–5). They include such portlets as Search, Saved Searches, Favorites, and My Notifications.

Note: You'll find information on the pre-built portlets in OracleAS Portal in the *Oracle Application Server Portal User's Guide*, available on the Oracle Application Server documentation CD and on Portal Center (`http://www.oracle.com/technology/products/ias/portal/index.html`, then click the **Search** icon in the right-hand corner of any page).

Who Is the Intended User?

Out-of-the-box portlets are best suited for use by end users and page designers, though they are available to users at all levels of expertise.

When Should It Be Used?

Use out-of-the-box portlets when your needs are satisfied by the functions the portlets offer, and the level of customization readily available is sufficient to complete the desired task.

Consider alternatives when you need to extend or customize the portlet, for example, you must change the user interface, or when the functionality you require is not available out of the box.

For more information on when you should use each technology, refer to Chapter 2, "Portlet Technologies Matrix".

1.4.2 Other Sources of Pre-Built Portlets

What Is It?

Other sources of pre-built portlets include partner portlets, integration solutions, and the Portal Knowledge Exchange.

Partner portlets are available through Oracle's partnerships with different types of leading system integrators, software vendors, and content providers. You can access these portlets through the Portal Catalog, available on Portal Center (`http://www.oracle.com/technology/products/ias/portal/index.html`). Examples of these include portlets for:

- Generating point-to-point driving directions
- Accessing IT information from a wide variety of sources
- Viewing summary information on news, stocks, and weather

Portal Center also provides integration solutions that are useful for customers who require basic functionality for popular applications such as Microsoft Exchange, Lotus Notes, SAP, IMAP, SMTP, and the like.

The Portal Knowledge Exchange, also accessible on Portal Center, is an offering from Portal Developer Services. Community members exchange portal expertise that includes portlet samples, tips, white papers, sample code, and the like. Members receive a personal folder on Portal Center, which they can use to upload portlet code and portal development insights and download and rate contributions from other developers.

Who Is the Intended User?

Fully developed, downloadable portlets are best suited for use by end users and page designers who understand how to download, install, and register Web and database providers in OracleAS Portal. They are available for use by all levels of experience.

When Should It Be Used?

Just like out-of-the-box portlets, use pre-built portlets from other sources when your needs are satisfied by the functions the portlets offer, and the level of customization readily available is sufficient to complete the desired task.

Consider alternatives when you need to extend or customize the portlet, for example, when you must change the user interface or when the functionality you require is not available out of the box.

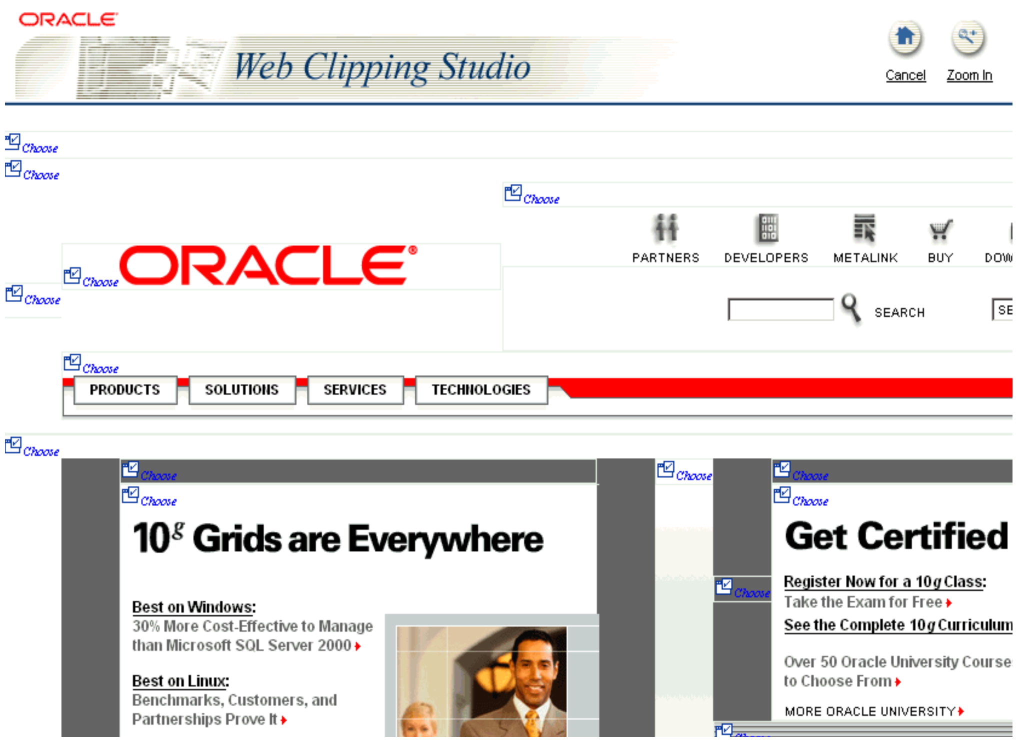
1.4.3 Web Clipping

What Is It?

Web Clipping is a browser-based declarative tool that enables you to integrate any Web application with OracleAS Portal. It is designed to give you quick integration by leveraging the Web application's existing user interface. Web Clipping has been implemented as a Web provider using the PDK-Java, which is a component of OracleAS Portal.

To create a Web Clipping portlet, the portal page designer uses a Web browser to navigate to the Web page that contains the desired content. Through the Web Clipping Studio, users can drill down through a visual rendering of the target page to choose just the content they want (Figure 1–6 and Figure 1–7).

Figure 1–6 Selecting Web Content through the Web Clipping Studio



10g Grids are Everywhere

Best on Windows:
30% More Cost-Effective to Manage than Microsoft SQL Server 2000

Best on Linux:
Benchmarks, Customers, and Partnerships Prove It

Get Certified

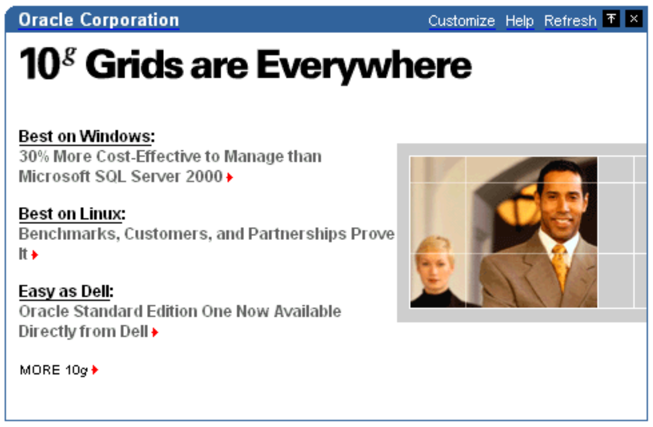
Register Now for a 10g Class:
Take the Exam for Free

See the Complete 10g Curriculum

Over 50 Oracle University Course to Choose From

MORE ORACLE UNIVERSITY

Figure 1–7 Clipped Content Rendered as a Portlet in Portal



10g Grids are Everywhere

Best on Windows:
30% More Cost-Effective to Manage than Microsoft SQL Server 2000

Best on Linux:
Benchmarks, Customers, and Partnerships Prove It

Easy as Dell:
Oracle Standard Edition One Now Available Directly from Dell

MORE 10g

Web Clipping supports:

- **Navigation through various styles of login mechanisms,** including form- and JavaScript-based submission and HTTP Basic and Digest Authentication with cookie-based session management.

- **Fuzzy matching of clippings.** If a Web clipping gets reordered within the source page or if its character font, size, or style changes, it will still be identified correctly by the Web Clipping engine and delivered as the portlet content.
- **Reuse of a wide range of Web content,** including basic support of pages written with HTML 4.0.1, JavaScript, applets, and plug-in enabled content, retrieved through HTTP GET and POST (form submission).
- **Customization,** allowing a page designer to expose input parameters that page viewers can modify when they customize the portlet. These parameters can be exposed as public parameters that a page designer can map as OracleAS Portal page parameters. This feature allows end users to obtain personalized clippings.
- **Integrate authenticated web content through Single Sign-On,** including integration with external applications, which enables you to leverage Oracle Application Server Single Sign-On and to clip content from authenticated external Web sites.
- **Inline rendering,** enabling you to set up Web Clipping portlets to display links within the context of the portlet. As a result, when a user clicks a link in the Web Clipping portlet, the results display within the same portlet. You can use this feature with internal and external Web sites.
- **Proxy Authentication,** including support for global proxy authentication and per-user authentication. You can specify the realm of the proxy server and whether all users will automatically log in using a user name and password you provide, each user will log in using an individual user name and password, or all users will log in using a specified user name and password.
- **Migration from URL-based portlets,** enabling you to migrate your URL-based portlets to Web Clipping.

  Note that although the URL was valid when this document was written, the URL may change in the future. You should be redirected to the new URLpage.

Who Is the Intended User?

Web Clipping is best suited for use by page designers and portlet developers who want to leverage an existing Web page for rapid portlet development. The Web Clipping portlet is accessible out of the box, and is available in the Portlet Repository of OracleAS Portal. This portlet can be added to a page by any user with the right privileges.

When Should It Be Used?

Use Web Clipping when you want to copy content and functionality from an existing Web page and expose it in your portal as a portlet. Consider alternatives if you want to change the way information is presented in the clipped portlet. That is, you don’t need to control the UI or application flow, and you are accessing Web-based applications. For a greater level of control, use the OracleAS Portal OmniPortlet’s Web page data source in lieu of Web Clipping.

Here are some examples of when you may consider using the Web Clipping portlet:

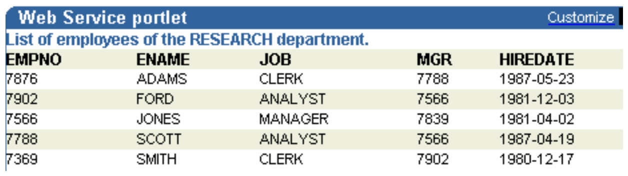
- Stock chart portlet. Suppose you want to create a portlet that displays the stock market’s daily performance chart from your financial advisor’s Web site. You could clip this information from an external Web site, even if your company is using a proxy.
- Personalized weather portlet. Suppose you want to create a portlet that displays weather information from a major Internet weather site, and you want your users to be able to personalize the portlet by providing the desired zip code.

- Web mail portlet. Suppose your users want to access their confidential Web mail accounts through a portlet, and their inboxes to display in the portlet.

For more information on using Web Clipping, refer to Chapter 4, "Building Content-Based Portlets with Web Clipping".

1.4.4 OmniPortlet

Figure 1–8 An OmniPortlet Using Tabular Format



List of employees of the RESEARCH department.

EMPNO	ENAME	JOB	MGR	HIREDATE
7876	ADAMS	CLERK	7788	1987-05-23
7902	FORD	ANALYST	7566	1981-12-03
7566	JONES	MANAGER	7839	1981-04-02
7788	SCOTT	ANALYST	7566	1987-04-19
7369	SMITH	CLERK	7902	1980-12-17

What Is It?

The OracleAS Portal OmniPortlet is a declarative portlet-building tool that enables you to build portlets against a variety of data sources, including XML files, comma-separated value files (for example, spreadsheets), Web Services, databases, Web pages, and SAP data sources. OmniPortlet users can also choose a pre-built layout for the data. Pre-built layouts include tabular, news, bullet, form, or chart.

Like Web Clipping, OmniPortlet supports proxy authentication, including support for global proxy authentication and per-user authentication. You can specify whether all users will automatically log in using a user name and password you provide, each user will log in using an individual user name and password, or all users will log in using a specified user name and password.

Note: You'll find information about OmniPortlet on Portal Center, (`http://www.oracle.com/technology/products/ias/portal/index.html`).

Who Is the Intended User?

OmniPortlet is best suited for use by page designers and developers.

When Should It Be Used?

Use OmniPortlet when you want to build portlets rapidly against a variety of data sources with a variety of layouts. Consider alternatives when you want complete control of the design and functionality of the portlet.

Here are some examples of when you may consider using OmniPortlet:

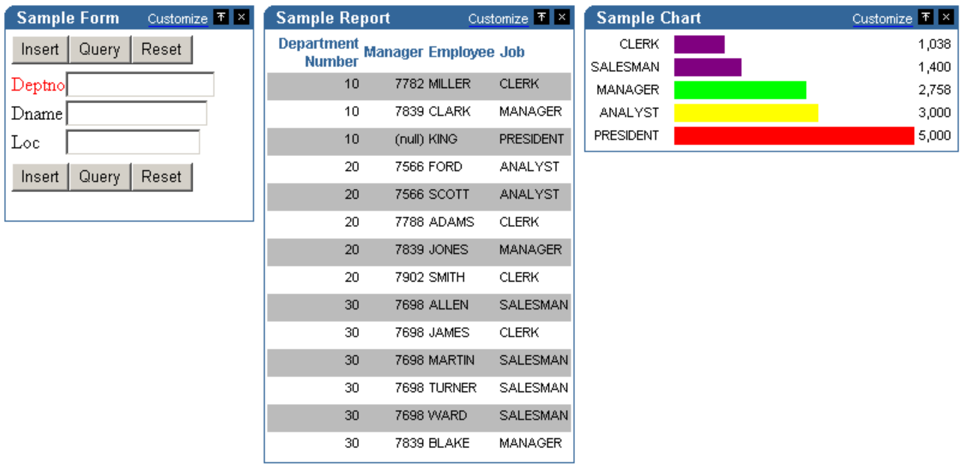
- RSS news feed portlet. Suppose you want to create a portlet that displays live, scrolling news information to your users. The data comes from a Really Simple Syndication news feed, such as the Oracle Technology Network Headlines. You also want the portlet to contain hyperlinks to the news source.
- Sales chart portlet. Suppose you want to present up-to-date information on your company's sales results. You also want to display data in the form of a pie chart, and your company stores its sales information in a remote relational database.
- SAP portlet. Suppose you want to display information from a company's SAP system. To minimize the load on the company's SAP Business Suite, the

information retrieved from the system must be cached on a per user basis for the entire day.

For more information on OmniPortlet, refer to Chapter 3, "Building Portlets with OmniPortlet".

1.4.5 Portlet Builder

Figure 1–9 Sample Form, Report, and Chart from the Portlet Builder



Department Number	Manager Employee	Job
10	7782 MILLER	CLERK
10	7839 CLARK	MANAGER
10	(null) KING	PRESIDENT
20	7566 FORD	ANALYST
20	7566 SCOTT	ANALYST
20	7788 ADAMS	CLERK
20	7839 JONES	MANAGER
20	7902 SMITH	CLERK
30	7698 ALLEN	SALESMAN
30	7698 JAMES	CLERK
30	7698 MARTIN	SALESMAN
30	7698 TURNER	SALESMAN
30	7698 WARD	SALESMAN
30	7839 BLAKE	MANAGER

CLERK	1,038
SALESMAN	1,400
MANAGER	2,758
ANALYST	3,000
PRESIDENT	5,000

What Is It?

OracleAS Portal includes a number of portlet-building wizards that are accessible through the Provider tab in the Portal Navigator. These wizards can be used to build charts, reports, forms, calendars, and lists of values.

When Should It Be Used?

It is recommended that you use OmniPortlet as an alternative to Portlet Builder whenever possible. OmniPortlet provides more flexibility and a separation of data and layout which enables you to change from a report to chart without re-creating the entire portlet (as is required with Portlet Builder). OmniPortlet also provides more options for deployment to many different portals simultaneously. OracleAS Portal will continue to support Portlet Builder as a portlet building option. However, new features and enhancements will be directed toward the OmniPortlet tool.

For more information on Portlet Builder, refer to Appendix A, "Building Portlets with the Portlet Builder".

1.4.6 Programmatic Portlets

What Is It?

The OracleAS PDK contains a set of portlet-building APIs that you can use to create programmatic portlets.

Note: You'll find more information about these APIs on Portal Center (`http://www.oracle.com/technology/products/ias/portal/index.html`).

Who Is the Intended User?

These tools are best used by experienced and knowledgeable IT developers.

When Should It Be Used?

Use the PDK when you have very specialized business rules or logic and when you require customized authentication, granular processing of dynamic results, and complete user interface control. Additionally, use the PDK when:

- You're building a portlet from the start and need complete control over all of its functionality.
- You know Java or PL/SQL.
- You'd like a functioning starting point.
- You are comfortable with the PDK and the configuration of OracleAS Portal Providers.

Consider using this approach when the out-of-the-box declarative tools do not address your needs.

Note: The PDK-PL/SQL is not described in detail in this manual. For specific information on the PDK-PL/SQL, refer to the Developer Services area on Portal Center (`http://www.oracle.com/technology/products/ias/portal/index.html`).

Here are some examples of when you may consider using Java portlets created with Oracle Application Server Portal Developer Kit:

- Discussion forum portlet. Suppose you want to create a portlet that integrates your company's JSP-based discussion forum application with OracleAS Portal. The discussion forum posts are stored in a relational database. The portlet must also follow the strict look and feel of your company's Internet Web site.
- Email portlet. Suppose you want to create a portlet that enables users to send email from the company's intranet portal. You must integrate the email portlet with the company's LDAP server so that the users can use the address book on the LDAP server.

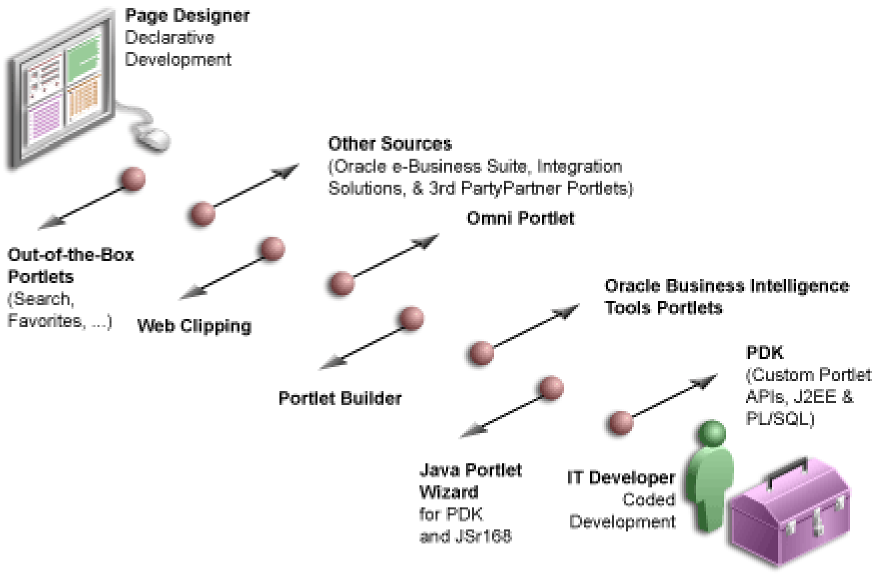
For more information on using the PDK-Java and PDK-PL/SQL, refer to Chapter 5, "Building Java Portlets" and Chapter 6, "Building PL/SQL Portlets".

1.4.7 Deciding Which Tool to Use

Figure 1–10 illustrates the spectrum of portlet resources described in the previous section. Notice how one end of the spectrum is geared toward a page designer while the other end speaks to the portlet developer. You can choose your tool depending on which type of user mostly closely approximates your expertise.

For more information on deciding which tool to use, refer to Chapter 2, "Portlet Technologies Matrix".

Figure 1–10 Portlet Resources from Page Designers to Experienced Developers



1.5 Summary

In this chapter, you learned what portlets are, as well as the various technologies available in OracleAS Portal. For more information on these tools and technologies, refer to Chapter 2, "Portlet Technologies Matrix".

Part II

Portlet Technologies

Part II contains the following chapter:

- Chapter 2, "Portlet Technologies Matrix"

2

Portlet Technologies Matrix

This chapter describes portlet features, characteristics, technologies, and tools to help you decide which portlet building technology best suits your needs. It includes the following sections:

- The Portlet Technologies Matrix
- General Suitability
- Expertise Required
- Deployment Type
- Caching Style
- Development Tool
- Portlet Creation Style
- User Interface Flexibility
- Ability to Capture Content from Web Sites
- Ability to Render Content Inline
- Charting Capability
- Public Portlet Parameters Support
- Private Portlet Parameter Support
- Event Support
- Ability to Hide and Show Portlets Based on User Privileges
- Multi-lingual Support
- Pagination Support
- Single Sign-On and External Application Integration

2.1 The Portlet Technologies Matrix

Table 2–1, " Portlet Building Technologies Comparison Matrix" summarizes the technologies and tools you can use with OracleAS Portal on one axis, and the features and characteristics on the other. The matrix contains those tools and technologies that are covered in this book in detail: OmniPortlet, Web Clipping, the Java Portlets (PDK-Java) including Standards, Portlet Builder as an appendix, and PL/SQL Portlets (PDK-PL/SQL) (in the matrix only).

Note: While these are the primary tools for building portlets, additional tools and technologies exist, such as other Oracle products, including Oracle Reports and Oracle Discoverer. These other tools are not covered in this book.

The other sections in this chapter provide further detail on the characteristics listed in Table 2-1. Use the table to quickly scan all the features and characteristics, then refer to the subsequent sections for more in-depth information.

Table 2–1 Portlet Building Technologies Comparison Matrix

Web Clipping	OmniPortlet	PDK-Java	Standards	Portlet Builder	PDK-PL/SQL
General Suitability					
A simple wizard-based tool that helps you retrieve and present Web content, originating from other Web sites, in your portal.	Wizard-based tool, accessible from the browser. Capable of retrieving and presenting data from a wide variety of data sources.	APIs for portlets built specifically for OracleAS Portal. The most flexible programmatic approach.	Portlets that should work with portals of other vendors. Oracle supports both WSRP and JSR-168.	Wizard-based tool, accessible from the browser. Best suited for simple, DB-centric applications or portlets.	APIs for portlets built specifically for OracleAS Portal. The most flexible programmatic approach.
Expertise Required					
No expertise required.	Basic understanding of one or more supported data sources and the concepts of portlet and page parameters and events.	Java, Servlet, JSP knowledge.	Java, Servlet, JSP knowledge.	Basic understanding of relational DB concepts. Optionally SQL, PL/SQL.	SQL, PL/SQL, PL/SQL Web Toolkit.
Supported Data Sources (for details, see **Expertise Required**)					
Any Web site accessible on the network over HTTP or HTTPS.	CSV, XML, Web Service, SAP, SQL, Web site, JCA.	No limitations.	No limitations.	SQL (local DB or remote DB via DB link)	SQL (local DB or remote DB via DB link)
Deployment Type					
Web provider	Web provider	Web provider	WSRP	Database provider	Database provider
Caching Style					
Expiry-based caching, invalidation-based caching (auto invalidate when customized).	Expiry-based caching, invalidation-based caching (auto invalidate when customized).	Expiry-based, validation, and invalidation caching, ESI.	Validation and expiry-based caching.	Expiry-based caching.	Expiry-based, validation, and invalidation caching.
Development Tool					
Browser - wizard.	Browser - wizard.	Oracle JDeveloper - Java Portlet Wizard (or any other Java development environment - without the Wizard).	Oracle JDeveloper - Java Portlet Wizard (or any other Java development environment - without the Wizard).	Browser - optionally PL/SQL development environment.	PL/SQL development environment.
Portlet Creation Style					

Table 2–1 (Cont.) Portlet Building Technologies Comparison Matrix

Web Clipping	OmniPortlet	PDK-Java	Standards	Portlet Builder	PDK-PL/SQL
Develop in place.	Develop in place.	No in-place portlet building experience. Add portlet to page, edit defaults, and personalize.	No in-place portlet building experience. Add portlet to page, edit defaults, and personalize.	Develop first, then add and Develop in place.	No in-place portlet building experience. Add portlet to page, edit defaults, and personalize.
User Interface Flexibility					
N/A	Limited.	Very flexible.	Very flexible	Limited.	Very flexible.
Ability to Capture Content from Web Sites					
Yes, by its nature.	Yes, by using the Web Data Source.	Yes, by using the java.net package	Yes, by using the java.net package	No	Yes, by using the UTIL_HTTP package.
Ability to Render Content Inline					
Yes. (Supported by Web Clipping 9.0.4.0.2 or later.)	No URL rewriting support, but can be achieved by using public portlet parameters and events.	By using private portlet parameters.	Include servlets and JSPs (using the PortletContext.getRequestDispatcher() method).	Pagination in reports and charts is rendered inline.	By using private portlet parameters.
Charting Capability					
N/A	Yes, 2D-3D charts.	By using BI Beans.	By using BI Beans.	HTML charts.	Programmatically, HTML charts.
Public Portlet Parameters Support					
Yes. (Supported by Web Clipping 9.0.4.0.2 or later.)	Yes	Yes	No	Yes	Yes
Private Portlet Parameter Support					
N/A	N/A	Yes	Yes	No	Yes
Event Support					
Yes	Yes	Yes	Portlet private events (actions).	No	No
Ability to Hide and Show Portlets Based on User Privileges					
No, though it is possible to apply security managers that are not exposed through the UI.	No, though it is possible to apply security managers that are not exposed through the UI.	Yes, by using the Security managers.	Yes, the Servlet security model is supported by using methods such as PortletRequest.isUserInRole() and PortletRequest.getUserPrincipal().	Yes	Yes, by using the Security APIs.
Multi-lingual Support					
N/A	Yes	Yes	Yes	No	Yes
Pagination Support					
N/A	No	Programmatically	Programmatically	Yes	Programmatically
Single Sign-On and External Application Integration					
Web Clipping 9.0.4.0.2 and above supports external application integration.	Basic authentication support if the data source requires it.	External application integration supported. LDAP integration is supported when the portlet is running behind the same firewall as the LDAP server.	No. (Feasible through custom user attributes.) LDAP integration is supported.	No. (It runs in the OracleAS Portal repository, it does not require SSO integration.)	SSO is enabled by using mod_oso.

2.2 General Suitability

This section describes each portlet-building technology in terms of its usage characteristics (for example, wizard-based or programmatic).

2.2.1 Web Clipping

Web Clipping is a simple wizard-based tool that helps you retrieve and present Web content, originating from other Web sites, in your portal. Web Clipping does not require any technical background.

2.2.1.1 Examples of portlets you can build using Web Clipping

- Stock chart portlet
- Personalized weather portlet
- Web mail portlet

2.2.2 OmniPortlet

If you are looking for an easy to use, wizard-based tool to present information from a wide variety of data sources, you should consider OmniPortlet. OmniPortlet runs completely in the browser. All you need to do is drop your OmniPortlet on a portal page, select your data source from a list of available data sources.

- Spreadsheet
- SQL
- XML
- Web Service
- Web page

Although OmniPortlet does not require an additional development tool or a strong technical background, you can build reusable and high-performing portlets with it.

2.2.2.1 Examples of portlets you can create with OmniPortlet

- RSS news feed portlet
- Sales chart portlet
- SAP Business Suite portlet

2.2.3 Java Portlets

If the wizard-based portlet building tools do not satisfy your needs, you can build your portlets programmatically using Java. The Java Community Process standardized the Java portlet APIs in 2003. Portlets built against the Java Specification Request (JSR) 168 standard are interoperable across different portal platforms. The Java Portlet Wizard, an Oracle JDeveloper plug-in, helps you get started with your Java portlets.

Note: When building portlets in Java, you have full control over your portlet's functionality. For example, you can control what it looks like and how it behaves.

2.2.3.1 Examples of portlets you can build using Java

- Discussion forum portlet
- Email portlet

2.2.4 Portlet Builder

Portlet Builder is a wizard-based tool to create data-driven portlets, where the data resides in an Oracle database. You can build interactive forms to insert, update, and delete database records. You can create flexible reports and HTML bar charts to display information from the database. Portlet Builder also enables you to pass parameters and navigate between your data-driven portlets by using dynamic links.

2.2.4.1 Examples of portlets you can build using the Portlet Builder

- Data entry portlet
- Dynamic list of partners portlet
- Sales results portlet

2.2.5 PL/SQL Portlets

Similar to Java portlets, PL/SQL portlets provide a flexible approach to build Web applications that cannot be satisfied by built-in portlets. For example, your application may require implementation of special business rules or logic or meet custom-designed authorization requirements. PL/SQL portlets are commonly used when you need to perform data intensive operations by using SQL and PL/SQL. OracleAS Portal offers a rich set of PL/SQL APIs such as programmatic provider registration, object level privilege management, user interface control, or multilingual support.

For example, any information provider can create custom portlets to display an application to users through OracleAS Portal. Developers simply build their portlets according to OracleAS Portal Developer Kit (PDK) specifications and register the provider with OracleAS Portal. Developers can use PDK to develop portlets to suit their needs.

2.2.5.1 Examples of portlets you can build using PL/SQL

- Content upload portlet
- Site map portlet
- Sophisticated data entry and report portlet

2.3 Expertise Required

While some of the portlet building tools do not require portlet development skills, others assume a strong technical background. This section describes each tool in terms of the level of knowledge required to use it effectively.

2.3.1 Web Clipping

Web Clipping is a tool that does not require any technical background at all. However, if you want to parameterize the Web page content that you clipped, you need to have an understanding of public portlet parameters and page parameters.

2.3.2 OmniPortlet

OmniPortlet requires you to have basic knowledge of the data source you want to leverage in your portlet.

Data source	What you need to know about the data source
Spreadsheet	The URL that points to the spreadsheet containing the data that you want to display in the portlet.
SQL	The connection information to the data source and the SQL query that retrieves the data from the database.
XML	The location of the XML source and optionally the address of the XSL filter and the XML schema.
Web service	The WSDL URL, the method of the Web service, and optionally the XSL filter URL and the XML schema URL.
Web page	The Web page data source uses the same environment as Web Clipping. No technical background is required.
J2EE Connector Architecture	Although not displayed on the Type page of the OmniPortlet wizard, a J2EEtm Connector Architecture (JCA) 1.0 adapter is also available. JCA provides a mechanism to store and retrieve enterprise data such as that held in ERP systems (Oracle Financials, SAP, PeopleSoft, and so on).

2.3.3 Java Portlets

To build Java portlets, you must know at least a subset of J2EE. Knowing HTML, Java servlets and XML is a must, and JSP experience is recommended. Additional Java knowledge is optional, depending on the task you want to perform. Using Java portlets you can access any data source (supported by the Java language).

2.3.4 Portlet Builder

If you want to use Portlet Builder, you must have a good understanding of relational database concepts. Depending on what you want to achieve, SQL and/or PL/SQL knowledge may be required, as well. Using Portlet Builder, you can consume data from the local (Oracle Application Server infrastructure) database or remote databases using database links.

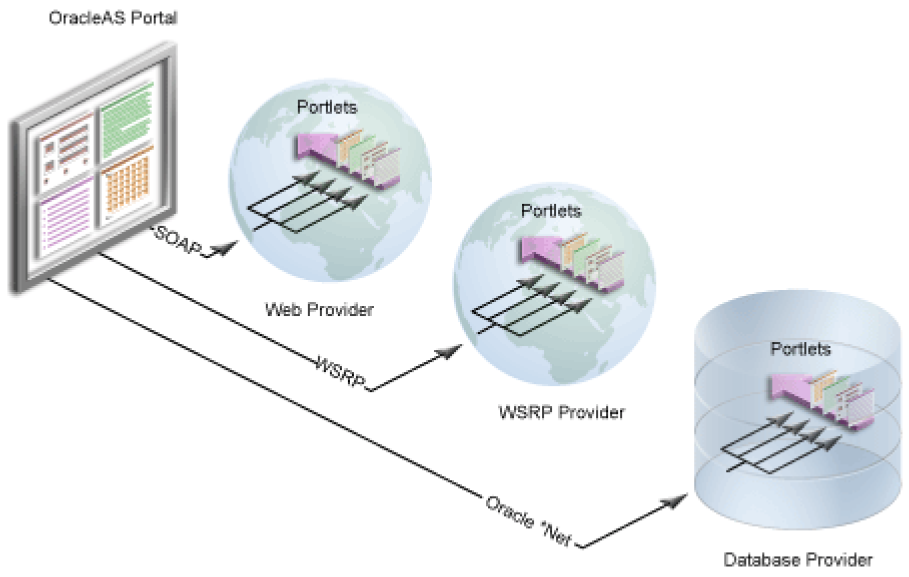
2.3.5 PL/SQL Portlets

To build PL/SQL portlets, you must know how to write SQL statements, code and debug PL/SQL program units using SQL*Plus or similar development tool that enables you to connect to Oracle database. You should also know HTML and PL/SQL Web Toolkit to generate the portlet content. Experience of coding PL/SQL Server Pages (PSP) is optional.

2.4 Deployment Type

As shown in Figure 2-1, portlets can be deployed to OracleAS Portal through three provider types: Web providers, WSRP providers, and database providers. Web providers are deployed to a J2EE application server, which is often remote and communicates with OracleAS Portal through Simple Object Access Protocol (SOAP) over HTTP. Web Services for Remote Portlets (WSRP), an OASIS standard, is supported in the Developer's Preview of OracleAS Portal. Database providers are implemented in PL/SQL and deployed in the Oracle database where OracleAS Portal is installed.

Figure 2–1 Portlet Provider Overview



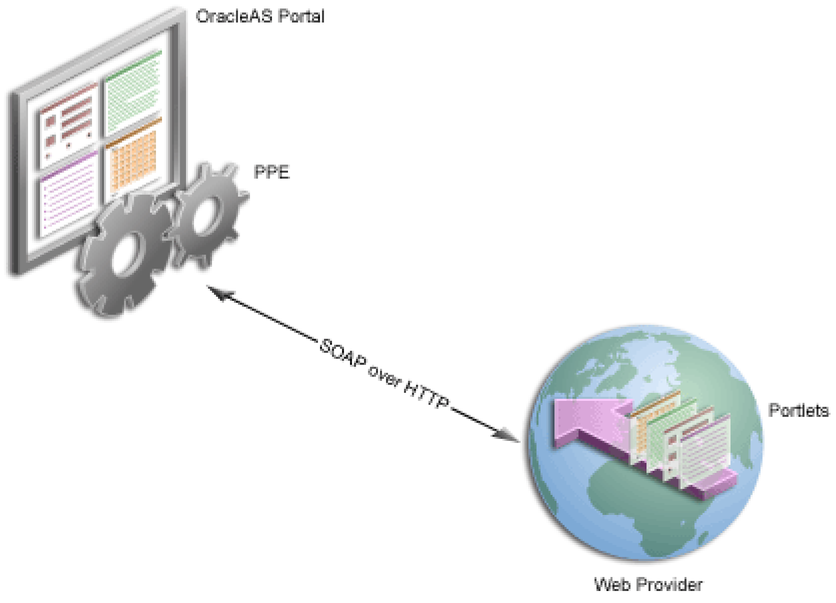
2.4.1 Web Providers

Web providers are the most commonly used and flexible type of provider. They may reside on the same application server as OracleAS Portal, on a remote application server, or anywhere on the network. A Web provider could be implemented using virtually any Web technology. However, the Oracle Application Server Portal Developer Kit provides a Java framework that simplifies the task of building Web providers.

Web providers use open standards, such as XML, SOAP, HTTP, or J2EE for deployment, definition, and communication with OracleAS Portal. Also, because Web providers can be deployed to a J2EE container, they do not put an additional load on the OracleAS Portal Repository database.

Note: While in most cases Web providers are more beneficial than database providers, note that Web providers do not support the export and import of system level customization (edit defaults) and end user personalization.

Figure 2–2 Web Providers



There are several benefits when developing portlets and exposing them as Web providers:

- Deploy portlets remotely.
- Leverage existing Web application code to create portlets.
- Declarative specification of providers.
- More functionality than database providers.
- Portlets of Web providers can be developed using standard Java technologies (for example, servlets and JSPs).

To expose your portlets using a Web provider, you must create a provider that manages your portlets and can communicate with OracleAS Portal using SOAP. To learn how to expose your portlets using a Web provider, visit Portal Studio (`http://www.oracle.com/technology/products/ias/portal/index.html`), click **General**, then refer to "An Overview of Writing Portlets for Web Providers."

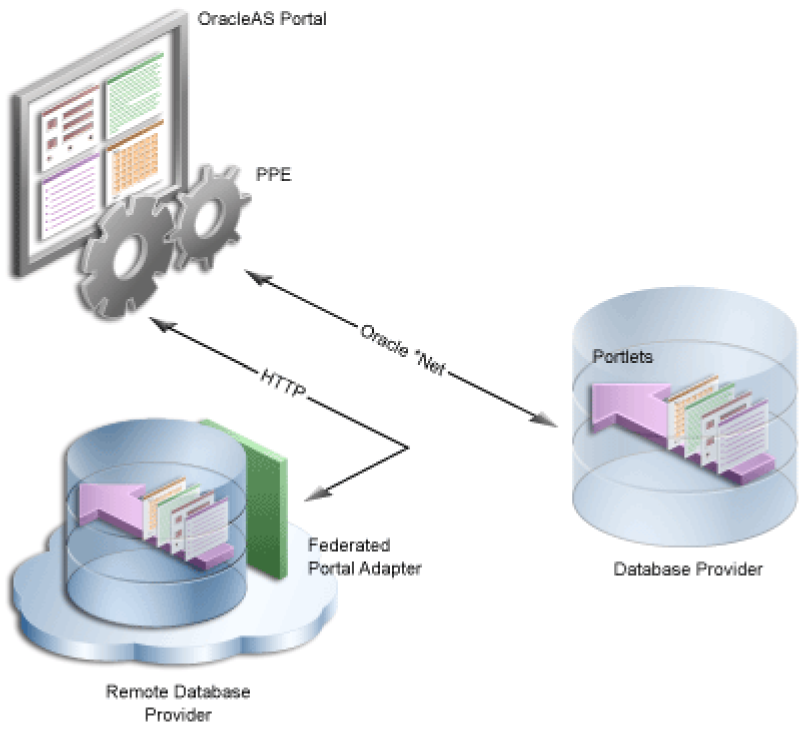
2.4.2 WSRP Providers

WSRP, a Web services standard, allows interoperability between a standards enabled container and any WSRP portal. From an architecture perspective, WSRP is very similar to Web providers.

2.4.3 Database Providers

You can also create a database provider that owns one or more PL/SQL portlets. Database provider and their PL/SQL portlets reside in the Oracle Application Server Metadata Repository database and are implemented as PL/SQL packages. To access database providers on remote servers, you can use the Federated Portal Adapter. For more information, see "Understanding the Federated Portal Adapter" on the **General** page of Portal Studio (`http://www.oracle.com/technology/products/ias/portal/index.html`).

Figure 2–3 Database Providers



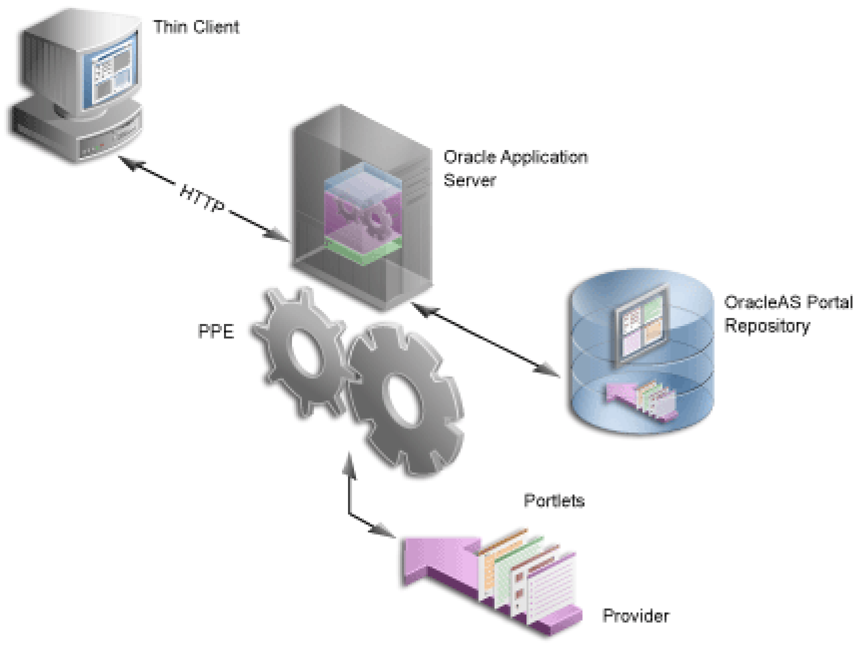
Database providers are ideal when you must perform data-intensive operations using PL/SQL. An example of this is when you are building forms or charts with the OracleAS Portal user interface or the PL/SQL APIs provided in the PDK.

To learn how to expose your PL/SQL portlets using a database provider, visit Portal Studio, click **General**, then refer to "An Overview of Writing Portlets for Database Providers."

Note: To learn more about portlets, refer to the Documentation page on Portal Center (`http://www.oracle.com/technology/products/ias/portal/index.html`).

2.4.4 Provider Architecture

Figure 2–4 Provider Architecture



This figure illustrates the basic architecture of portlet providers. When users display the portal page in their Web browsers, the flow of the request works like this:

1. The user requests a portal page from the Web browser by entering a URL in the browser's address field.
2. The Parallel Page Engine (PPE), which resides in the Oracle Application Server's middle tier, retrieves the portal page layout information (also called the page metadata) from the OracleAS Portal Repository.

   Note: The PPE is responsible for constructing the requested portal page based on the page metadata.

3. The PPE contacts all the providers for the portlet content.
4. The providers make the necessary calls to their portlets so that the portlets generate the portlet content in the form of HTML or XML code.
5. The providers return the portlet content back to the PPE.
6. The PPE assembles the portal page, and the Oracle Application Server returns the page to the Web browser.

   Note: For more information about the portlet and provider architecture, visit Portal Center (`http://www.oracle.com/technology/products/ias/portal/index.html`), then, under Developer Services, click **Portal Studio/PDK**.

Web Clipping, OmniPortlet, and Java portlets communicate with OracleAS Portal through Web providers. After you install OracleAS Portal, Web Clipping and OmniPortlet are ready to use; their providers are registered with OracleAS Portal out of the box. You have to register the provider of your Java portlets explicitly.

Note: Web Clipping and OmniPortlet are developing very rapidly. The most recent versions of these portlets are available for download on OTN. If you decide to go with the latest version of these tools, you must deploy them to OC4J and register them with OracleAS Portal as Web providers.

Data-driven portlets, built with Portlet Builder, communicate with OracleAS Portal through database providers. You do not need to register the Portlet Builder providers with OracleAS Portal explicitly; they are automatically registered for you.

PL/SQL portlets communicate with OracleAS Portal through a database provider. You have to register the database provider explicitly.

2.5 Caching Style

Caching plays an essential role in ensuring that your portal is highly performant. OracleAS Portal supports caching on various levels, such as caching pages, portlets, styles, and page metadata. Caching portlets is key to delivering accurate information in a timely manner to your users. All portlet building technologies, available with OracleAS Portal, support caching.

As OracleAS Portal supports user customization of pages and portlets, the view of a page can vary from user to user. OracleAS Portal's caching is designed to allow content to vary on a per-user basis. Therefore, portal objects, including portlets, can be cached at two levels: user level and system level.

- User-level caching is for a specific user; the cache entries stored are unique for that user and cannot be accessed by other users. Good candidates for user-level caching are portlets supporting customization, such as e-mail or stock ticker portlets.
- System-level caching allows users to share a single cache entry and, therefore, there is no need to cache a copy of the object for every user. Examples of content that might be suitable for system-level caching are not customizable news portlets, or custom-built navigation portlets.

When not using caching, accessing various data sources with Web Clipping, OmniPortlet, and Portlet Builder might be time consuming. When you enable caching, you instruct OracleAS Portal or OracleAS Web Cache to maintain a copy of the portlet content. If the portlet is requested and the content was cached previously, the portlet does not have to spend time contacting the data source and regenerating its content again. Simply, the previously cached portlet content is returned.

- **Expiry-based caching**: You can use expiry-based caching when the portlet content is static or when it is not critical that the most up-to-date content be displayed. When using expiry-based caching, you must specify the caching period.
- **Validation-based caching:** Validation-based caching can be used for portlets with dynamic content that changes frequently or unpredictably. The portlet associates its content with a caching key and returns the key value along with the content. When the portlet content is requested, the portlet decides, based on the caching key, if the current content is valid. If the portlet content is valid, then it returns a

response indicating that the cached content can be used (that is, the content is valid) or generates the new portlet content and returns it along with a new caching key for that content.

- **Invalidation-based caching:** Invalidation-based caching is the most complex, but also the most flexible, form of caching. It combines the efficiency of expiry-based caching with the ability to invalidate the cache content any time. Objects in OracleAS Web Cache are considered valid until they are invalidated explicitly.

2.5.1 Web Clipping, OmniPortlet, and Portlet Builder

For portlets built with Web Clipping, OmniPortlet, and Portlet Builder you can specify a period of time for which they are cached (expiry-based caching). In addition to this, portlets built with Web Clipping and OmniPortlet are refreshed automatically when the end user personalizes them

2.5.2 Java Portlets

Java portlets support three types of caching: expiry-, validation-, and invalidation-based caching. With Java portlets, you can combine invalidation-based caching with either expiry-based or validation-based caching.

In addition to caching all your portlet's content, you can also cache fragments of your portlets by using Edge Side Includes (ESI).

2.5.3 PL/SQL Portlets

Similar to Java portlets, PL/SQL portlets also support three types of caching: expiry-, validation-, and invalidation-based caching.

2.6 Development Tool

This section describes the type of development tool you may use to build the portlet. For example, OmniPortlet is built in a browser-based wizard while Java portlets may be built in a tool like Oracle JDeveloper.

2.6.1 Web Clipping, OmniPortlet, and Portlet Builder

OmniPortlet, Web Clipping, and Portlet Builder use a browser-based wizard as the development tool.

2.6.2 Java Portlets

To build Java portlets, the only requirement is the JDK. It is highly recommended, though, that you use Oracle JDeveloper, a professional, integrated development environment (IDE). While you can consider other IDEs, the PDK contains an Oracle JDeveloper plug-in that includes the Java Portlet Wizard, to minimize your Java portlet development efforts.

The Java Portlet Wizard generates a starting skeleton and file structure for both JSR 168 and PDK-Java portlets. You need to only add your own business logic to the skeleton. JDeveloper can also package and deploy your applications to your J2EE container, such as OracleAS Containers for J2EE (OC4J). Also, Oracle JDeveloper helps you test your portlet provider. You can use the integrated standalone OC4J that is shipped with Oracle JDeveloper as your development Java portlet runtime environment, if the version matches that of the platform on which you plan to deploy.

2.6.3 PL/SQL Portlets

When developing a PL/SQL portlet, you create PL/SQL program units that access OracleAS Portal by calling OracleAS Portal PL/SQL APIs. To enable this access, you create a schema, the provider schema, to store the provider and portlet PL/SQL packages in the same database in which OracleAS Portal is installed. The provider schema must be granted execute privileges on the OracleAS Portal PL/SQL APIs.

To facilitate the development of database providers and PL/SQL portlets, you can use the PL/SQL Generator, a hosted utility that creates installable PL/SQL code for a database provider and its PL/SQL portlets. The PL/SQL Generator is a Web application that receives the provider and portlet definitions in the form of an XML file. The syntax of the XML tags that are used for the provider and portlet definition is a subset of the XML tags that are used for defining Web providers with the PDK-Java. The output of the PL/SQL Generator is a SQL script that can be run from SQL*Plus. The script contains SQL commands for installing the provider and portlet packages.

The hosted PL/SQL Generator is available in the Portal Studio Web site at `http://www.oracle.com/technology/products/ias/portal/index.html`.

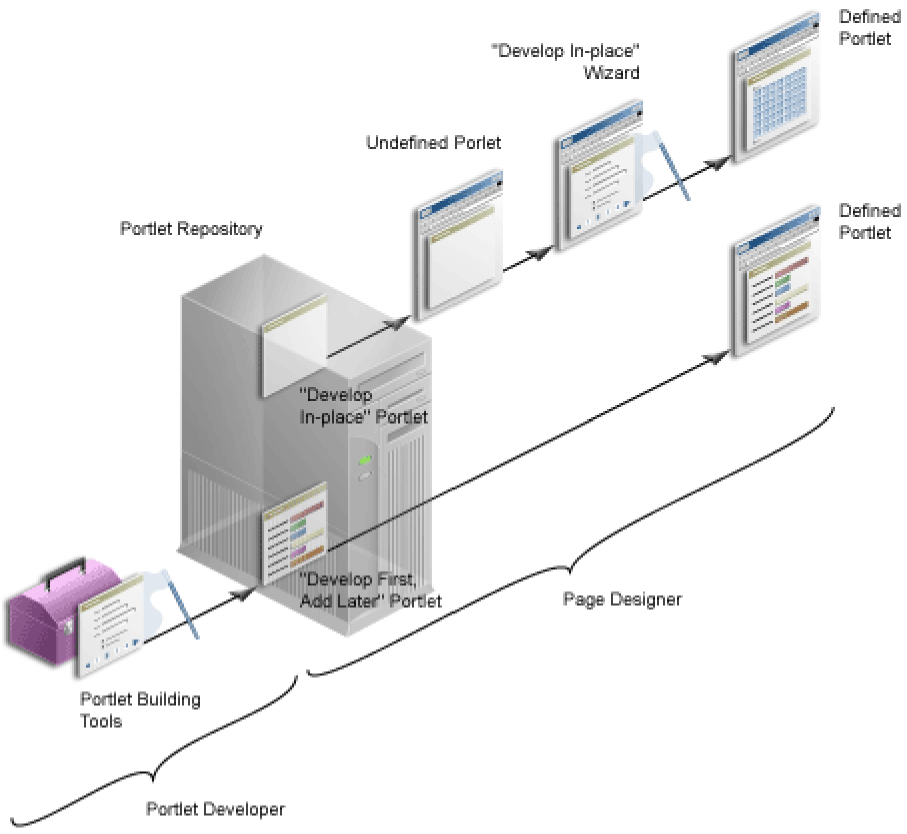
2.7 Portlet Creation Style

OracleAS Portal supports two types of portlet creation as shown in Figure 2–5:

- "Develop in-place"
- "Develop first, add later"

The figure also indicates that the "Develop first, add later" portlet creation is usually the task of the portlet developer, while the "Develop in-place" portlet creation is the page designer's responsibility.

Figure 2–5 Portlet Creation Style



2.7.1 OmniPortlet and Web Clipping

OmniPortlet and Web Clipping offer the same approach to creating portlets. First you add the portlets to a portal page and then you define them in place on the page.

2.7.2 Java Portlets

Java portlets do not tend to provide a develop in place experience. You can easily add edit defaults and personalization to your Java portlets.

Note: With extensive coding, you can create develop in place Java portlets. For example, Web Clipping and OmniPortlet are both Java portlets.

2.7.3 Portlet Builder

With Portlet Builder you define the portlets first. The previously defined portlets are then made available to you in the portlet repository so you can add them to your pages. For simple portlets, though, Portlet Builder offers you the develop in place experience, similar to OmniPortlet and Web Clipping.

Note: Portlets built with Portlet Builder's develop in place technology are somewhat limited as compared to those built using the Navigator.

2.7.4 PL/SQL Portlets

Similar to the Java portlets, PL/SQL portlets typically follow the "Develop first, add later" creation path. Extensive coding is required to develop in place PL/SQL portlets. For example, simple in place portlets that are offered by Porltet Builder are written in PL/SQL.

2.8 User Interface Flexibility

This section describes the portlet building tools in terms of the control you have over the user interface.

2.8.1 Web Clipping

Because of its nature, Web Clipping always displays the remote Web site content, therefore UI flexibility is not a requirement for this portlet.

2.8.2 OmniPortlet and Portlet Builder

While you can be very productive in building portlets with OmniPortlet and Portlet Builder, they are somewhat limiting with respect to the user interface.

OmniPortlet enables you to select from a few layout styles, including tabular, chart, news, bullet, and form. Depending on the layout styles, you can fine-tune the appearance of your portlet. For example, you can opt for alternating versus plain table row background, choose from three types of charts (bar, pie, and line), or specify the bullet style (disc, circle, square, numeric, and so on). You can also specify the column labels, as well as their alignment.

2.8.3 Java Portlets and PL/SQL Portlets

In Java portlets and PL/SQL portlets you have full control over your portlet's user interface. Your portlet is free to generate any HTML content that conforms the rendering rules for OracleAS Portal pages.

2.9 Ability to Capture Content from Web Sites

This section describes the portlet building tools in terms of their ability to include content from other sources.

2.9.1 Web Clipping

In the event that you have to create a portlet that displays the content from a remote Web site as it is presented at the source location, the best tool to use is Web Clipping. Web Clipping can tolerate the changes of the source HTML page to some extent. If a clipped table moves from one place to another in the source page, the Web Clipping engine can find the table again using the internal "fuzzy match" algorithm. Portlets built with Web Clipping can also maintain session to the remote Web site. Web Clipping also supports the end user personalization of HTML form values.

2.9.2 OmniPortlet

Another possible scenario is that you are interested in the data only, not in the way it is presented on the remote Web site. You want to retrieve the data, process the data (format, filter, and so on), and present it in a portlet in a tabular, chart, or news format.

For this purpose, OmniPortlet is the best choice. OmniPortlet is a powerful tool that extracts data from Web pages by using its Web data source.

2.9.3 Java Portlets

In your Java portlets, similarly to other Java applications, you can always take advantage of the low-level Java networking APIs to retrieve and process content from remote Web sites. To avoid unnecessary development efforts, before choosing Java always make sure that Web Clipping or OmniPortlet are not viable options for you.

2.9.4 PL/SQL Portlets

PL/SQL portlets can communicate with Web servers to access data on the Internet by using procedures and functions from the UTL_HTTP package. The package makes HTTP callouts from SQL and PL/SQL. The package also supports HTTP over the Secured Socket Layer protocol (SSL), also known as HTTPS, directly or through an HTTP proxy. Other Internet-related data-access protocols (such as the File Transfer Protocol (FTP) or the Gopher protocol) are also supported using an HTTP proxy server that supports those protocols.

2.10 Ability to Render Content Inline

Active elements in your portlets, such as links or form buttons, enable your users to navigate to remote URLs. In a News portlet for example, you can click a hyperlink to navigate to a news site with detailed information about the news you are interested in. You leave the portal page; the News site replaces it in your browser.

However, you may be required to keep your users within the context of the portal page by rendering the requested content within the same portlet container. You have to display the detailed news on the portal page within the boundaries of the same portlet.

2.10.1 Web Clipping

Web Clipping has URL rewriting support to achieve this functionality: it can process the links, originating from the source Web site, and modify (rewrite) them to achieve the desired functionality.

You can choose from the following three options:

- You can select not to rewrite the links, in which case clicking the link takes the users out of Portal to the Web site providing the clip.
- Depending on whether your clip requires authentication, you can also choose Login Server or Inline. By choosing Login Server you instruct Web Clipping to rewrite the URLs within the portlet so the user is always logged in to the external Web site first before proceeding.
- The Inline option rewrites all URLs within the portlet to point back to the portal page so that all browsing within the Web Clipping portlet remains within Portal.

2.10.2 OmniPortlet

OmniPortlet does not offer URL rewriting directly, but you can achieve the inline rendering functionality by using public portlet parameters and events. Then you have to map the events to the same portal page where your OmniPortlet resides.

2.10.3 Java Portlets

Since you have full control over the links and buttons in Java portlets, you can easily implement the inline rendering functionality. You have to append the private portlet parameters to the page URL.

If you use Struts in your portlet, the PDK-Struts integration framework renders your content always in the same portlet container.

If your portlet consists of multiple JSPs (for example several steps in a survey, or in a Wizard), your portlet can make use of a special parameter to specify at runtime, which JSP should be used to render the content.

2.10.4 Portlet Builder

Portlets built with Portlet Builder do not have inherent inline rendering support. You can, however, construct your links in SQL-based reports and charts so that they point to specific portal pages. If required, you can also pass parameters to portal pages, which in turn can then be mapped to portlet parameters.

2.10.5 PL/SQL Portlets

Similar to Java portlets, you have full control over the active elements in PL/SQL portlets and, therefore, you can achieve the inline rendering functionality programmatically by implementing private portlet parameters.

2.11 Charting Capability

This section describes the portlet building tools in terms of their charting functionality.

2.11.1 Web Clipping

Because of its nature, Web Clipping can retrieve and present HTML content containing charts, but it does not support the creation of charts.

2.11.2 OmniPortlet

OmniPortlet supports the following three chart types: bar, line, and pie. Charts in OmniPortlet are dynamically generated images, optionally, with event-enabled hyperlinks included.

2.11.3 Java Portlets

You can create more sophisticated chart portlets programmatically in your Java portlets using Oracle's Business Intelligence (BI) Beans.

Note: Oracle Reports and Oracle Discoverer portlets use BI Beans to create professional graphs.

2.11.4 Portlet Builder

With Portlet Builder, you can build HTML-based bar chart portlets. Among other features, you can specify the color and orientation of the bars.

2.11.5 PL/SQL Portlets

In PL/SQL portlets, HTML-based charting can be achieved by extensive coding.

2.12 Public Portlet Parameters Support

There are three types of parameters in OracleAS Portal: page parameters, public portlet parameters, and private portlet parameters.

- **Page parameters:** A page parameter is used to pass a value to a page. Using page parameters, the information that is displayed on a page can vary depending on where the page is called from and who is viewing the page. Using page parameters, you can synchronize the portlets on your page by passing them the same values. This gives you the ability to reuse and tailor portlets on pages by merely integrating them with page parameters. Without this functionality, you would have to code portlets individually to use different parameter values.
- **Public portlet parameters:** A public portlet parameter is used to pass a value to a portlet. Using portlet parameters, the information that is displayed in a portlet can be specific to a particular page or a user. Portlet parameters are created by the portlet developer and are exposed to the page designer, through the user interface. After adding a portlet to a page, page designers can assign values to the public portlet parameters to make the information displayed in the portlet specific to the page.

  Page designers can assign values to public portlet parameters by providing a specific value (constant), a system variable (for example, the portal user name), or a page parameter. At run time, the portlet receives the values from the sources that you specified. In this way, page designers have complete control over the source of the parameter, whereas you have complete control over how the data is used after it is transmitted to the portlet.
- **Private portlet parameters:** You can use private portlet parameters to implement internal navigation in your portlet. You can pass parameters to your portlets every time the page is requested. Private portlet parameters can be passed exclusively from the portlet instance to the same portlet instance.

Portlets supporting public portlet parameters enable page designers to tailor the portlets' data input for each portlet instance. In this case, the portlet developer can focus on the portlet logic, while page designers can easily reuse portlets and address the interaction between the page and the portlets.

All five portlet building technologies discussed in this chapter (OmniPortlet, Web Clipping, Java portlets, Portlet Builder, and PL/SQL portlets) support public portlet parameters. OmniPortlet, Web Clipping, and Portlet Builder provide complete support through their wizard interface. You can add public portlet parameter support to your Java portlets programmatically or with the Java Portlet Wizard. PL/SQL portlets support public parameters only programmatically.

Note: The JSR 168 standard does not cover the notion of public portlet parameters. If you want to utilize public portlet parameters in your Java portlets, you have to use PDK-Java.

2.13 Private Portlet Parameter Support

This section describes the portlet building tools in terms of their support for private parameters.

2.13.1 OmniPortlet, Web Clipping, and Portlet Builder

OmniPortlet, Web Clipping, and Portlet Builder do not provide access to the portlet developer to private portlet parameters.

2.13.2 Java Portlets and PL/SQL Portlets

In your Java portlets and PL/SQL portlets, you can implement internal navigation by using private portlet parameters.

Note: PL/SQL portlets do not support private and public parameters simultaneously. You need to decide which parameter type to support before coding your PL/SQL portlet.

2.14 Event Support

An event is a user action that is defined by the portlet developer to invoke a Portal page. User actions include clicking a link or a button in a portlet. Page designers specify what to do when an event occurs in a portlet on your page. When an event occurs, page designers can either redisplay the current page or navigate the user to another portal page, optionally passing values to that page's parameters.

2.14.1 Web Clipping, OmniPortlet, and Java Portlets

Web Clipping, OmniPortlet, and Java portlets support events.

2.14.2 Portlet Builder and PL/SQL Portlets

Portlet Builder and PL/SQL portlets do not support events.

2.15 Ability to Hide and Show Portlets Based on User Privileges

This section describes the portlet building tools in terms of their support for authorization functionality.

2.15.1 Web Clipping and OmniPortlet

You can hide and show portlets built with Web Clipping and OmniPortlet on portal pages dynamically by using security managers. Although Web Clipping and OmniPortlet do not expose security managers through the user interface, you can apply them by editing their XML provider definition file.

2.15.2 Java Portlets

The PDK provides a number of security managers for Java portlets. For example:

- **Group security manager:** The group security manager makes the portlet appear to users who are members of a specified group, while hiding it from those who are not members.
- **Authentication level security manager:** You can use the authentication level security manager to control access to the portlets based on the user's authentication level. For example you may hide the portlet from public users but display it to authenticated users.

JSR 168 portlets support the standard servlet mechanisms.

2.15.3 Portlet Builder

Portlet Builder provides a declarative user interface to control access to portlets.

2.15.4 PL/SQL Portlets

The PDK provides the Security APIs to implement hiding and showing content in PL/SQL portlets.

2.16 Multi-lingual Support

This section describes the portlet building tools in terms of their support for other languages.

2.16.1 Web Clipping, OmniPortlet, Java Portlets, and PL/SQL Portlets

Web Clipping, OmniPortlet, Java portlets, and PL/SQL portlets display textual information in the language selected by the portal user.

2.16.2 Portlet Builder

Portlets built with Portlet Builder support English only.

2.17 Pagination Support

Support for pagination is useful when you are required to display a relatively large set of records in your portlets.

2.17.1 OmniPortlet

OmniPortlet does not support pagination.

2.17.2 Java Portlets and PL/SQL Portlets

You can implement pagination in your Java portlets and PL/SQL portlets programmatically.

2.17.3 Portlet Builder

Portlet Builder has built-in support for pagination.

2.18 Single Sign-On and External Application Integration

This section describes the portlet building tools in terms of authentication for external application.

2.18.1 Web Clipping

Web Clipping's integration with the external application framework provides a fully automated mechanism to store passwords to external Web sites. All you have to do is to associate an External Application ID to the Web Clipping Provider during provider registration time.

2.18.2 OmniPortlet

OmniPortlet enables you to store connection information when the data source is password protected. The credentials to access the data source can either be shared across all users, or saved on a per user basis. OmniPortlet is capable of storing database credentials, as well as HTTP basic authentication user name-password pairs. The credentials are stored in the secured data repository of OmniPortlet, in an Oracle database.

2.18.3 Java Portlets

Java portlets can integrate with the external application framework as well as any LDAP server, such as Oracle Internet Directory (OID), programmatically.

2.18.4 PL/SQL Portlets

You can build PL/SQL portlets that enable single sign-on by using mod_osso, an authentication module on the Oracle HTTP Server. mod_osso is a simple alternative to the single sign-on SDK, used in earlier releases to integrate partner applications. mod_osso simplifies the authentication process by serving as the sole partner application to the Single Sign-On server.

PL/SQL portlets can integrate with the external application framework programmatically.

Part III

Building Portlets

Part III contains the following chapters:

- Chapter 3, "Building Portlets with OmniPortlet"
- Chapter 4, "Building Content-Based Portlets with Web Clipping"
- Chapter 5, "Building Java Portlets"
- Chapter 6, "Building PL/SQL Portlets"

3 Building Portlets with OmniPortlet

OmniPortlet is a subcomponent of Oracle Application Server Portal that enables page designers and content contributors to easily publish data from different data sources using a variety of layouts. You can learn more about other portlet development tools in Chapter 2, "Portlet Technologies Matrix".

The OmniPortlet enables page designers and content contributors to:

- Display data from multiple sources (CSV, XML, SQL, and so on)
- Sort the data to display
- Format data using a variety of layouts (bulleted list, chart, form, and so on)
- Use portlet parameters
- Raise portlet events
- Expose customizable settings to page viewers

OmniPortlet also supports proxy authentication, including support for global proxy authentication and per-user authentication. You can specify whether all users will automatically log in using a user name and password you provide, each user will log in using an individual user name and password, or all users will log in using a specified user name and password.

The OmniPortlet provider is installed as part of OracleAS Portal. You can upgrade to later releases of OmniPortlet from the product area on OTN (`http://www.oracle.com/technology/products/ias/portal/index.html`) as part of the Oracle Application Server Portal Developer Kit (PDK). Instructions for installing, configuring, and registering the OmniPortlet provider are provided within the `pdksoftware.zip` file containing the JPDK and Portal Tools.

Note: You can find more information about developing portlets in Chapter 1, "Understanding Portlets", and information about providers and other portlet technologies in Chapter 2, "Portlet Technologies Matrix".

This chapter provides an overview of the OmniPortlet, then steps you through defining various types of OmniPortlets. It contains the following sections:

- What is OmniPortlet?
- Parameters and Events
- Using OmniPortlet

3.1 What is OmniPortlet?

OmniPortlet is specifically targeted at enabling page designers and developers to quickly and easily publish data from various data sources in a variety of layouts using a Web-based wizard. An OmniPortlet can be based on almost any kind of data source, such as a spreadsheet (comma-separated values), XML, and even application data from an existing Web page.

To display personalized data, you can refine the retrieved data by filtering the results returned from a data source, and parameterize the credential information used to access secure data. Out of the box, OmniPortlet provides the most common layout for portlets: tabular, chart, news, bulleted list, and form.

OmniPortlet is provided with Oracle Application Server 10*g*; you can add an OmniPortlet from the OracleAS Portal Repository in the Portlet Builders folder. If you've downloaded OmniPortlet as part of the Oracle Application Server Portal Developer Kit, you must register it before you can use it. To learn more about registering a Web provider, refer to the documentation on Portal Center. You can then add an OmniPortlet to any portal page.

Note: You can find more information about building portal pages in the *Oracle Application Server Portal User's Guide* on Portal Center.

The OmniPortlet wizard contains six tabs. When you first define your OmniPortlet, you set the data source type, data source options, filter options, view options, and layout. When you've completed these steps of the wizard, you can re-enter the wizard by clicking the Edit Defaults icon on the Portal page. When you re-enter the wizard, you can change the definitions on the Source, Filter, View, and Layout tabs, as well as set up the events parameters on the Events tab.

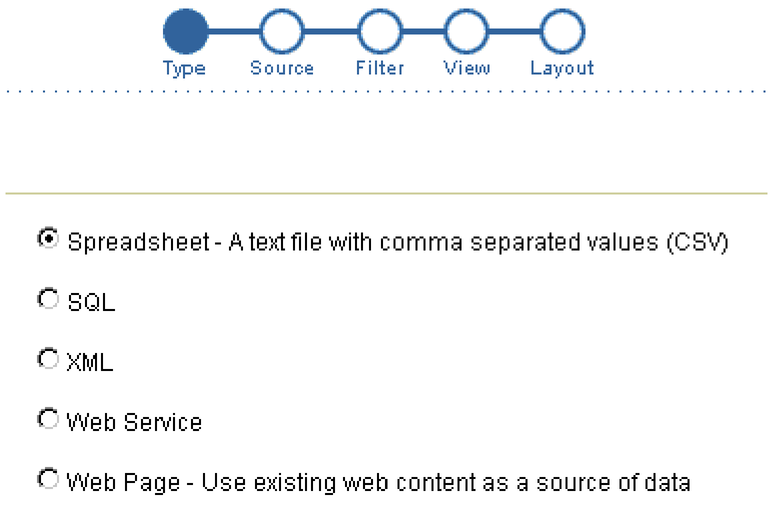
This section provides a high-level overview of the six tabs (see the table below), as well as a detailed description of each tab. You can also find information in the online help (accessible by clicking the Help link in the product), which describes the options on each tab.

Table 3–1 Omniportlet Wizard

Tab	Description
Type	Provides your data source options. Displays only in the initial definition of the portlet.
Source	Provides the options for the selected data source, such as the URL of the Web Service you wish to use.
Filter	Provides sorting options at the OracleAS Portal level to enable you to refine your results.
View	Provides options for displaying portlet header and footer text, the layout style, and caching.
Layout	Provides detailed options for customizing the layout.
Events	Provides options for adding events to the portlet. Displays only after the portlet has been defined in the Edit Defaults mode of the wizard.

3.1.1 Type

Figure 3–1 Type Tab of the OmniPortlet Wizard



[✓ Spreadsheet - A text file with comma separated values (CSV)]

SQL

XML

Web Service

Web Page - Use existing web content as a source of data

Note: If you've downloaded and installed an additional data source from Portal Center (`http://www.oracle.com/technology/products/ias/portal/index.html`), the data source will display on the Type tab.

When you first launch the OmniPortlet, the Type tab displays, which enables you to choose your data source. Out of the box, OmniPortlet supports the following data sources:

Table 3–2 Supported Data Source Types

Data Source Type	Description
Spreadsheet	Displays data from a text file containing comma-separated values (CSV).
SQL	Displays data from a database using SQL.
XML	Displays data from an XML file.
Web Service	Displays data from a discrete business service that can be accessed over the Internet using standard protocols.
Web Page	Displays data based on existing Web content.

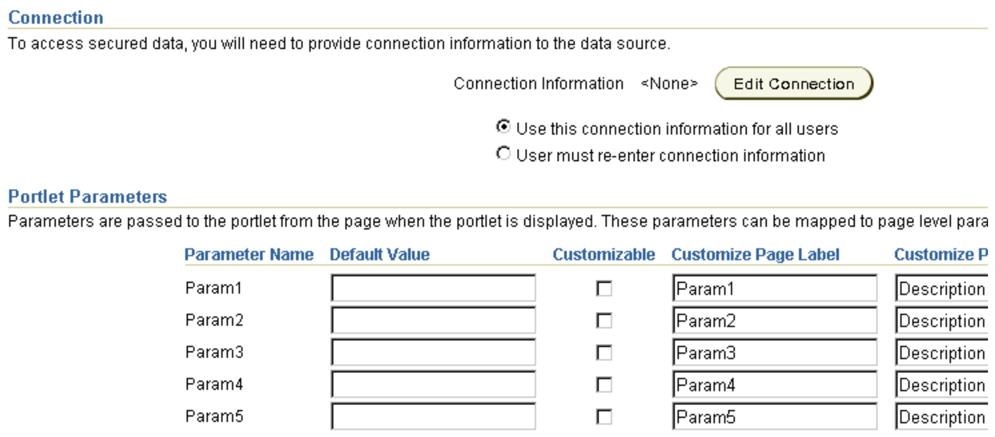
3.1.2 Source

After you've chosen your data source type, the Source tab of the OmniPortlet wizard displays. This tab adapts to the data source you've chosen, to enable you to specify the options offered by that data source. The top portion of the screen is dedicated to the specified source. If the OmniPortlet provider has been set up to use proxy authentication that requires your login, a Proxy Authentication section displays where you can enter this information.

Note: The Proxy Authentication section only displays for the data sources that may require you to use a proxy server to access them: CSV (comma-separate values), XML, Web Service, and Web Page. For more information on configuring the OmniPortlet provider to use proxy authentication, see the online help topic that displays when you click Help on the Edit Providers: OmniPortlet Provider page.

Each Source tab, except for the Source tab for the Web Page data source, contains a Connection section, where you can define the connection information to access secured data, and a Portlet Parameters section, where you can define the parameters for the portlet. You can then map the portlet parameters to the page-level parameters.

Figure 3–2 Source Tab: Connection and Portlet Parameters Section


Connection

To access secured data, you will need to provide connection information to the data source.

Connection Information <None>

[✓ Use this connection information for all users]

User must re-enter connection information

Portlet Parameters

Parameters are passed to the portlet from the page when the portlet is displayed. These parameters can be mapped to page level para

Parameter Name	Default Value	Customizable	Customize Page Label	Customize P
Param1		☐	Param1	Description
Param2		☐	Param2	Description
Param3		☐	Param3	Description
Param4		☐	Param4	Description
Param5		☐	Param5	Description

The following sections describe the portion of the Source tab specific to each data source.

- Spreadsheet
- SQL
- XML
- Web Service
- Web Page

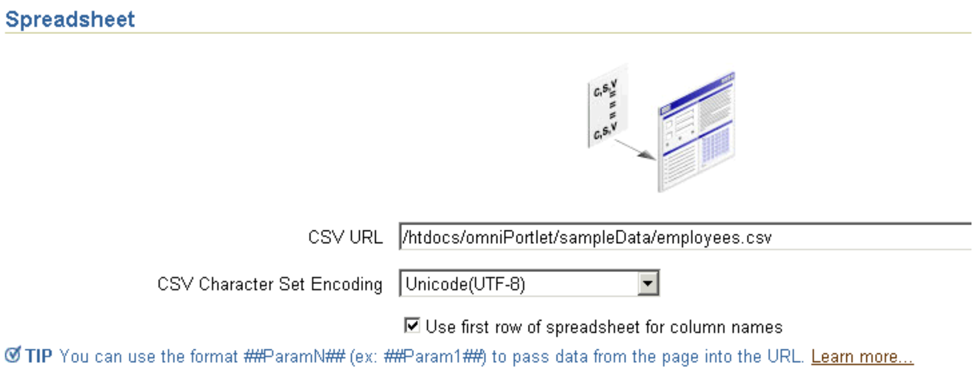
Note: For more information on the Source tab options, click Help in the upper right corner of the page.

3.1.2.1 Spreadsheet

Spreadsheets are a common method of storing small data sets. OmniPortlet enables you to share spreadsheets by supporting comma-separated values (CSV) as a data source. On the Source tab, you specify the location of the CSV file. If the file is located on a secure server, you can specify the connection information in the Connection section described above. You can also select the character set to use when OracleAS Portal reads the file.

Note: Since the OmniPortlet provider exists and executes in a tier different than the OracleAS Portal application and does not have access to the OracleAS Portal session information, you must expose CSV files that are uploaded to OracleAS Portal as PUBLIC in order for OmniPortlet to access them.

Figure 3–3 Source Tab: Spreadsheet



CSV URL: /htdocs/omniPortlet/sampleData/employees.csv

CSV Character Set Encoding: Unicode(UTF-8)

☑ Use first row of spreadsheet for column names

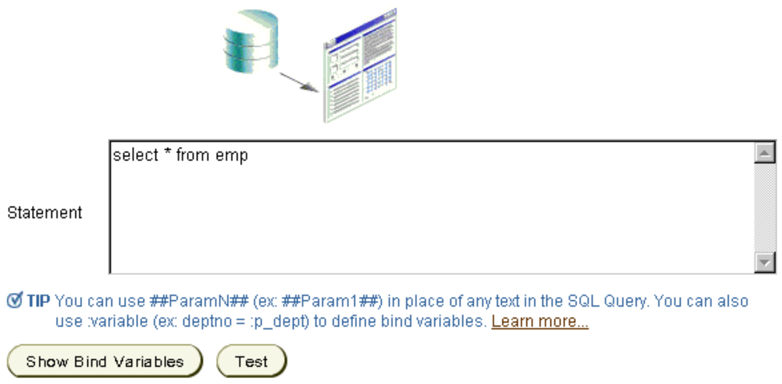
TIP You can use the format ##ParamN## (ex: ##Param1##) to pass data from the page into the URL. Learn more...

3.1.2.2 SQL

The relational database is the most common place to store data. OmniPortlet enables you to use standard JDBC drivers and provides out-of-the-box access to Oracle and any JDBC database. You can specify the driver type when you configure the connection information.

Note: You can also use the DataDirect JDBC drivers to access other relational databases. To configure OmniPortlet to use these drivers, refer to "OmniPortlet: How to Use DataDirect JDBC Drivers" under Developer Services > OmniPortlet & Web Clipping on Portal Center (`http://www.oracle.com/technology/products/ias/portal/index.html`).

Figure 3–4 Source Tab: SQL



Statement:
```
select * from emp
```

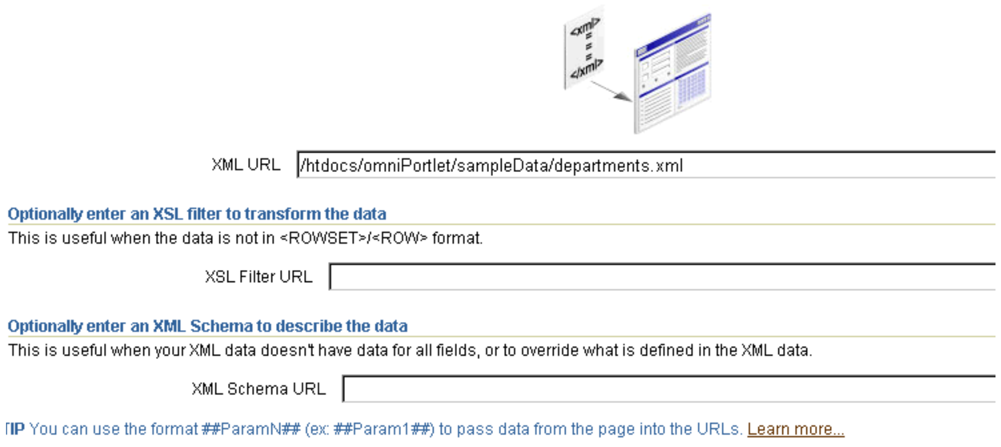
TIP You can use ##ParamN## (ex: ##Param1##) in place of any text in the SQL Query. You can also use :variable (ex: deptno = :p_dept) to define bind variables. Learn more...

Note: For more information on using the SQL data source, refer to Defining a Portlet Based on a SQL Data Source.

3.1.2.3 XML

Although still a relatively new method of storing data, XML is increasingly used to control access to data sets over the intranet, and even the Internet. On the Source tab, you can specify the URL of the XML file that contains your data.

Figure 3–5 Source Tab: XML


XML URL: /htdocs/omniPortlet/sampleData/departments.xml

Optionally enter an XSL filter to transform the data

This is useful when the data is not in <ROWSET>/<ROW> format.

XSL Filter URL:

Optionally enter an XML Schema to describe the data

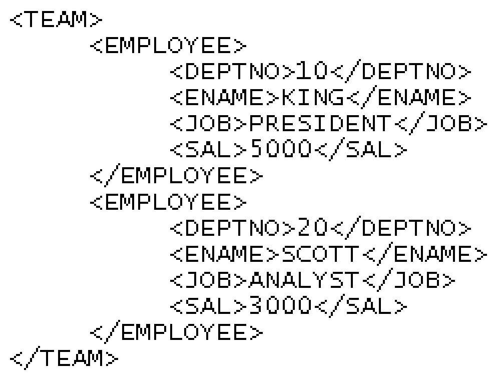
This is useful when your XML data doesn't have data for all fields, or to override what is defined in the XML data.

XML Schema URL:

TIP You can use the format ##ParamN## (ex: ##Param1##) to pass data from the page into the URLs. Learn more...

The specified XML file can either be in tabular (ROWSET/ROW) structure, or you can provide an XML Style Sheet (XSL) to transform the data. The following image shows an example of the ROWSET/ROW structure of an XML data source.

Figure 3–6 ROWSET/ROW Structure of an XML Data Source


```
<TEAM>
     <EMPLOYEE>
          <DEPTNO>10</DEPTNO>
          <ENAME>KING</ENAME>
          <JOB>PRESIDENT</JOB>
          <SAL>5000</SAL>
     </EMPLOYEE>
     <EMPLOYEE>
          <DEPTNO>20</DEPTNO>
          <ENAME>SCOTT</ENAME>
          <JOB>ANALYST</JOB>
          <SAL>3000</SAL>
     </EMPLOYEE>
</TEAM>
```

In the above example, the <TEAM> tags delineate the rowset, and the <EMPLOYEE> tags delineate the rows.

Regardless of the format of the XML file, OmniPortlet automatically inspects the XML to determine the column names, which will then be used to define the layout. If you want to specify this information yourself, you can supply a URL to an XML schema that describes the data.

Similar to the other data sources, you can also specify the connection information for this data source, if the XML file is located on a secured server protected by HTTP Basic Authentication.

Note: Since the OmniPortlet provider exists and executes in a different tier to OracleAS Portal and does not have access to the OracleAS Portal session information, you must expose XML files that are uploaded to OracleAS Portal as PUBLIC in order for OmniPortlet to access them.

3.1.2.4 Web Service

A Web Service is a discrete business service that can be programmatically accessed over the Internet using standard protocols, such as SOAP and HTTP. Web Services are non-platform and non-language specific, and are typically registered with a Web Service broker. When you find a Web Service you wish to use, you must obtain the URL to the WSDL (Web Service Description Language) file that describes the Web Service, specifies the methods that can be called, the expected parameters, and a description of the returned data.

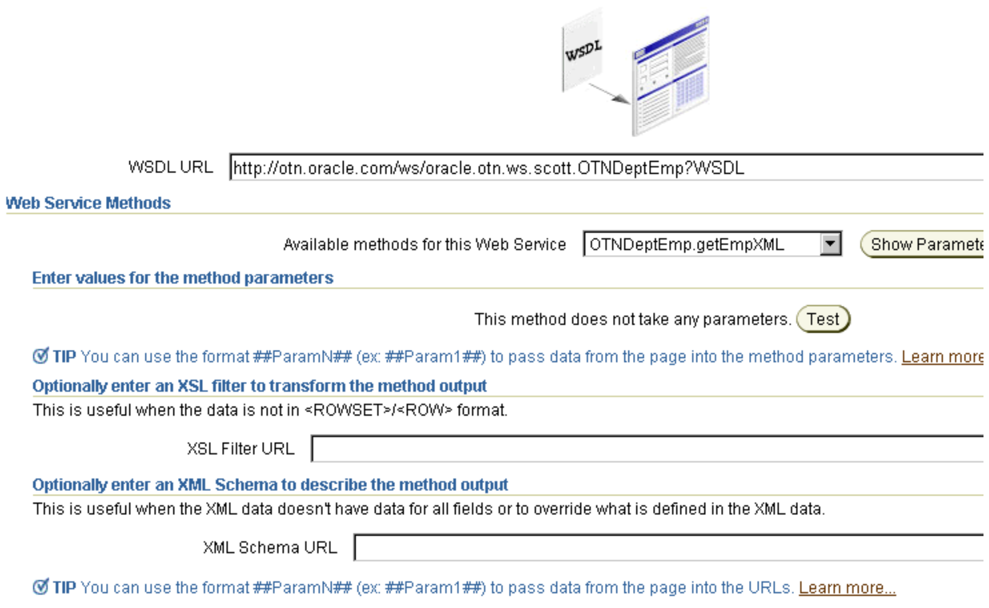
OmniPortlet supports both types of Web Services (Document and RPC (Remote Procedure Calls)). Once a WSDL is supplied, it is parsed, and the available methods that can be called display on the Source tab.

Similar to the XML data source, OmniPortlet expects the Web Service data in ROWSET/ROW format, though you can also use an XSL file to transform the data. OmniPortlet inspects the WSDL to determine the column names, though you may also specify an XML schema to describe the returned data set.

If you are new to Web services, you may want to first review the New to Web Services guide on the Oracle Web Services Center (`http://www.oracle.com/technology/tech/webservices/learner.html`).

Note: For more information on using the Web Service data source, refer to Defining a Portlet Based on a Web Service.

Figure 3–7 Source Tab: Web Service



WSDL URL http://otn.oracle.com/ws/oracle.otn.ws.scott.OTNDeptEmp?WSDL

Web Service Methods

Available methods for this Web Service OTNDeptEmp.getEmpXML Show Paramete

Enter values for the method parameters

This method does not take any parameters. Test

TIP You can use the format ##ParamN## (ex: ##Param1##) to pass data from the page into the method parameters. Learn more

Optionally enter an XSL filter to transform the method output

This is useful when the data is not in <ROWSET>/<ROW> format.

XSL Filter URL

Optionally enter an XML Schema to describe the method output

This is useful when the XML data doesn't have data for all fields or to override what is defined in the XML data.

XML Schema URL

TIP You can use the format ##ParamN## (ex: ##Param1##) to pass data from the page into the URLs. Learn more...

3.1.2.5 Web Page

OmniPortlet enables you to use existing Web content as a source of data to publish information to your portal. It provides and renders clipped Web content as a data source.

The Web Page data source extends the scope offered by the Web Clipping Portlet to include scraping functionality. It also supports the following features:

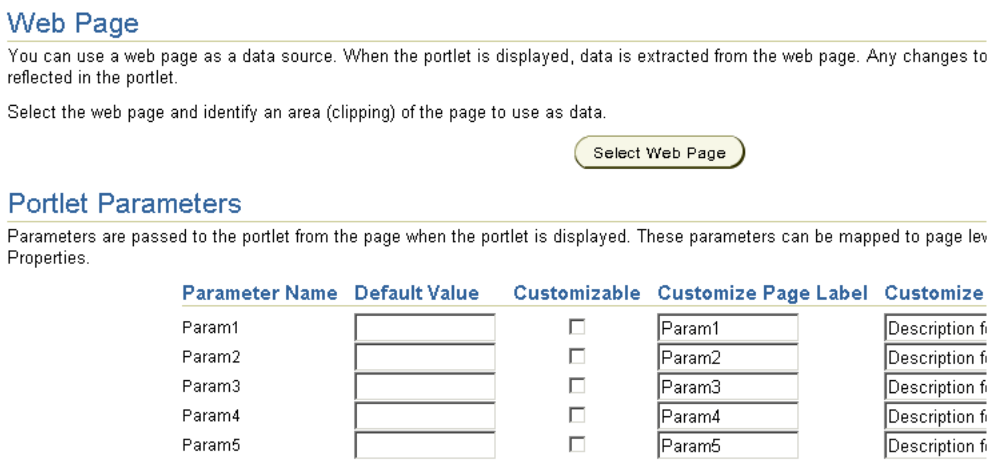
- **Navigation through various login mechanisms**, including form- and JavaScript-based submission, and HTTP Basic and Digest Authentication with cookie-based session management.
- **Fuzzy matching of clippings.** If a Web clipping gets reordered within the source page or if its character font, size, or style changes, it will still be identified correctly by the Web page data source and delivered as the portlet content.
- **Reuse of a wide range of Web content**, including basic support of pages written with HTML 4.0.1 and JavaScript, retrieved through HTTP GET and POST (form submission).

All Web clipping definitions are stored persistently in the Oracle Application Server infrastructure database or on another Oracle database. Any secure information, such as passwords, is stored in encrypted form, according to the DES (Data Encryption Standard), using Oracle9*i* database encryption technology.

The Source tab of the OmniPortlet wizard enables you to launch the Web Clipping Studio by clicking the Select Web Page button. Once you launch the Web Clipping Studio, you can refer to the Oracle Application Server Web Clipping Online Help.

Note: For more information on using the Web Page data source, refer to Defining a Portlet Based on an Existing Web Page.

Figure 3–8 Source Tab: Web Page



Web Page

You can use a web page as a data source. When the portlet is displayed, data is extracted from the web page. Any changes to
reflected in the portlet.

Select the web page and identify an area (clipping) of the page to use as data.

Select Web Page

Portlet Parameters

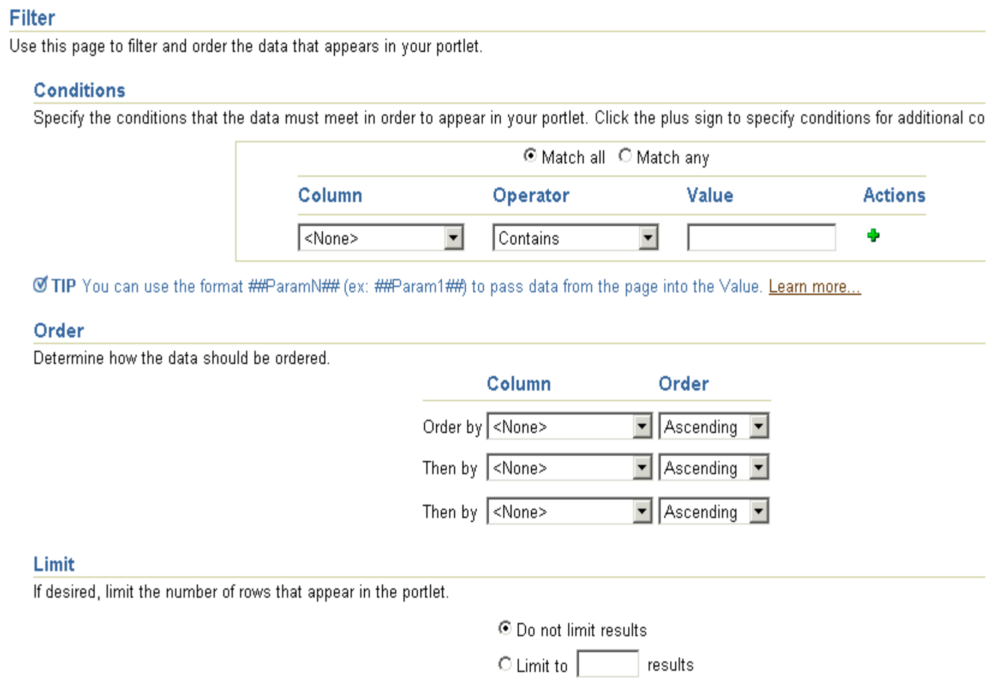
Parameters are passed to the portlet from the page when the portlet is displayed. These parameters can be mapped to page lev
Properties.

Parameter Name	Default Value	Customizable	Customize Page Label	Customize
Param1		☐	Param1	Description fi
Param2		☐	Param2	Description fi
Param3		☐	Param3	Description fi
Param4		☐	Param4	Description fi
Param5		☐	Param5	Description fi

3.1.3 Filter

After you've selected the data source and specified the data source options, you can further refine your data by using OmniPortlet's filtering options. To use filtering efficiently, it is better to refine the data as much as possible at the data source level (that is, on the Source tab), then use the options on the Filter tab to streamline the data. For example, if you are using a SQL data source, you could use a WHERE clause to return only specific data from the specified columns. In this case, you could skip the Filter tab and continue to the View page of the wizard. However, if there are no filtering options at the data source level, you can use the options on the Filter tab to sort your data.

Figure 3–9 Filter Tab



Filter

Use this page to filter and order the data that appears in your portlet.

Conditions

Specify the conditions that the data must meet in order to appear in your portlet. Click the plus sign to specify conditions for additional co

[✓ Match all] Match any

Column	Operator	Value	Actions
<None>	Contains		+

TIP You can use the format ##ParamN## (ex: ##Param1##) to pass data from the page into the Value. Learn more...

Order

Determine how the data should be ordered.

	Column	Order
Order by	<None>	Ascending
Then by	<None>	Ascending
Then by	<None>	Ascending

Limit

If desired, limit the number of rows that appear in the portlet.

[✓ Do not limit results]

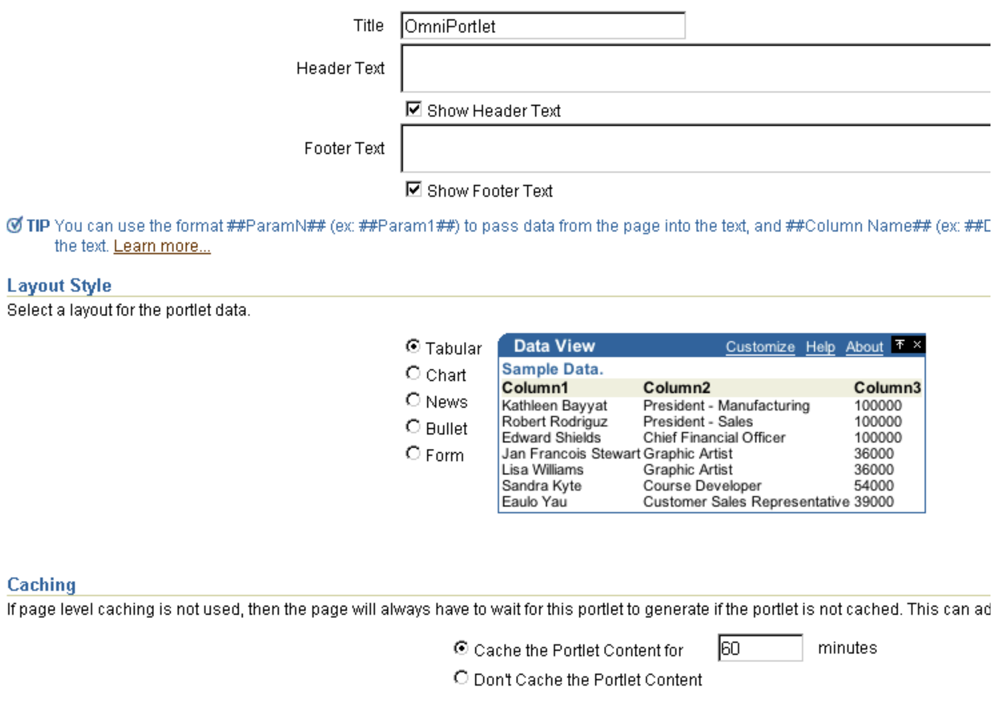
Limit to results

3.1.4 View

Once you've specified the data and sorted it, you can choose the view options and layout for your OmniPortlet. The View tab enables you to add Header and Footer text, choose a Layout style that you can later refine on the Layout tab, and enable caching. You can choose from the following layouts:

- tabular
- chart
- news
- bullet
- form

Figure 3–10 View Tab



Title: OmniPortlet

Header Text:

[✓] Show Header Text

Footer Text:

[✓] Show Footer Text

TIP You can use the format ##ParamN## (ex: ##Param1##) to pass data from the page into the text, and ##Column Name## (ex: ##D
the text. Learn more...

Layout Style

Select a layout for the portlet data.

[✓ Tabular] Chart News Bullet Form

Data View Customize Help About

Sample Data.

Column1	Column2	Column3
Kathleen Bayyat	President - Manufacturing	100000
Robert Rodriguz	President - Sales	100000
Edward Shields	Chief Financial Officer	100000
Jan Francois Stewart	Graphic Artist	36000
Lisa Williams	Graphic Artist	36000
Sandra Kyte	Course Developer	54000
Eaulo Yau	Customer Sales Representative	39000

Caching

If page level caching is not used, then the page will always have to wait for this portlet to generate if the portlet is not cached. This can ad

[✓ Cache the Portlet Content for] 60 minutes

Don't Cache the Portlet Content

Note: For more information on the different layout styles you can use with OmniPortlet, click Help in the upper right corner of the page.

3.1.5 Layout

The Layout tab changes depending on the Layout Style you chose on the View tab, and enables you to further customize the appearance of your portlet. OmniPortlet supports drill-down hyperlinks in the chart layout. That is, you can set up the chart so that when a user clicks on a specific part of the chart, an action occurs (for example, jump to another URL).

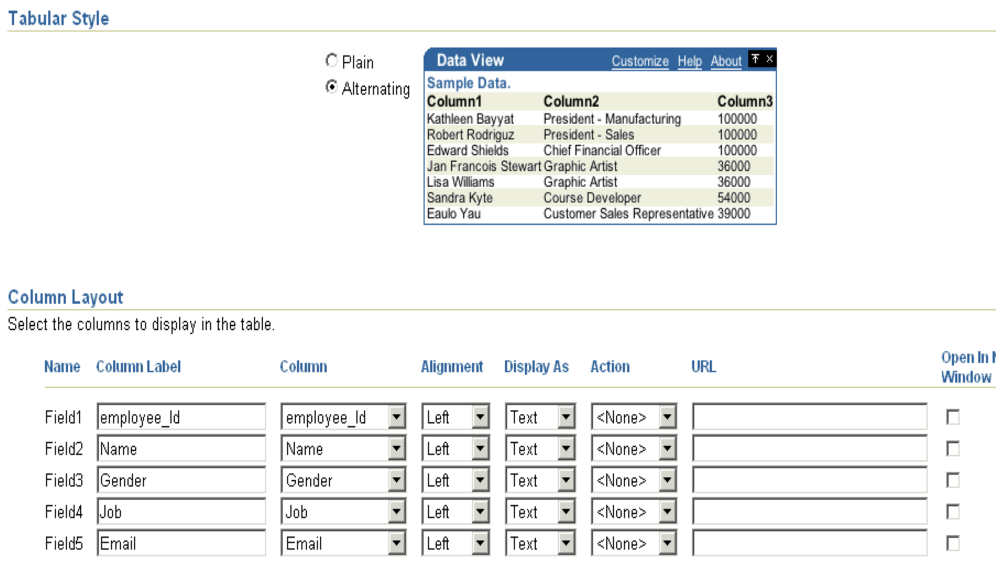
For the other layout styles, you can define each column to display in a specific format, such as plain text, HTML, an image, button, or field. For example, suppose you selected a data source that includes a URL to an image. To see this image, you can select Image for the display of this column. Each column can also be mapped to an action, similar to the behavior of chart hyperlinks.

The following layout styles are available with OmniPortlet:

- Tabular Layout
- Chart Layout
- News Layout
- Bullet Layout
- Form Layout

3.1.5.1 Tabular Layout

Figure 3–11 Layout Tab: Tabular Style



Tabular Style

( ) Plain
[✓ Alternating]

Data View — Customize Help About

Sample Data.

Column1	Column2	Column3
Kathleen Bayyat	President - Manufacturing	100000
Robert Rodriguz	President - Sales	100000
Edward Shields	Chief Financial Officer	100000
Jan Francois Stewart	Graphic Artist	36000
Lisa Williams	Graphic Artist	36000
Sandra Kyte	Course Developer	54000
Eaulo Yau	Customer Sales Representative	39000

Column Layout

Select the columns to display in the table.

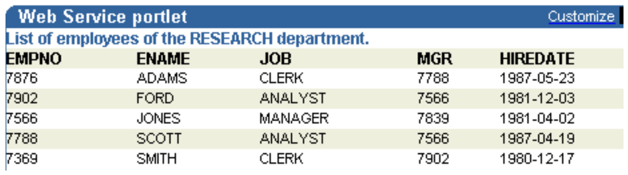
Name	Column Label	Column	Alignment	Display As	Action	URL	Open In Window
Field1	employee_Id	employee_Id	Left	Text	<None>		☐
Field2	Name	Name	Left	Text	<None>		☐
Field3	Gender	Gender	Left	Text	<None>		☐
Field4	Job	Job	Left	Text	<None>		☐
Field5	Email	Email	Left	Text	<None>		☐

Once you've chosen the tabular style on the View tab, you can refine the layout on the Layout tab. Typically, you use the tabular layout if you have one or more columns of data that you want to display in a table. You can choose Plain to display all rows in the table without any background color, or Alternating to display a background color for every other row in the table.

> **Note:** You can control the background color of a portlet through the portal page style. For more information on using portal page styles, refer to the *Oracle Application Server Portal User's Guide*.

In the Column Layout section, you can choose which data columns to display in the portlet, then select a display format for the data. Here, you can set a column to display a hyperlink, so that a secondary Web page displays when the user clicks that column in the table. You can also specify whether the secondary Web page displays in a new window.

Figure 3–12 Example of an OmniPortlet Using a Tabular Layout



Web Service portlet | Customize

List of employees of the RESEARCH department.

EMPNO	ENAME	JOB	MGR	HIREDATE
7876	ADAMS	CLERK	7788	1987-05-23
7902	FORD	ANALYST	7566	1981-12-03
7566	JONES	MANAGER	7839	1981-04-02
7788	SCOTT	ANALYST	7566	1987-04-19
7369	SMITH	CLERK	7902	1980-12-17

Note: For more information on using the OmniPortlet wizard, click the Help link in the upper right corner of the Layout tab.

3.1.5.2 Chart Layout

Figure 3–13 Layout Tab: Chart

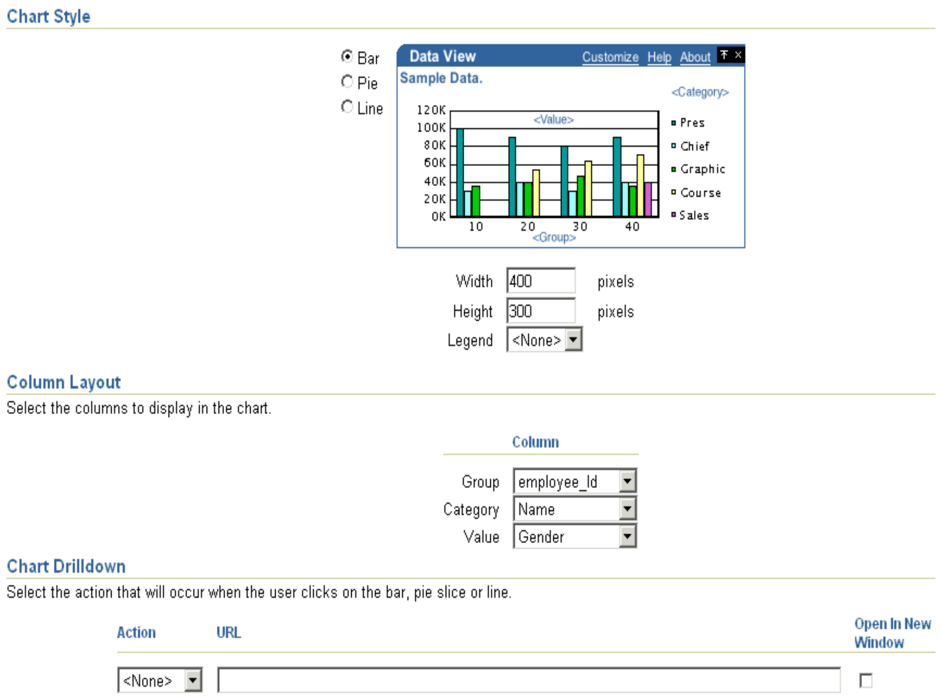


Chart Style

[✓ Bar] Pie Line

Width 400 pixels

Height 300 pixels

Legend <None>

Column Layout

Select the columns to display in the chart.

	Column
Group	employee_Id
Category	Name
Value	Gender

Chart Drilldown

Select the action that will occur when the user clicks on the bar, pie slice or line.

Action	URL	Open In New Window
<None>		☐

You can use the chart layout to display your data graphically, as a bar, pie, or line chart. On the Layout tab, you select the chart style and the column layout. When you choose the column layout, you can choose the groups, or columns on which the labels will be based. The category defines the values that will be used to create the chart legend, and the value determines the relative size of the bars, lines, or slices in the chart. You can also select whether the sections of the chart should point to a hyperlink, and whether the targeted information should display in a new window.

Note: To group the information in the chart, you must group the information at the data level (for example, in your SQL query statement). Also, if numeric values in a data source contain formatted strings, commas, or currency (for example, $32,789.00), they are considered to be text and ignored when the chart is generated. You can instead remove these formatting characters from numerical values so that they are correctly read.

Figure 3–14 Example of the Layout Tab for a Pie Chart Layout

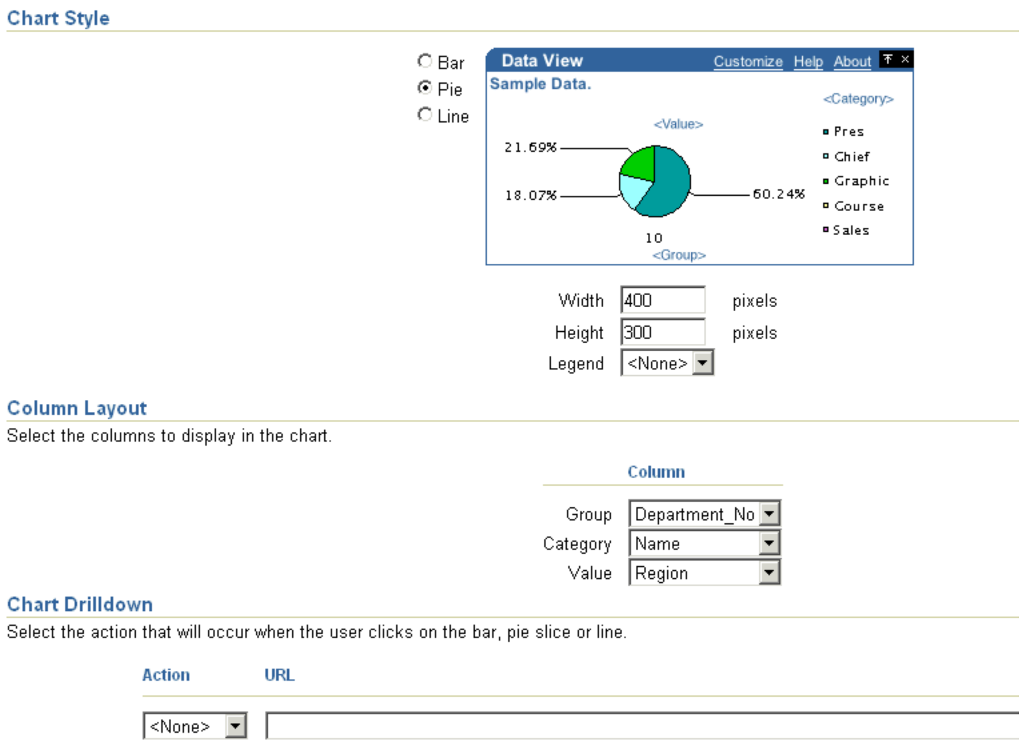


Chart Style

Bar [✓ Pie] Line

Data View Customize Help About

Sample Data.

Width	400	pixels
Height	300	pixels
Legend	<None>	

Column Layout

Select the columns to display in the chart.

	Column
Group	Department_No
Category	Name
Value	Region

Chart Drilldown

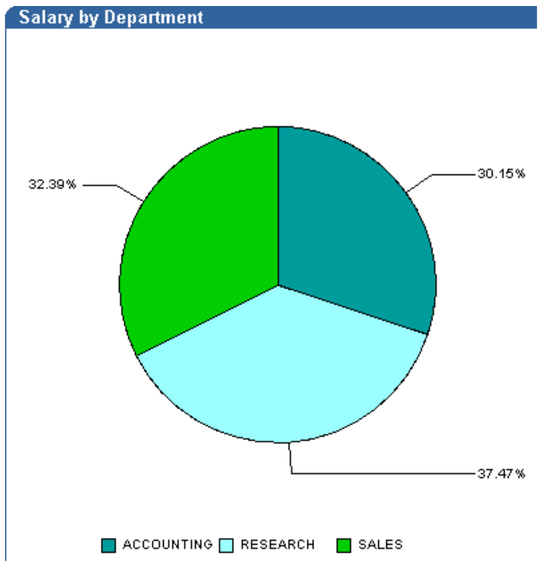
Select the action that will occur when the user clicks on the bar, pie slice or line.

Action	URL
<None>	

You can also define chart hyperlinks so that each bar, pie section, or line, links to another Web page. For example, you can display a chart portlet and a report portlet on your portal page, then set up the chart hyperlink to display a row in the report that displays more detailed information about the selected data.

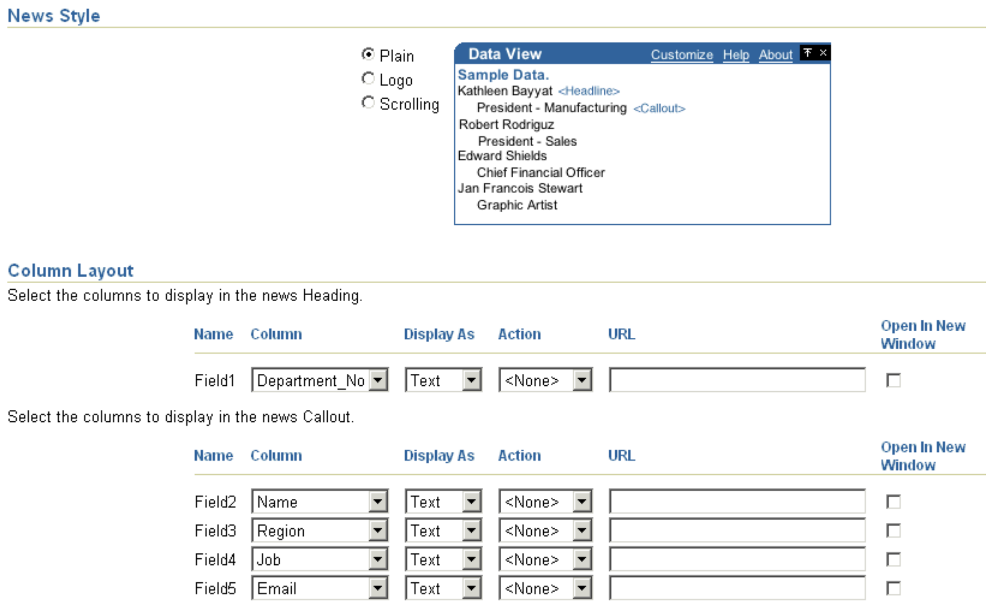
In Figure 3–15, you can see the results of the options selected on the Layout tab in the previous image. Below the chart, you can see that the category, which was Department on the Layout tab, is used for the legend.

Figure 3–15 Example of an OmniPortlet Using a Pie Chart Layout



3.1.5.3 News Layout

Figure 3–16 Layout Tab: News



News Style

(● Plain) ○ Logo ○ Scrolling

Data View — Customize Help About

Sample Data.
Kathleen Bayyat <Headline>
President - Manufacturing <Callout>
Robert Rodriguz
President - Sales
Edward Shields
Chief Financial Officer
Jan Francois Stewart
Graphic Artist

Column Layout

Select the columns to display in the news Heading.

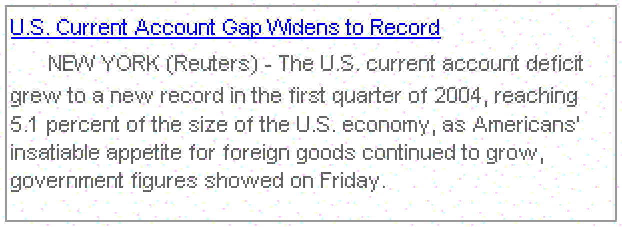
Name	Column	Display As	Action	URL	Open In New Window
Field1	Department_No	Text	<None>		☐

Select the columns to display in the news Callout.

Name	Column	Display As	Action	URL	Open In New Window
Field2	Name	Text	<None>		☐
Field3	Region	Text	<None>		☐
Field4	Job	Text	<None>		☐
Field5	Email	Text	<None>		☐

You can use the news layout to display links to articles with brief descriptions for each. You can use this layout to publish information in standard XML formats, such as RDF (Resource Description Framework) or RSS (RDF Site Summary) to your portal page. In the Column Layout section, you can add a heading that displays at the top of the portlet. You can also add a logo, or use the scrolling layout so that the user can view all the information in the portlet as it moves vertically. Here, also, you can enter a URL so that another Web page displays when the user clicks on specific data in the portlet. You can also specify whether the secondary Web page displays in a new window.

Note: The News Layout Scroll type in OmniPortlet is supported on Microsoft Internet Explorer and Netscape 7.0.



U.S. Current Account Gap Widens to Record

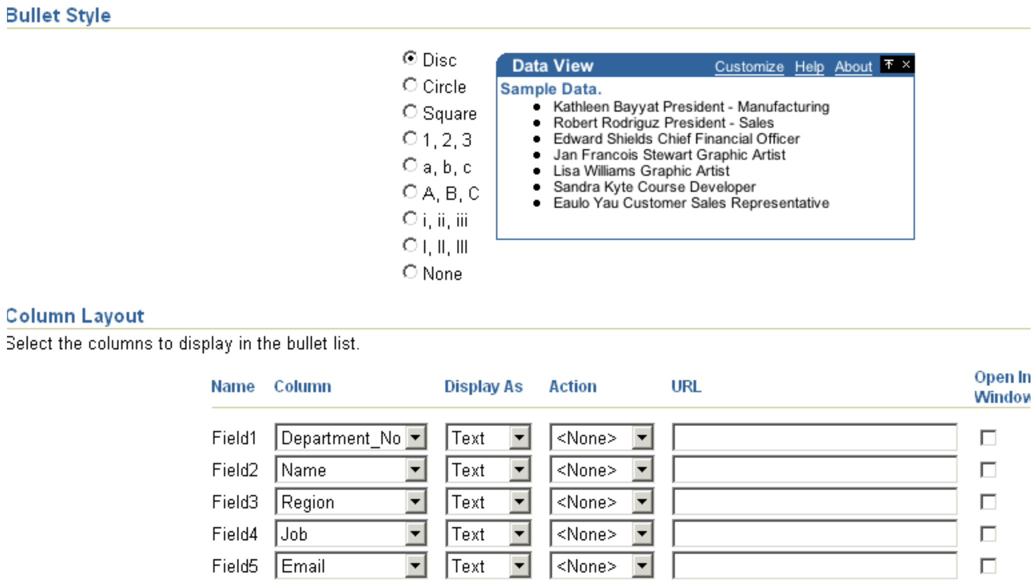
NEW YORK (Reuters) - The U.S. current account deficit grew to a new record in the first quarter of 2004, reaching 5.1 percent of the size of the U.S. economy, as Americans' insatiable appetite for foreign goods continued to grow, government figures showed on Friday.

Figure 3–17 Example of an OmniPortlet Using a News Layout

Note: For more information on using the OmniPortlet wizard, click the Help link in the upper right corner the Layout tab.

3.1.5.4 Bullet Layout

Figure 3–18 Layout Tab: Bullet



Bullet Style

[✓ Disc] Circle Square 1, 2, 3 a, b, c A, B, C i, ii, iii I, II, III None

Sample Data.

- Kathleen Bayyat President - Manufacturing
- Robert Rodriguz President - Sales
- Edward Shields Chief Financial Officer
- Jan Francois Stewart Graphic Artist
- Lisa Williams Graphic Artist
- Sandra Kyte Course Developer
- Eaulo Yau Customer Sales Representative

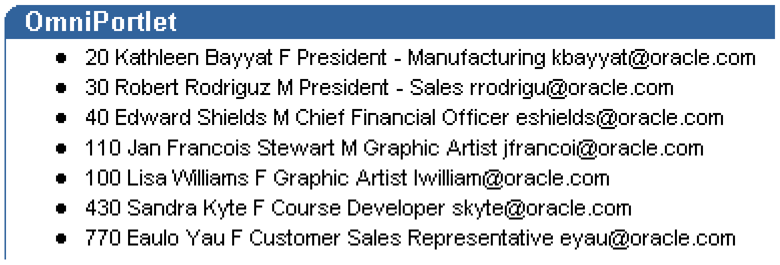
Column Layout

Select the columns to display in the bullet list.

Name	Column	Display As	Action	URL	Open In Window
Field1	Department_No	Text	<None>		☐
Field2	Name	Text	<None>		☐
Field3	Region	Text	<None>		☐
Field4	Job	Text	<None>		☐
Field5	Email	Text	<None>		☐

You can use the bullet layout to display your data in a bulleted list. The Layout tab provides a variety of different bullet and numbered bullet styles. In the Column Layout section, you can choose how the columns will display in the portlet, as well as whether a second Web page will display when the user clicks that column. You can also specify whether the second Web page displays in a new window.

Figure 3–19 Example of an OmniPortlet Using a Bullet Layout



OmniPortlet

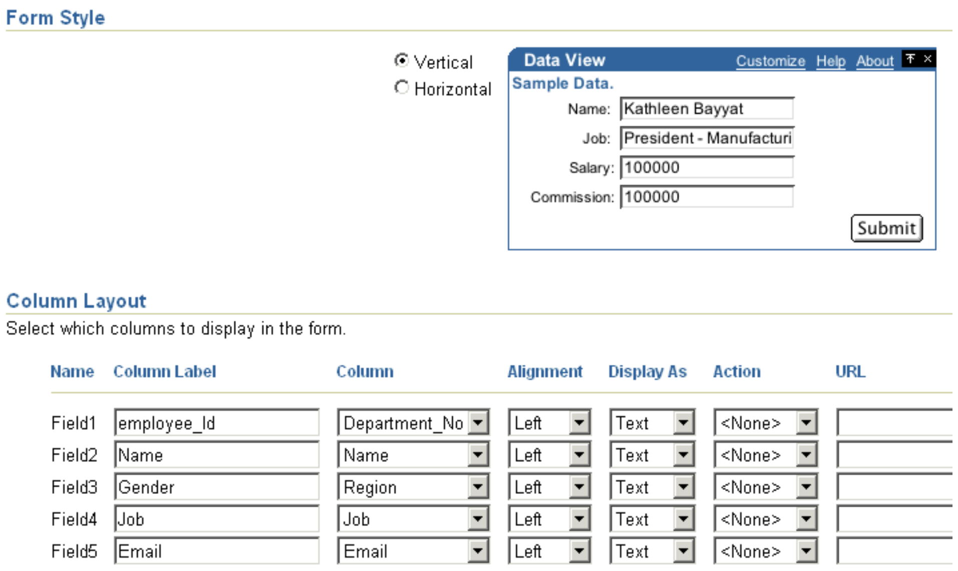
- 20 Kathleen Bayyat F President - Manufacturing kbayyat@oracle.com
- 30 Robert Rodriguz M President - Sales rrodrigu@oracle.com
- 40 Edward Shields M Chief Financial Officer eshields@oracle.com
- 110 Jan Francois Stewart M Graphic Artist jfrancoi@oracle.com
- 100 Lisa Williams F Graphic Artist lwilliam@oracle.com
- 430 Sandra Kyte F Course Developer skyte@oracle.com
- 770 Eaulo Yau F Customer Sales Representative eyau@oracle.com

---

Note: For more information on using the OmniPortlet wizard, click the Help link in the upper right corner the Layout tab.

3.1.5.5 Form Layout

Figure 3–20 Layout Tab: Form



Form Style

Vertical [✓] / Horizontal

Name: Kathleen Bayyat
Job: President - Manufacturi
Salary: 100000
Commission: 100000

Column Layout

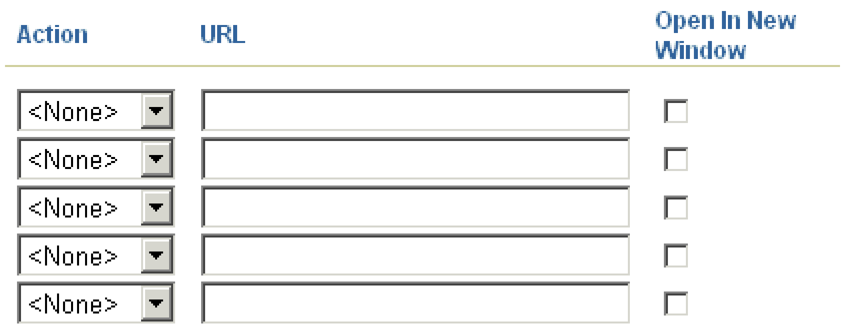
Select which columns to display in the form.

Name	Column Label	Column	Alignment	Display As	Action	URL
Field1	employee_Id	Department_No	Left	Text	<None>	
Field2	Name	Name	Left	Text	<None>	
Field3	Gender	Region	Left	Text	<None>	
Field4	Job	Job	Left	Text	<None>	
Field5	Email	Email	Left	Text	<None>	

You can use the form layout if you have data you want to display as labels or default values in a form, such as Name: <name>. You can then use the portlet parameters and events to pass data to the selected row.

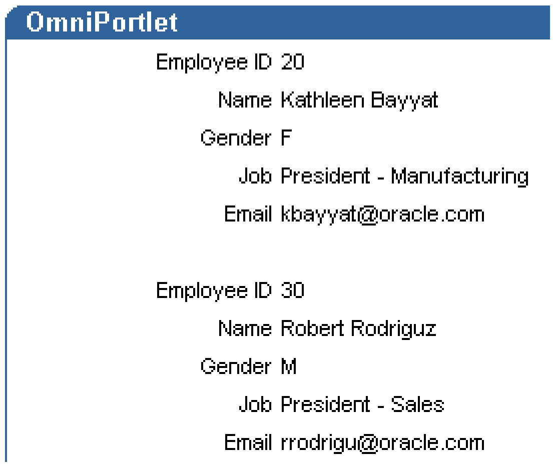
You can also specify whether to display the target of a URL in a new window:

Figure 3–21 Open In New Window Checkbox



Action	URL	Open In New Window
<None>		☐
<None>		☐
<None>		☐
<None>		☐
<None>		☐

Figure 3–22 Example of an OmniPortlet Using a Form Layout



OmniPortlet

Employee ID	20
Name	Kathleen Bayyat
Gender	F
Job	President - Manufacturing
Email	kbayyat@oracle.com
Employee ID	30
Name	Robert Rodriguz
Gender	M
Job	President - Sales
Email	rrodrigu@oracle.com

Note: For more information on using the OmniPortlet wizard, click the Help link in the upper right corner the Layout tab

3.1.6 Edit Defaults mode

After you've created your OmniPortlet and returned to your Portal page, you can then click the Edit Defaults icon to re-enter the wizard. You'll notice that you can click the tabs (except for the Type tab) to directly access each page of the wizard. You can also access the Events tab through the Edit Defaults wizard, which is explained in the next section.

When you edit an OmniPortlet through the Edit Defaults mode, keep in mind the following notes:

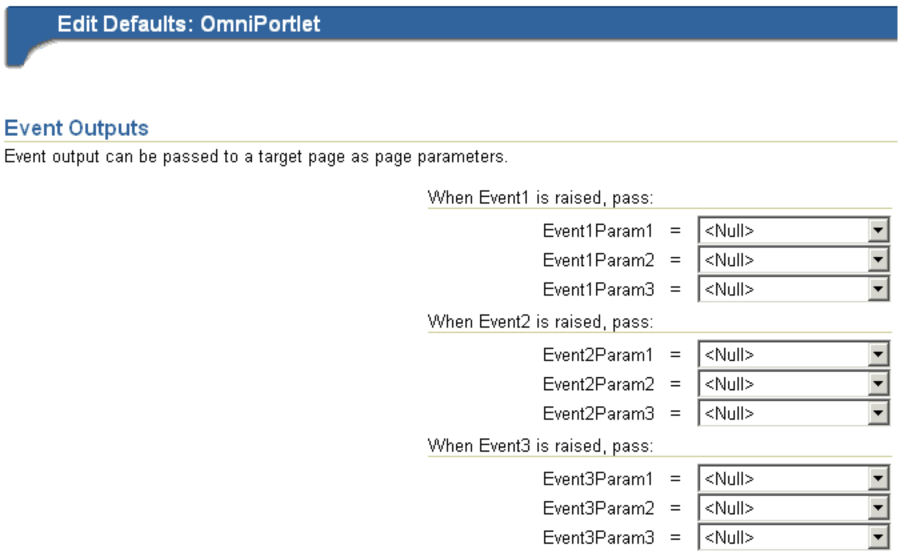
- A new mode, "none," is now the default setting for the Locale Personalization Level of OmniPortlet and the Simple Parameter form. This new mode indicates that, when you edit the portlet defaults through the Edit Defaults mode, the changes apply to all users, regardless of the current OracleAS Portal session language and the locale of your browser. For more information about these settings, refer to the "Configuring the OmniPortlet Provider" document included with the `pdksoftware.zip` file.
- You can also customize the portlet at runtime by clicking the Customize link on the portlet. When you customize the portlet, a complete copy of the

personalization object file is created. Since all properties are duplicated, subsequently modifying the portlet through Edit Defaults will not be reflected in the customized version of the portlet. To ensure the latest changes are made to the portlet, you must click **Customize** again (after the modifications through the Edit Defaults wizard are made), then select the **Reset to Defaults** option.

3.1.7 Events

On the Events tab in the Edit Defaults mode of the OmniPortlet wizard, you can identify event parameters based on the portlet parameters you selected on the Source tab.

Figure 3–23 Events Tab of the OmniPortlet Wizard



Event Outputs

Event output can be passed to a target page as page parameters.

When Event1 is raised, pass:

Event1Param1	=	<Null>
Event1Param2	=	<Null>
Event1Param3	=	<Null>

When Event2 is raised, pass:

Event2Param1	=	<Null>
Event2Param2	=	<Null>
Event2Param3	=	<Null>

When Event3 is raised, pass:

Event3Param1	=	<Null>
Event3Param2	=	<Null>
Event3Param3	=	<Null>

3.2 Parameters and Events

Out of the box, OmniPortlet can receive up to five parameters and raising up to three events. Each of the events can send one or more parameters. For example, you can set up a chart that displays the employees in a department. When the user clicks one piece of the chart (for example, a department name), an event is raised that sends a parameter to the page. The page may then pass a parameter to all the portlets on that page that display information about the employees. Then, all the portlets on the page display information about the employees in the selected department.

Note: To learn how to use parameters and events with OmniPortlet, follow the steps in Using OmniPortlet. If you are comfortable with the `provider.xml` file, you can add more parameters and events by editing the file.

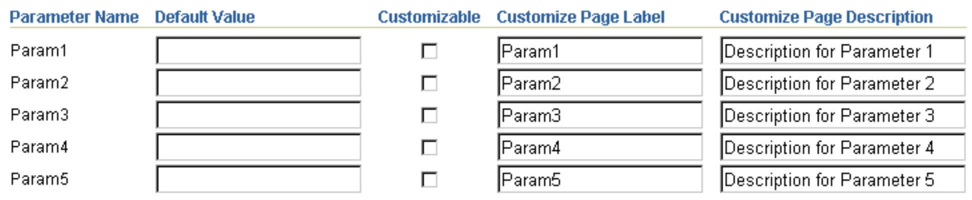
To set up parameters and events, you must first enable the page group to accept parameters and events. In Oracle Application Server 10*g*, parameters and events are by default enabled. Then, you set up each portlet to accept the necessary parameters, and raise the required events. After you've set up the portlet parameters, you can link the portlets together by setting up the page-level parameters and events. The steps to do this are described in detail later in this chapter.

3.2.1 Portlet Parameters and Events

Out of the box, you can define up to five portlet parameters for an OmniPortlet. You can do this:

- On the Source tab of the wizard when you define the OmniPortlet
- On the Source tab when you select Edit Defaults for the OmniPortlet

Figure 3–24 Source Tab: Portlet Parameters Section



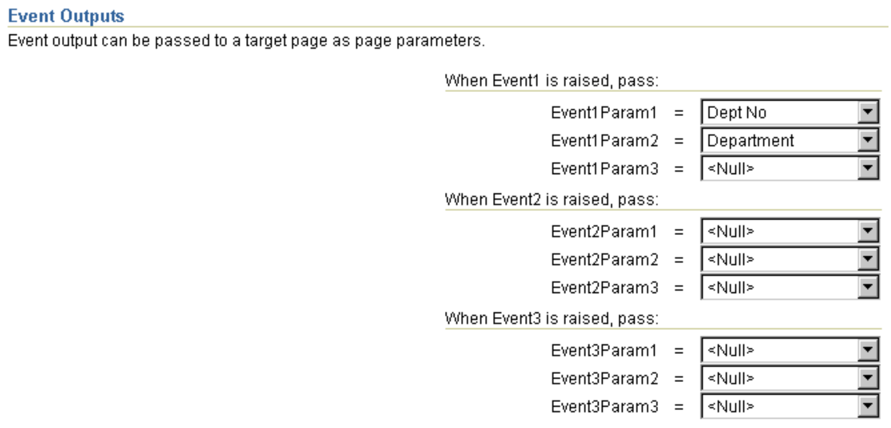
Parameter Name	Default Value	Customizable	Customize Page Label	Customize Page Description
Param1		☐	Param1	Description for Parameter 1
Param2		☐	Param2	Description for Parameter 2
Param3		☐	Param3	Description for Parameter 3
Param4		☐	Param4	Description for Parameter 4
Param5		☐	Param5	Description for Parameter 5

Parameter values determine what data is displayed in the portlet. You can also use a parameter to pass a value in a URL or to embed a value in the portlet text.

Note: You can learn more about portlet parameters in the online help, which you can access by clicking the Help link on the Source tab in the OmniPortlet wizard. The online help describes portlet parameters in detail, and how to set them up for your OmniPortlet. You can also refer to the chapter in the *Oracle Application Server Portal User's Guide*.

You can set up each OmniPortlet to raise up to three events. Each event can pass up to three parameters. Each parameter can be a portlet parameter, such as Param1, or a data source column, such as Department_No. You set up events on the Events tab when you select Edit Defaults for the OmniPortlet

Figure 3–25 Events Tab



Event Outputs

Event output can be passed to a target page as page parameters.

When Event1 is raised, pass:

Event1Param1	=	Dept No
Event1Param2	=	Department
Event1Param3	=	<Null>

When Event2 is raised, pass:

Event2Param1	=	<Null>
Event2Param2	=	<Null>
Event2Param3	=	<Null>

When Event3 is raised, pass:

Event3Param1	=	<Null>
Event3Param2	=	<Null>
Event3Param3	=	<Null>

3.2.2 Page Parameters and Events

After you've set up the parameters and events for each OmniPortlet on a portal page, you can map the portlet parameters and events to other portlets on the same page. For more information on using page parameters and events, refer to the OracleAS Portal online help and the *Oracle Application Server Portal User's Guide*.

3.3 Using OmniPortlet

The following sections will show you how to create three types of OmniPortlets (SQL data source, Web Service data source, and Web Page data source). You will learn how to create and modify these OmniPortlets, as well as use portlet parameters and events and page parameters and events to add interactivity to your portal page.

Note: To learn more about specific pages and tabs in OmniPortlet, click the Help link in the top right corner. The help topic describes the contents of the selected page or tab.

3.3.1 Adding an OmniPortlet to a Portal Page

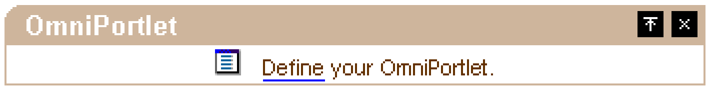
In this section, you will learn how to add an OmniPortlet instance to your portal page.

To add an OmniPortlet instance to a page:

1. In the Edit mode of the page where you want to add the OmniPortlet, click the **Add Portlets** icon.
2. On the Add Portlets page, in the Available Portlets list, click the Portlet Builders link.
3. Click the OmniPortlet link.

   If you cannot find this link, use the Search field to find OmniPortlet.
4. Click OK. The new instance of OmniPortlet now displays on your portal page.

Figure 3–26 OmniPortlet Instance on a Portal Page



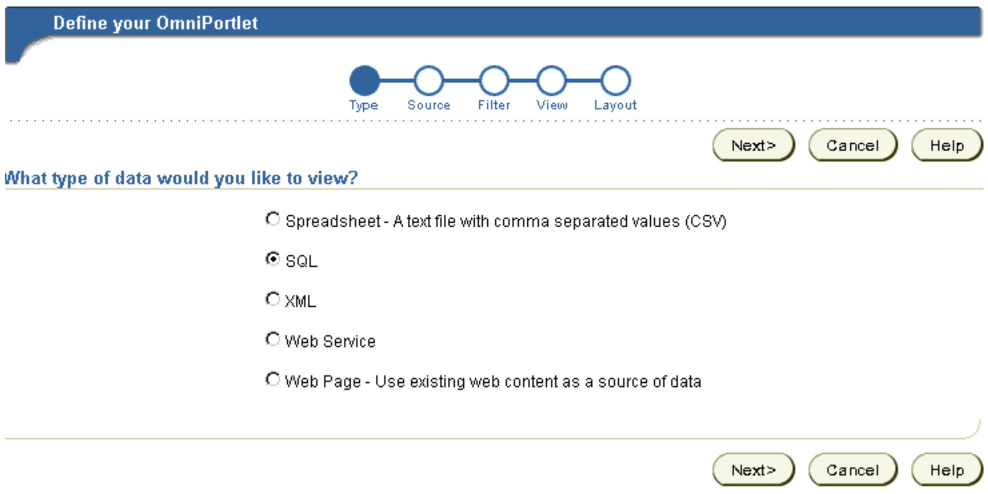
3.3.2 Defining a Portlet Based on a SQL Data Source

In this section, you'll learn how to use OmniPortlet to publish data from a database using a SQL statement.

To select the SQL data source for the portlet:

1. In the new OmniPortlet instance, click the **Define** link.
2. On the Type tab of the OmniPortlet wizard, you can choose the data source you wish to use. Select the **SQL** radio button, then click **Next**.

Figure 3–27 Type Tab of the OmniPortlet Wizard

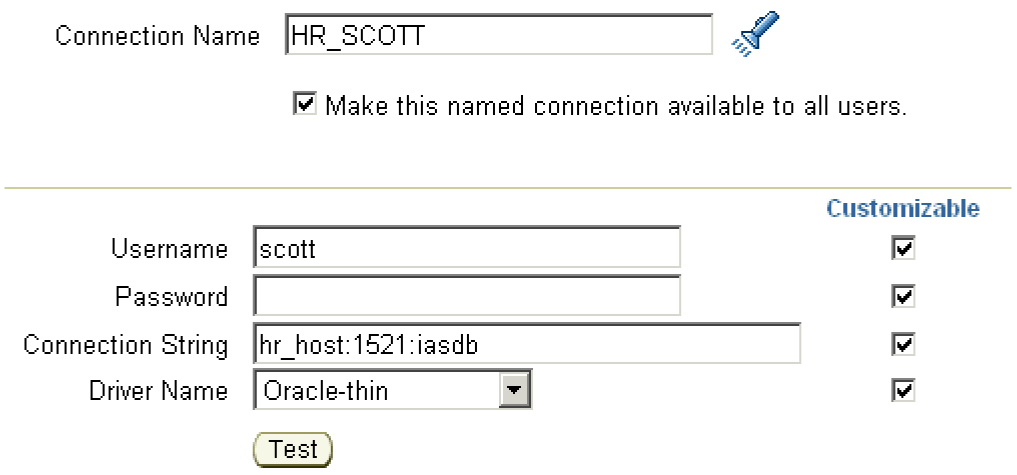


What type of data would you like to view?

- ○ Spreadsheet - A text file with comma separated values (CSV)
- [✓ SQL]
- ○ XML
- ○ Web Service
- ○ Web Page - Use existing web content as a source of data

3. On the Source tab, set the connection information for your data source by clicking **Edit Connection**.
4. On the Connection Information page, in the **Connection Name** field, enter HR_SCOTT, and make sure that the **Make this named connection available to all users** checkbox is selected.
5. In the **Username** field, enter the login ID for the sample database: `scott`.
6. In the **Password** field, enter the password for the user ID: `tiger`.
7. In the Connection String field, enter the connection string for your database, for example: `hostname:port:oracle_sid`, where `hostname` is the location where the database is installed.
8. Make sure Oracle-thin is selected from the **Driver Name** drop-down list, then click **Test** to make sure the connection information is correct.

Figure 3–28 Connection Information for the HR Scott Connection



Connection Name: HR_SCOTT

☑ Make this named connection available to all users.

		Customizable
Username	scott	☑
Password		☑
Connection String	hr_host:1521:iasdb	☑
Driver Name	Oracle-thin	☑

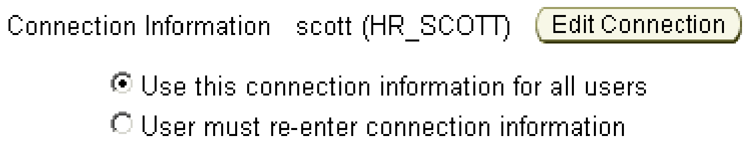
Note: If the connection does not work, verify you've entered the information appropriate to your database, or contact your database administrator.

---

9. Once the connection returns a Successful message, close the Test dialog box, then click **OK**.

   The new Connection Name now displays on the SQL Data Source: Source tab.

Figure 3–29 Connection Name on SQL Data Source Tab



Connection Information scott (HR_SCOTT)

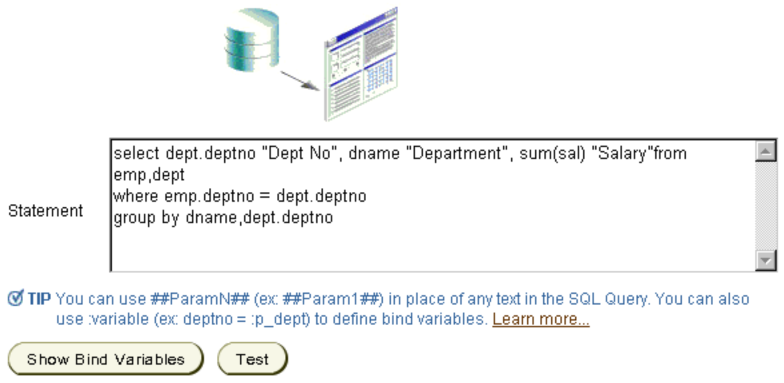
(●) Use this connection information for all users
( ) User must re-enter connection information

10. In the Statement field, enter the following code:

```
select dept.deptno "Dept No", dname "Department", sum(sal)
"Salary" from emp,dept
where emp.deptno = dept.deptno
group by dname,dept.deptno
```

11. Click **Test** to test the query against your data source, then click **Close** to close the Test dialog box.

Figure 3–30 SQL Statement on the Source Tab



Statement

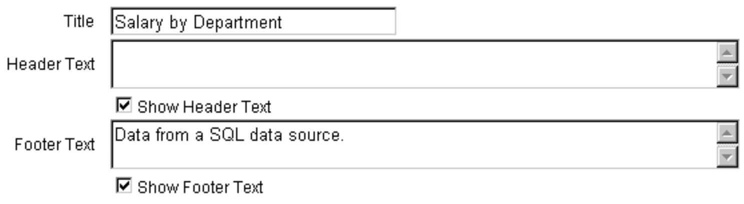
```
select dept.deptno "Dept No", dname "Department", sum(sal) "Salary"from
emp,dept
where emp.deptno = dept.deptno
group by dname,dept.deptno
```

TIP You can use ##ParamN## (ex: ##Param1##) in place of any text in the SQL Query. You can also use :variable (ex: deptno = :p_dept) to define bind variables. Learn more...

12. Once you've validated your query against the database, click **Next**.
13. Since we've already sorted our data using our SQL statement, we do not need to set the conditions on the Filter tab.

    Click **Next**.
14. On the View tab, in the **Title** field, enter `Salary by Department`.
15. Click **Next**.
16. In the Footer Text field, enter `Data from a SQL data source`.

Figure 3–31 Header and Footer section of the View Tab



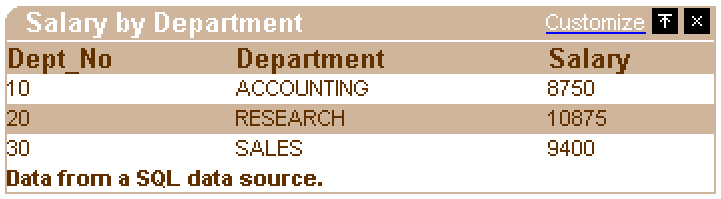
Title	Salary by Department
Header Text	
	☑ Show Header Text
Footer Text	Data from a SQL data source.
	☑ Show Footer Text

Note: In the Header Text and Footer Text fields, you can also enter HTML tags. For example, if you entered `<i>Data from a SQL data source</i>`, the text would display at runtime as *Data from a SQL data source*.

---

17. On the Layout tab, leave the default Tabular definition selected and click **Finish**. The OmniPortlet displays the results from your SQL query on your portal page.

Figure 3–32 Tabular Results from SQL Data Source



Salary by Department

Dept_No	Department	Salary
10	ACCOUNTING	8750
20	RESEARCH	10875
30	SALES	9400

Data from a SQL data source.

In this section, you learned how to use the OmniPortlet wizard to create a portlet that displays data from a SQL data source in a tabular layout. Follow the steps in the next section to learn how to modify the layout of your new OmniPortlet.

3.3.3 Defining a Portlet Based on a Web Service

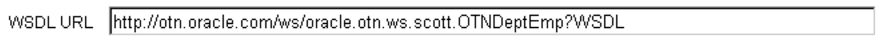
In this section, you will learn how to build a portlet based on a Web Service. For the purposes of this example, you will use a Web service provided on the Oracle Technology Network (`http://www.oracle.com/technology/index.html`), but you can use any Web Service with OmniPortlet.

To create a portlet based on a Web Service:

1. Follow the steps in Using OmniPortlet to add an instance of OmniPortlet to your portal page.
2. In the Edit mode of your page, click the **Define** link in the new OmniPortlet instance.
3. On the Type tab, select the **Web Service** radio button, then click **Next**.
4. On the Source tab, set the definition location for the Web Service (WSDL).

   Enter the following in the WSDL URL field:
   `http://otn.oracle.com/ws/oracle.otn.ws.scott.OTNDeptEmp?WSDL`

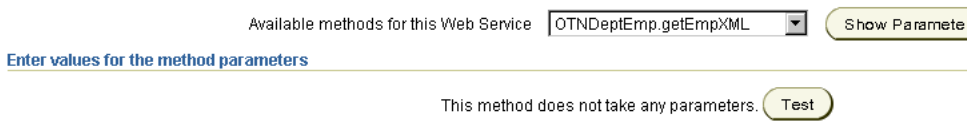
Figure 3–33 WSDL URL Field of the Source Tab


WSDL URL http://otn.oracle.com/ws/oracle.otn.ws.scott.OTNDeptEmp?WSDL

5. Click **Show Methods** to see what methods are available from this Web service.
6. From the **Available methods for this Web Service** drop-down list, choose **OTNDeptEmp.getEmpXML**, then click **Show Parameters** to see what parameters this method accepts.

> **Note:** If you use a method that has parameters, the parameters will display in this section of the tab. You can enter a sample value for the parameter, then click Test to view the sample XML data, the SOAP response, and the SOAP Request.

Figure 3–34 Web Service Methods Section of the Source Tab


Available methods for this Web Service OTNDeptEmp.getEmpXML

Enter values for the method parameters

This method does not take any parameters.

7. Click **Next**.
8. On the Filter tab, you can restrict the data returned from the data source. In this example, since the Web Service returns all data from the EMP table, you will restrict the returned data to a single department.

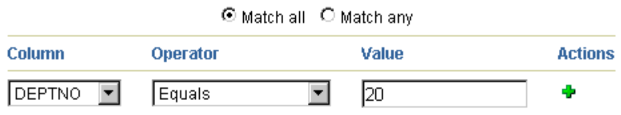
   In the Conditions section, set DEPTNO = 20 by choosing:

   a. From the Column drop-down list, choose **DEPTNO**.

   b. From the Operator drop-down list, choose **Equals**.

   c. In the Value field, enter `20`.

Figure 3–35 Conditions Section of the Filter Tab

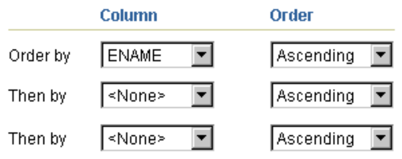

[✓ Match all] Match any

Column	Operator	Value	Actions
DEPTNO	Equals	20	✚

9. In the Order section, set the results to display in alphabetical order by employee name.

   Next to Order By, choose **ENAME** from the Column drop-down list, and make sure **Ascending** is selected in the Order drop-down list.

Figure 3–36 Order Section of the Filter Tab

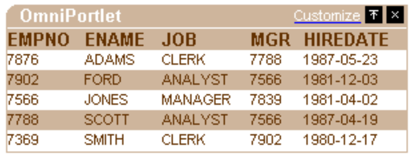


	Column	Order
Order by	ENAME	Ascending
Then by	<None>	Ascending
Then by	<None>	Ascending

10. Click **Finish**.

    Your new portlet should look like the following:

Figure 3–37 Web Service Portlet on the Portal Page



EMPNO	ENAME	JOB	MGR	HIREDATE
7876	ADAMS	CLERK	7788	1987-05-23
7902	FORD	ANALYST	7566	1981-12-03
7566	JONES	MANAGER	7839	1981-04-02
7788	SCOTT	ANALYST	7566	1987-04-19
7369	SMITH	CLERK	7902	1980-12-17

In this section, you learned how to create a portlet using a Web service as a data source, and how to use the options on the Filter tab of the OmniPortlet wizard to sort your data. In the next section, you learn how to add interactivity to your portal page by using parameters and events.

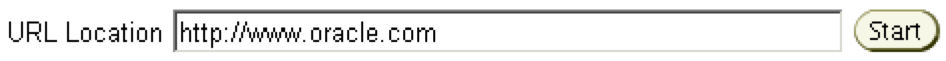
3.3.4 Defining a Portlet Based on an Existing Web Page

The steps in this section will show you how to add information from an existing Web page to your portal page. In this example, you will use information from the Oracle Web site (`http://www.oracle.com`).

To create a portlet using a Web page data source:

1. Follow the steps in Using OmniPortlet to add an OmniPortlet instance to your portal page.
2. In the Edit mode of your page, click the **Define** link in the new OmniPortlet instance.
3. On the Type tab of the OmniPortlet wizard, select the **Web Page** radio button, then click **Next**.
4. On the Source tab, click **Select Web Page**.
5. On the Web Clipping Studio page that displays, enter `http://www.oracle.com` in the URL Location field.

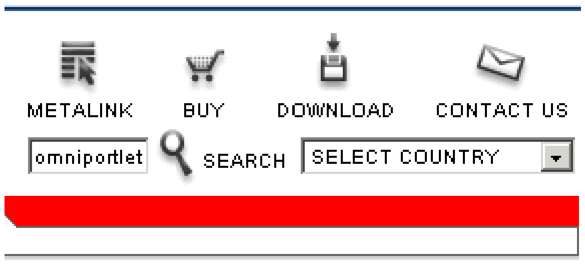
Figure 3–38 URL Location Field



URL Location: http://www.oracle.com

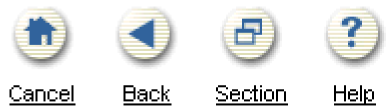
6. Click **Start** to display the Web Clipping Studio.
7. On the home page, you can type a search string in the Search field. For this example, enter `omniportlet`, then click the **Search** icon.

Figure 3–39 Searching for Information on the www.oracle.com Home Page



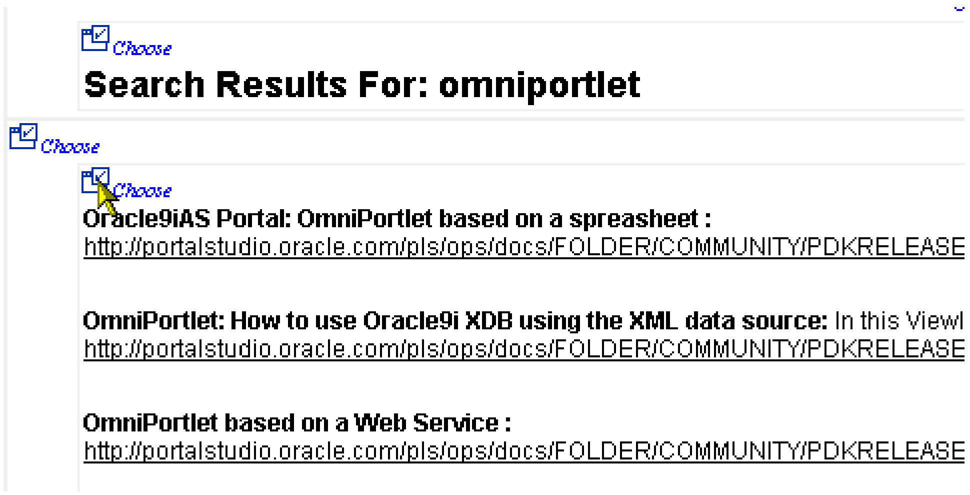
8. The Search result is displayed in the Web Clipping Studio. Click **Section** to divide the target Web page into its clippable sections.

Figure 3–40 Sectioning the Target Web Page



9. At the top left corner of the search results, click **Choose** to select the table that encloses the multiple search results.

Figure 3–41 Choosing the Search Results



Search Results For: omniportlet

Oracle9iAS Portal: OmniPortlet based on a spreasheet :
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDKRELEASE

OmniPortlet: How to use Oracle9i XDB using the XML data source: In this Viewl
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDKRELEASE

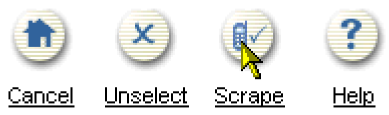
OmniPortlet based on a Web Service :
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDKRELEASE

Note: In this example, you'll notice that you can select URLs as part of your Web clipping. In Oracle Application Server Portal 10.1.2 and later, the context of the application is maintained. So, for example, any images that display on the hyperlinked page will be maintained.

---

10. After you've chosen the clipping, you can refine your data further by scraping the data, that is, selecting specific cells you wish to display in your portlet. Click **Scrape**.

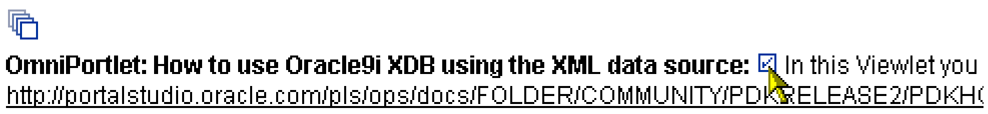
Figure 3–42 Scraping the Data in the Clipping



11. While in Scraping mode, you can identify the text pieces in the Web clipping by selecting the check boxes next to each item. You can repeat these items at the column level, row level, or table level. In this example, we want to show the title and the description of each resulting article from our search. We can repeat the title and description at the row level, so that each result returned by the search displays only the title and the description of every result. In general, you choose the text items in the first row that contains all the pieces you wish to repeat for each row.

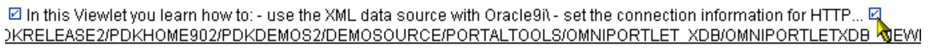
    In this step, select the output you want by selecting the check boxes next to the items. In this case, we want to display the title of the article and its description.

Figure 3–43 Selecting the Title



OmniPortlet: How to use Oracle9i XDB using the XML data source: ☑ In this Viewlet you
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDK/RELEASE2/PDKH

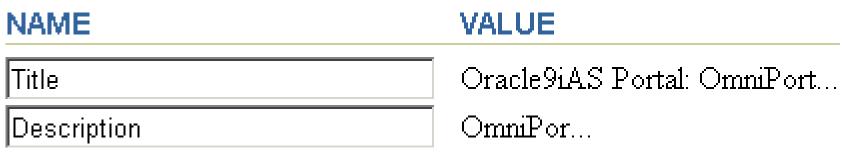
Figure 3–44 Selecting the Description



☑ In this Viewlet you learn how to: - use the XML data source with Oracle9i - set the connection information for HTTP... ☑
KRELEASE2/PDKHOME902/PDKDEMOS2/DEMOSOURCE/PORTALTOOLS/OMNIPORTLET_XDB/OMNIPORTLETXDB_VIEWI

12. Change the name of the output to be more meaningful. The following image shows an example of the new names:

Figure 3–45 Changing the Product Name



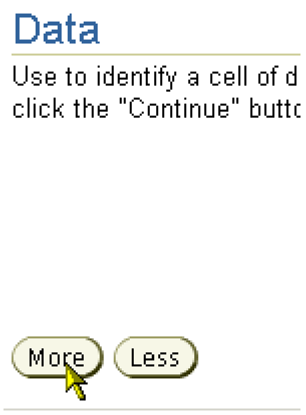
NAME	VALUE
Title	Oracle9iAS Portal: OmniPort...
Description	OmniPor...

13. After you've defined the base unit of the repeating rows (in this case, the article title and description), you can now specify the repeat level. In the Data section, if you select one or more cells in the first row of a table, you can choose to repeat the selected cells at the row, column, or table level. When you click the **More** button, check boxes in all the cells in the row of the base unit (in this case, the article titled *OmniPortlet: How to use Oracle9i XDB using the XML data source*) are highlighted to identify the repeating group.

> **Note:** The check boxes you have manually selected (when choosing the title and description) have a white background, while the check boxes you selected using the More button have a grey (or highlighted) background.

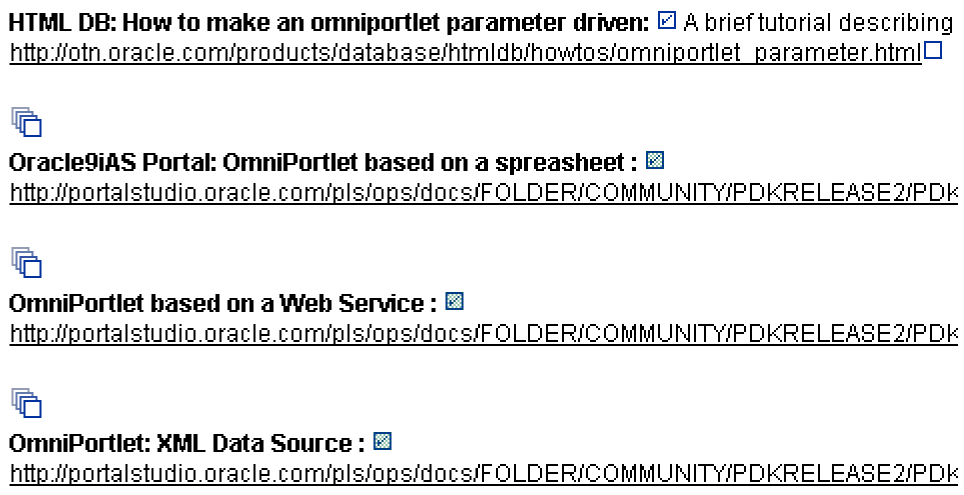
By clicking the More button, you can see that you have activated the repeating action, starting with the column level (notice that all the columns in the first row are highlighted with the first click of the **More** button). As you click the **More** and **Less** buttons, you can see how the highlighted check box positions change to identify all the repeating possibilities. You may have to click the More and Less buttons several times to find the correct repeating pattern for the data you wish to display.

Figure 3–46 More and Less Buttons



In this example, click the More button once to change the repeating level to row. Now, you will see that, for the first row, the article title and description are still selected (as indicated with the selected check boxes with a white background). In the subsequent rows, the article title and description are now also selected (as indicated with the selected check boxes with a grey background), which shows that the repeating level is at the row level.

Figure 3–47 Highlighted Check Boxes at the Row Level (Title)



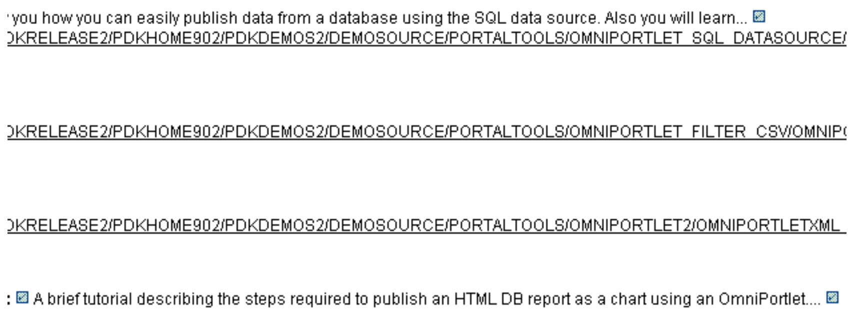
HTML DB: How to make an omniportlet parameter driven: ☑ A brief tutorial describing
http://otn.oracle.com/products/database/htmldb/howtos/omniportlet_parameter.html☐

Oracle9iAS Portal: OmniPortlet based on a spreasheet : ▨
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDKRELEASE2/PDk

OmniPortlet based on a Web Service : ▨
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDKRELEASE2/PDk

OmniPortlet: XML Data Source : ▨
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDKRELEASE2/PDk

Figure 3–48 Highlighted Check Boxes at the Row Level (Description)


'you how you can easily publish data from a database using the SQL data source. Also you will learn... ☑
)KRELEASE2/PDKHOME902/PDKDEMOS2/DEMOSOURCE/PORTALTOOLS/OMNIPORTLET_SQL_DATASOURCE/

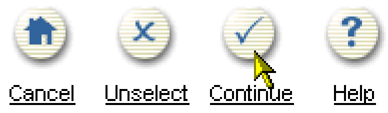
)KRELEASE2/PDKHOME902/PDKDEMOS2/DEMOSOURCE/PORTALTOOLS/OMNIPORTLET_FILTER_CSV/OMNIP(

)KRELEASE2/PDKHOME902/PDKDEMOS2/DEMOSOURCE/PORTALTOOLS/OMNIPORTLET2/OMNIPORTLETXML_

: ☑ A brief tutorial describing the steps required to publish an HTML DB report as a chart using an OmniPortlet.... ☑

14. Click the **Continue** icon.

Figure 3–49 Continue Icon



15. In the Clipping Attributes section, you can update the title and description of this portlet, so that users can see what clipped information this portlet contains. You can also change the connection time out, or the length of time in seconds, that the portlet will wait to establish a connection to the content source of the Web clipping.

Note: For more information on using the Web Clipping Studio, click the Help icon in the upper right-hand corner to view the online help.

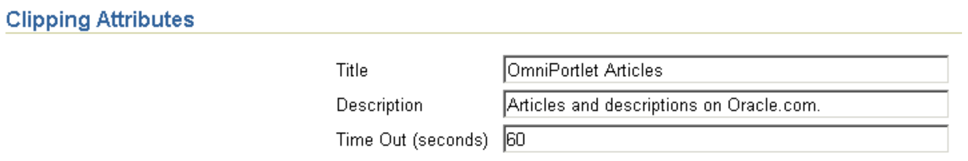
In the Title field, enter `OmniPortlet Articles`.

16. In the Description field, enter `Display all articles regarding OmniPortlet on oracle.com`.

17. In the Time Out (seconds) field, enter `60`.

    The Clipping Attributes section of the Web Clipping Studio should now look like the following image:

Figure 3–50 Clipping Attributes Section of the Web Clipping Studio


Clipping Attributes

Title	OmniPortlet Articles
Description	Articles and descriptions on Oracle.com.
Time Out (seconds)	60

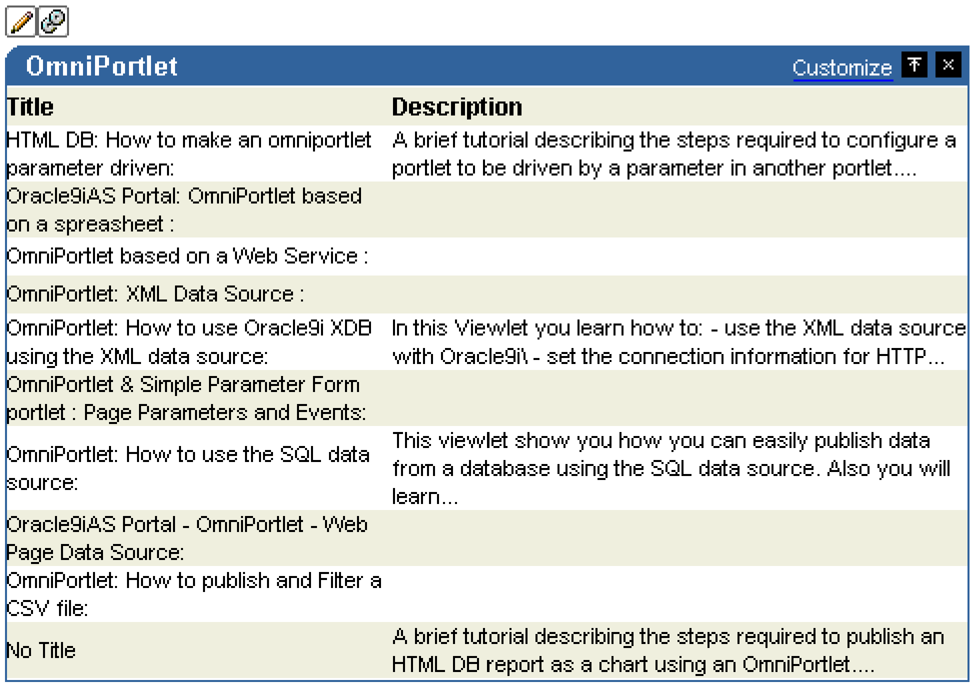
18. Click **OK**.

19. On the Source tab of the OmniPortlet wizard, the new title and description now display. To edit the Web clipping in the Web Clipping Studio, you can click the Select Web Page button again.

Click **Finish**.

20. Your new Web Page portlet displays on the portal page, and should look like the following image.

Figure 3–51 Portlet Based on Existing Web Page Content



Title	Description
HTML DB: How to make an omniportlet parameter driven:	A brief tutorial describing the steps required to configure a portlet to be driven by a parameter in another portlet....
Oracle9iAS Portal: OmniPortlet based on a spreasheet :	
OmniPortlet based on a Web Service :	
OmniPortlet: XML Data Source :	
OmniPortlet: How to use Oracle9i XDB using the XML data source:	In this Viewlet you learn how to: - use the XML data source with Oracle9i\ - set the connection information for HTTP...
OmniPortlet & Simple Parameter Form portlet : Page Parameters and Events:	
OmniPortlet: How to use the SQL data source:	This viewlet show you how you can easily publish data from a database using the SQL data source. Also you will learn...
Oracle9iAS Portal - OmniPortlet - Web Page Data Source:	
OmniPortlet: How to publish and Filter a CSV file:	
No Title	A brief tutorial describing the steps required to publish an HTML DB report as a chart using an OmniPortlet....

3.3.5 Modifying the Layout of an Existing OmniPortlet

After you've created a portlet, you may want to modify its appearance. With OmniPortlet, you can do so using its reentrant wizard.

In this section, you will learn how to modify the layout of the portlet you created in the previous section.

To modify the OmniPortlet layout:

1. In the Edit mode of the page where you added the OmniPortlet, click the **Edit Defaults** icon in the OmniPortlet.

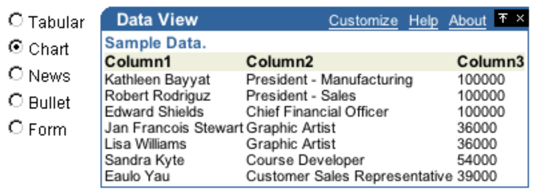
> **Note:** When you modify your OmniPortlet using the Edit Defaults mode, the changes may not appear in the Show mode of the portlet if you or the end user have already customized the portlet. If you do modify the portlet in the Edit Defaults mode, your end user must select the Reset to Defaults option on the Customize page in order for the Edit Defaults changes to appear.

2. You can modify various aspects of your OmniPortlet in the Edit Defaults mode, by clicking on the desired tab.

   To change the layout, first click the **View** tab.

3. Under Layout Style, select the **Chart** radio button, then click the **Layout** tab to modify the layout definition.

Figure 3–52 Layout Style Section of the View Tab



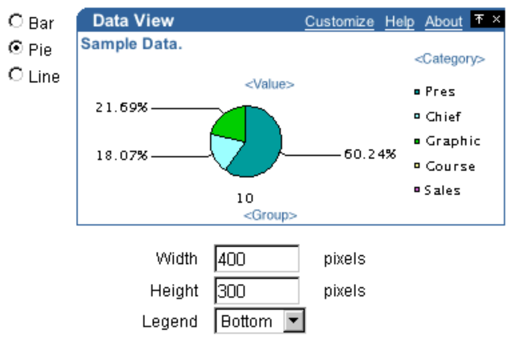
Tabular [✓ Chart] News Bullet Form

Sample Data.

Column1	Column2	Column3
Kathleen Bayyat	President - Manufacturing	100000
Robert Rodriguz	President - Sales	100000
Edward Shields	Chief Financial Officer	100000
Jan Francois Stewart	Graphic Artist	36000
Lisa Williams	Graphic Artist	36000
Sandra Kyte	Course Developer	54000
Eaulo Yau	Customer Sales Representative	39000

4. On the Layout tab, in the Chart Style section, select the **Pie** radio button to display a pie chart, then choose Bottom from the Legend drop-down list to display the chart's legend below the pie chart.

Figure 3–53 Chart Style Section of the Layout Tab



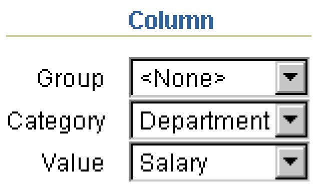
Bar [✓ Pie] Line

Width 400 pixels
Height 300 pixels
Legend Bottom

5. In the Column Layout section, select the columns to hide or display.

   From the Group drop-down list, choose **<None>** and leave the other columns as they are.

Figure 3–54 Column Layout Section of the Layout Tab



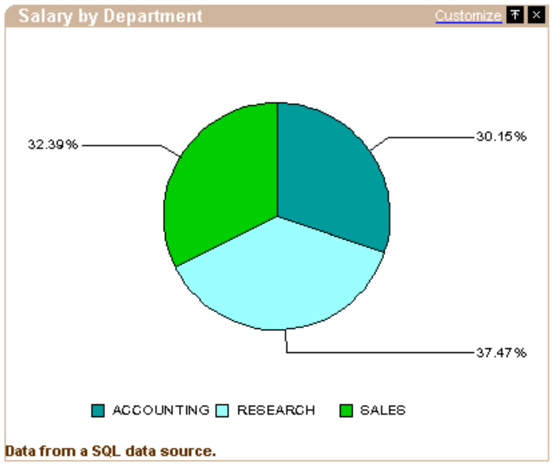
Column

Group	<None>
Category	Department
Value	Salary

6. Click **OK** to accept your changes and return to your portal page.

   Your new SQL portlet should look like the following:

Figure 3–55 SQL Data Source OmniPortlet Displaying as a Pie Chart



In this section, you learned how to return to the OmniPortlet wizard to modify the layout of your portlet, and to change the layout of your portlet from a tabular report to a pie chart.

3.3.6 Using Parameters and Events

Page parameters are used to pass values to a page and to the portlets on the page. The portlets that are configured to accept page parameters can trigger another action, depending on the contextual wiring of the parameters and events. For example, you could set up a page that displays a portlet that displays a stock ticker and a portlet that displays news about a particular company. A page parameter could then accept the stock symbol as a value, then display the relevant stock and news information in the appropriate portlets.

You can use parameters in your pages only if the Enable Parameters and Events page group setting is enabled. For information on enabling parameters and events in a page group, refer to the Oracle Application Server Portal User's Guide, Chapter 4, "Working with Page Groups," located on the Documentation page on Portal Center (`http://www.oracle.com/technology/products/ias/portal/index.html`).

Note: For more information on page parameters and events, refer to the *Oracle Application Server Portal User's Guide*, Chapter 7, Section 11, "Using Parameters and Events."

In this section, you will learn how to configure your page to support page parameters and events. You will then set up portlet parameters and events, then integrate them with your page parameters and events.

At the end of this section, the end user will be able to click a section of the pie chart that you created in Modifying the Layout of an Existing OmniPortlet. This action will display information about the selected department in the Web Service portlet you created in Defining a Portlet Based on a Web Service.

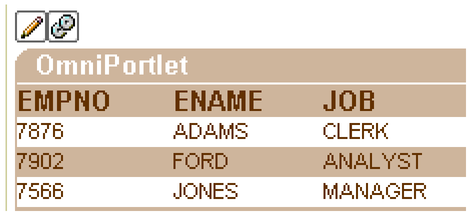
3.3.6.1 Adding Parameters to an Existing OmniPortlet

The steps in this section will show you how to add parameters to the Web Service portlet you created in Defining a Portlet Based on a Web Service.

To configure the OmniPortlet to accept parameters:

1. While you're in the Edit mode of your portal page, click the **Edit Defaults** icon in the upper left-hand corner of the Web service portlet.

Figure 3–56 Edit Defaults Icon for the Web Service Portlet

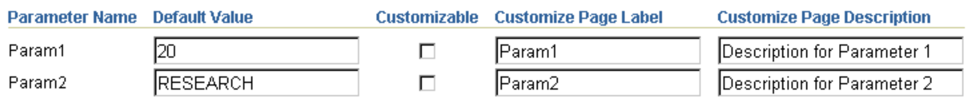


OmniPortlet

EMPNO	ENAME	JOB
7876	ADAMS	CLERK
7902	FORD	ANALYST
7566	JONES	MANAGER

2. On the Source tab that displays, locate the Portlet Parameters section, then set the following parameters:
   - Set the Default Value for Param1 to `20`
   - Set the Default Value for Param2 to `RESEARCH`

Figure 3–57 Portlet Parameters Section of the Source Tab

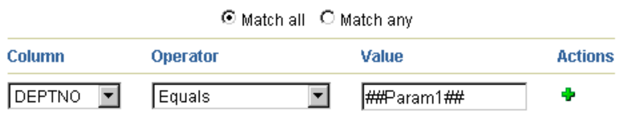


Parameter Name	Default Value	Customizable	Customize Page Label	Customize Page Description
Param1	20	☐	Param1	Description for Parameter 1
Param2	RESEARCH	☐	Param2	Description for Parameter 2

3. At the top of the screen, click the **Filter** tab.
4. Change the existing condition to:

   ```
   DEPTNO Equals ##Param1##
   ```

Figure 3–58 Condition Section of the Filter Tab



[✓ Match all] Match any

Column	Operator	Value	Actions
DEPTNO	Equals	##Param1##	✚

Note: By changing the value to ##Param1##, you set the portlet to use the value of Param1 (which you created on the Source tab) for the value of the DEPTNO column. For example, if Param1 is 10, then DEPTNO is set to 10.

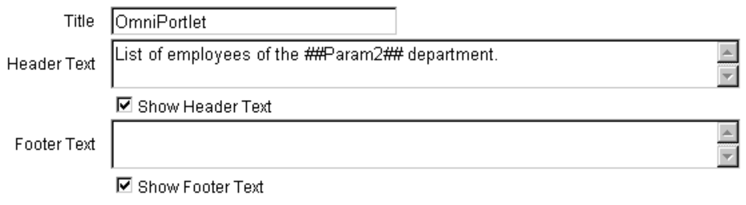
---

5. Click the **View** tab.
6. Add text to the portlet header that updates according to the selected department.

   In the Header and Footer Text section, in the Header Text field, enter the following:

```
List of employees of the ##Param2## department.
```

Figure 3–59 Header and Footer Section of the View Tab



Title	OmniPortlet
Header Text	List of employees of the ##Param2## department.
	☑ Show Header Text
Footer Text	
	☑ Show Footer Text

7. Click **OK** to save your changes to the Web service portlet.

In this section, you learned how to set portlet parameters, which you then integrated into the functionality of the portlet. Instead of simply showing the data for Department 20, portlet will display the information for the department number indicated in the parameter you used, and the header text will be updated according to the department indicated in the second parameter you used. After you set up the events for the SQL data source portlet, you will map page parameters to the portlet parameters you created in this section.

The next section will show you how to add events to the SQL portlet you created in Defining a Portlet Based on a SQL Data Source. Then, after you've set up the events, you will configure the parameters and events to work together on the portal page.

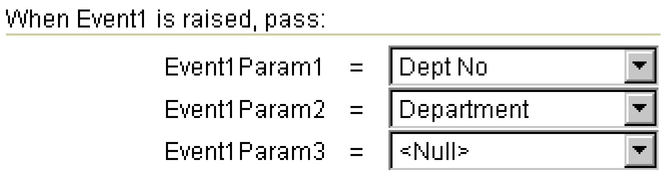
3.3.6.2 Adding Events to an Existing OmniPortlet

The steps in this section will show you how to raise an event when a parameter is set in another portlet on the same page.

To set up events in a portlet:

1. While the page is in Edit mode, click the **Edit Defaults** icon in the upper left-hand corner of the Salary by Departments portlet (that displays the pie chart).
2. Click the **Layout** tab.
3. In the Chart Drilldown section, choose **Event1** from the Action drop-down list, so that when a user clicks a section of the pie chart, Event1 occurs.
4. Click the **Events** tab.
5. Under "When Event1 is raised, pass:", set the following options:
   - Event1Param1 = **Dept No**
   - Event1Param2 = **Department**

Figure 3–60 Event Outputs Section of the Events Tab



When Event1 is raised, pass:

Event1 Param1	=	Dept No
Event1 Param2	=	Department
Event1 Param3	=	<Null>

In this section, you set up your SQL OmniPortlet to raise an event when the user clicks on a piece of the pie chart. Once the user clicks the piece, OmniPortlet retrieves the number and name for the selected department, and passes that information to the two event parameters (Event1Param1 and Event1Param2).

In the next section, you will set up the parameters you created in Adding Parameters to an Existing OmniPortlet and the events you created in this section to behave together on your portal page by mapping them to page parameters and events.

3.3.6.3 Relating Portlet Parameters and Events on a Page

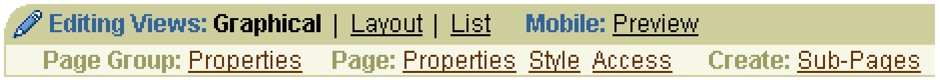
After you've created the parameters for the portlet that will control the data being passed, and set the event for the portlet that will control the data being passed, you must set the page parameters and events to relate the two portlets together on your portal page.

The steps in this section will show you how to set up the parameters at the page level, so that the parameters set in the Web Service portlet are passed to the SQL portlet, and then raise the appropriate event.

To set the page parameters and events:

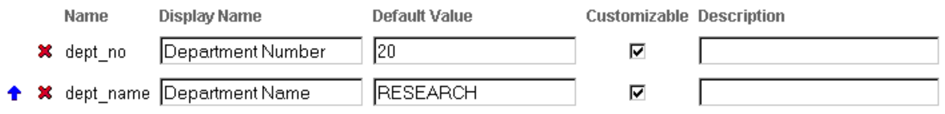
1. While you are in the Edit mode of your page, click the **Page : Properties** link in the upper left-hand corner of your page.

Figure 3–61 Page: Properties Link Located Next to the Page Group: Properties Link


Editing Views: Graphical | Layout | List Mobile: Preview
Page Group: Properties Page: Properties Style Access Create: Sub-Pages

2. Click the **Parameters** tab.
3. In the New Page Parameter section, in the Parameter Name field, enter `dept_no`, then click **Add**.
4. In the Page Parameter Properties section, next to the new dept_no parameter, enter `Department Number` in the Display Name field, and `20` in the Default Value field.
5. Follow the previous two steps to create a second page parameter called `dept_name`, and set the Display Name to `Department Name`, and the Default Value to `RESEARCH`.

Figure 3–62 Page Parameter Properties Section of the Parameters Tab



Name	Display Name	Default Value	Customizable	Description
dept_no	Department Number	20	☑	
dept_name	Department Name	RESEARCH	☑	

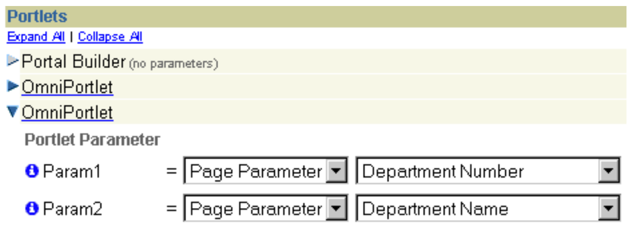
6. In the Portlet Parameter Values section, find the Web service portlet you created that contains the portlet parameters. If you created a new page specifically for the steps in this chapter, this portlet is the second instance of portlet: OmniPortlet in the list.

   Click the arrow to the left of portlet: OmniPortlet to display the Portlet Parameters for that portlet.
7. Next to Param1, choose **Page Parameter** from the drop-down list, then choose **Department Number** from the drop-down list that displays.

The portlet parameter Param1 is now mapped to the dept_no page parameter you created in the previous steps.

8. Map the Param2 portlet parameter to the Department Name page parameter.

Figure 3–63 Portlet Parameter Values Section of the Parameters Tab for the Page



Portlets

Expand All | Collapse All

▷Portal Builder (no parameters)

▶OmniPortlet

▼OmniPortlet

Portlet Parameter

Param1	=	Page Parameter	Department Number
Param2	=	Page Parameter	Department Name

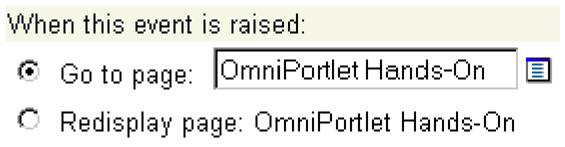
Note: Now, the portlet parameters you created for the Web service portlet in Adding Parameters to an Existing OmniPortlet are mapped to page-level parameters called dept_no (Department Number) and dept_name (Department Name).

---

9. Click the **Events** tab at the top of the screen.
10. Click the first instance of **portlets: Omniportlet**.
11. Make sure Event1 is highlighted under portlets: Omniportlet.
12. Under When this event is raised, make sure Go to Page is selected, then click the **List** icon next to the drop-down list.
13. Find your page name in the list, then click the Return Object link next to its name.

    Your page name now displays in the "Go to page" list.

Figure 3–64 Setting Portlet Event1 to Display Your Portal Page



When this event is raised:

[✓ Go to page:] OmniPortlet Hands-On

Redisplay page: OmniPortlet Hands-On

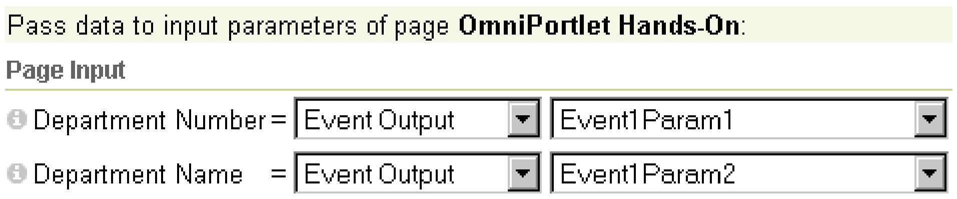
14. When you select your portal page, the page parameters display in the Page Input section.

    Next to Department Number, choose Event Output from the drop-down list, then choose Event1Param1 from the drop-down list that displays. You have now set the Department Number page parameter to display the contents of Event Parameter 1.
15. Set the Department Name page parameter to display the contents of Event Parameter 2.

    The Page Input section should now look like the following image.

Note: Now, the event parameters you created in Adding Events to an Existing OmniPortlet for the SQL data source portlet are mapped to the page parameters, which are linked to the portlet parameters. Thus, the two portlets on your page are now linked and interactive.

Figure 3–65 Page Input Section of the Event Tab for the Portal Page


Pass data to input parameters of page **OmniPortlet Hands-On**:

Page Input

Department Number =	Event Output	Event1Param1
Department Name =	Event Output	Event1Param2

16. Click **OK** to accept your changes.
17. Test your changes by clicking on a piece of the pie chart. When you click the **Accounting** section, for example, the Web service portlet automatically updates with the information for the Accounting department.

In this section, you learned how to use page parameters and events to link your portlets together on a single portal page.

3.4 Summary

In this chapter, you learned how to use OmniPortlet to build various types of portlets on a page. You also learned how to use parameters and events to integrate portlets on a page and create a portal application.

You can find more information about using the various tools in OmniPortlet by clicking the **Help** link on each of the pages in the wizard. For more information on using Web Clipping, refer to Chapter 4, "Building Content-Based Portlets with Web Clipping".

4 Building Content-Based Portlets with Web Clipping

Web Clipping is a browser-based declarative tool that enables you to integrate any Web application with OracleAS Portal. It is designed to give you quick integration by leveraging the Web application's existing user interface. Web Clipping has been implemented as a web provider using the Java Portal Developers Kit, which is a component of OracleAS Portal.

With Web Clipping, you can collect Web content into portlets in a single centralized Web page. You can use Web Clipping to consolidate content from Web sites scattered throughout a large organization.

This chapter contains the following sections:

- What Is Web Clipping?
- Adding a Web Clipping Portlet to a Page
- Integrating Authenticated Web Content Using Single Sign-On
- Example: Adding a Web Clipping That Users Can Customize
- Current Limitations for Web Clipping

4.1 What Is Web Clipping?

Web Clipping allows clipping of an entire Web page or a portion of it and reusing it as a portlet. Basic and HTML-form-based sites may be clipped. Use Web Clipping when you want to copy content from an existing Web page and expose it in your portal as a portlet. The Web Clipping portlet supports:

- **Navigation through various styles of login mechanisms,** including form- and JavaScript-based submission and HTTP Basic and Digest Authentication with cookie-based session management.
- **Fuzzy matching of clippings.** If a Web clipping gets reordered within the source page or if its character font, size, or style changes, it will still be identified correctly by the Web Clipping engine and delivered as the portlet content.
- **Reuse of a wide range of Web content,** including basic support of pages written with HTML 4.0.1, JavaScript, applets, and plug-in enabled content, retrieved through HTTP GET and POST (form submission).
- **Customization,** allowing a page designer to expose input parameters that page viewers can modify when they customize the portlet. These parameters can be exposed as public parameters that a page designer can map as OracleAS Portal page parameters. This feature allows end users to obtain personalized clippings.

- **Integrate authenticated web content through Single Sign-On,** including integration with external applications, which enables you to leverage Oracle Application Server Single Sign-On and to clip content from authenticated external Web sites.
- **Inline rendering,** enabling you to set up Web Clipping portlets to display links within the context of the portlet. As a result, when a user clicks a link in the Web Clipping portlet, the results display within the same portlet. You can use this feature with internal and external Web sites.
- **Proxy Authentication,** including support for global proxy authentication and per-user authentication. You can specify the realm of the proxy server and whether all users will automatically log in using a user name and password you provide, each user will log in using an individual user name and password, or all users will log in using a specified user name and password.
- **Migration from URL-based portlets,** enabling you to migrate your URL-based portlets to Web Clipping. For more information, refer to the following article:

  `http://www.oracle.com/technology/products/ias/portal/html/migrating_urlservices_to_webclipping.html`

  Note that although the URL was valid when this document was written, the URL may change in the future. You should be redirected to the new URL page.

By default, all Web clipping definitions are stored persistently in the Oracle Application Server infrastructure database. Any secure information, such as passwords, are stored in encrypted form, according to the Data Encryption Standard (DES), using Oracle encryption technology.

4.2 Adding Web Page Content to a Portal Page

To add Web page content to a portal page, you must first install, configure, and register the Web Clipping provider. Instructions are provided in the following file:

```
http://host:port/portalTools/webClipping/htdocs/help/configuring.webclipping.html
```

Then, you take the following steps:

1. Add the Web Clipping portlet to the portal page, as described in Section 4.2.1.
2. Use the Web Clipping Studio to navigate to the Web page containing the desired content, and then select the portion of the page to clip, as described in Section 4.2.2.
3. Set the properties of the Web Clipping portlet, as described in Section 4.2.3.

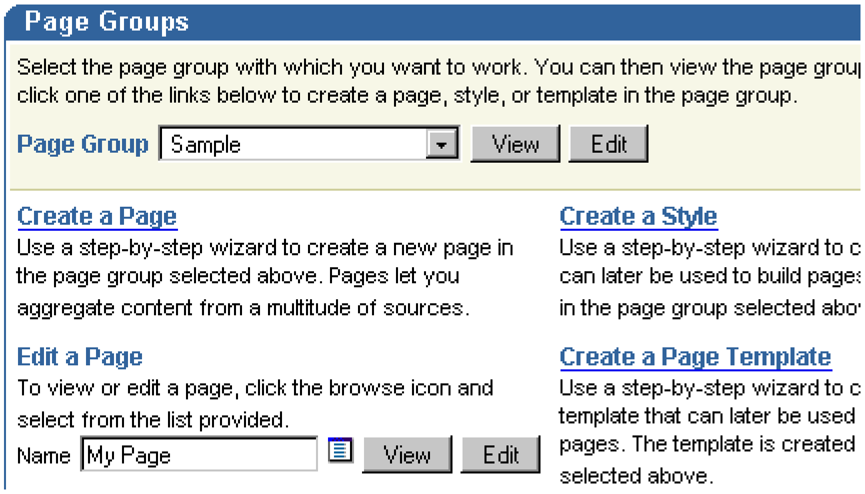
4.2.1 Adding a Web Clipping Portlet to a Page

To add a Web Clipping portlet to an OracleAS Portal page:

1. Navigate to the Page Groups portlet. By default, the Page Groups portlet is located on the **Build** tab of the Portal Builder page.
2. In the **Edit a Page** section of the Page Groups portlet, click the **Browse Pages** icon and select the page to which you want to add the Web Clipping portlet.

   Figure 4–1 shows the Page Groups portlet.

Figure 4–1 Selecting a Page


Page Groups

Select the page group with which you want to work. You can then view the page grou
click one of the links below to create a page, style, or template in the page group.

Page Group: Sample | View | Edit

Create a Page
Use a step-by-step wizard to create a new page in the page group selected above. Pages let you aggregate content from a multitude of sources.

Create a Style
Use a step-by-step wizard to c
can later be used to build page
in the page group selected abo

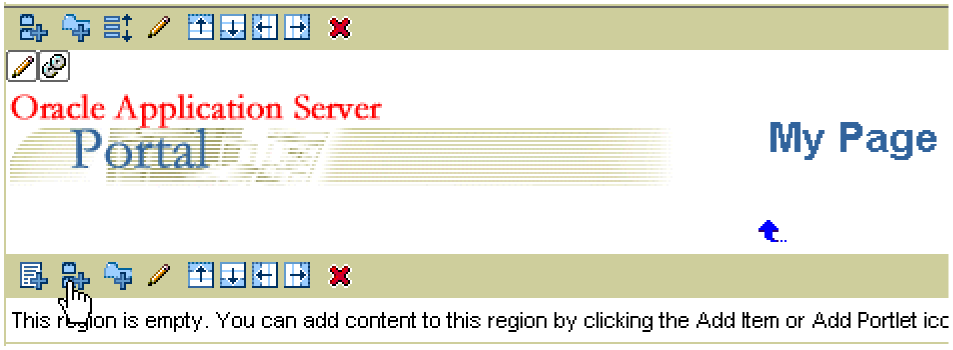
Edit a Page
To view or edit a page, click the browse icon and select from the list provided.
Name: My Page | View | Edit

Create a Page Template
Use a step-by-step wizard to c
template that can later be used
pages. The template is created
selected above.

3. Click **Edit.**
4. In the region in which you want to add the Web Clipping portlet, click the **Add Portlet** icon.

   Figure 4–2 shows a portion of the page.

Figure 4–2 Adding a Portlet to a Page



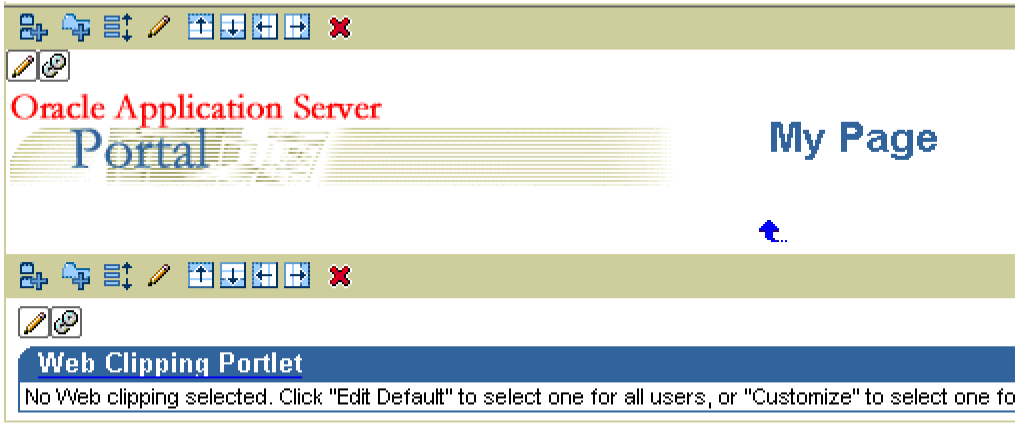
This region is empty. You can add content to this region by clicking the Add Item or Add Portlet icc

5. In the Add Portlets page, navigate to the **Web Clipping Portlet** link and click it. The **Web Clipping Portlet** moves to the **Selected Portlets** box.

   By default, the Web Clipping portlet is located in the Portal Builder page of the Portlet Repository. If you cannot find this page, use the **Search** field to find the portlet.
6. Click **OK** to add a Web Clipping portlet to your page.

   Figure 4–3 shows the Web Clipping portlet added to your page.

Figure 4–3 Web Clipping Portlet Added to a Page



No Web clipping selected. Click "Edit Default" to select one for all users, or "Customize" to select one fo

4.2.2 Selecting a Section of a Web Page to Display in the Web Clipping Portlet

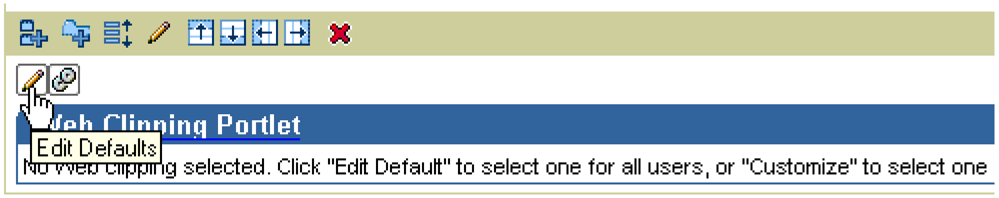
To select a section of a Web page to display in the Web Clipping portlet, you use the Web Clipping Studio. Using the Web Clipping Studio, you can:

- Browse for Web content
- Section the chosen target page
- Choose the exact portion of the Web content to clip
- Preview the clipped content as a portlet
- Save the clipped content as a portlet
- Set portlet properties and save the updated portlet information

To select a section of a Web page to display in the Web Clipping portlet:

1. Above the Web Clipping portlet, click the **Edit Defaults** icon, as shown in Figure 4–4.

Figure 4–4 Editing Default Settings

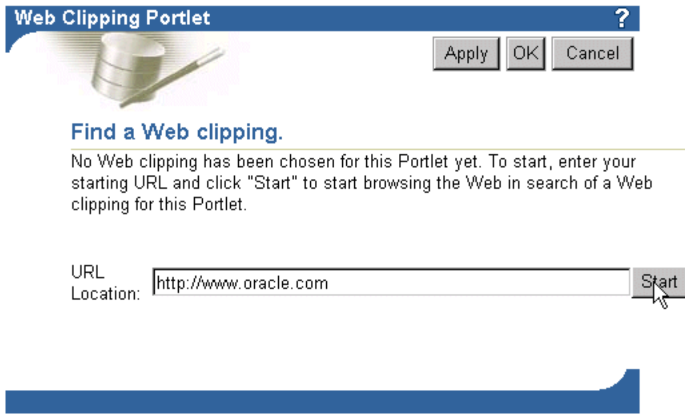


No Web clipping selected. Click "Edit Default" to select one for all users, or "Customize" to select one

   The Find a Web Clipping page is displayed.

2. In the **URL Location** field, enter the location of the starting Web page that links to the content you want to clip.
3. Click **Start**, as shown in Figure 4–5.

Figure 4–5 Specifying a URL



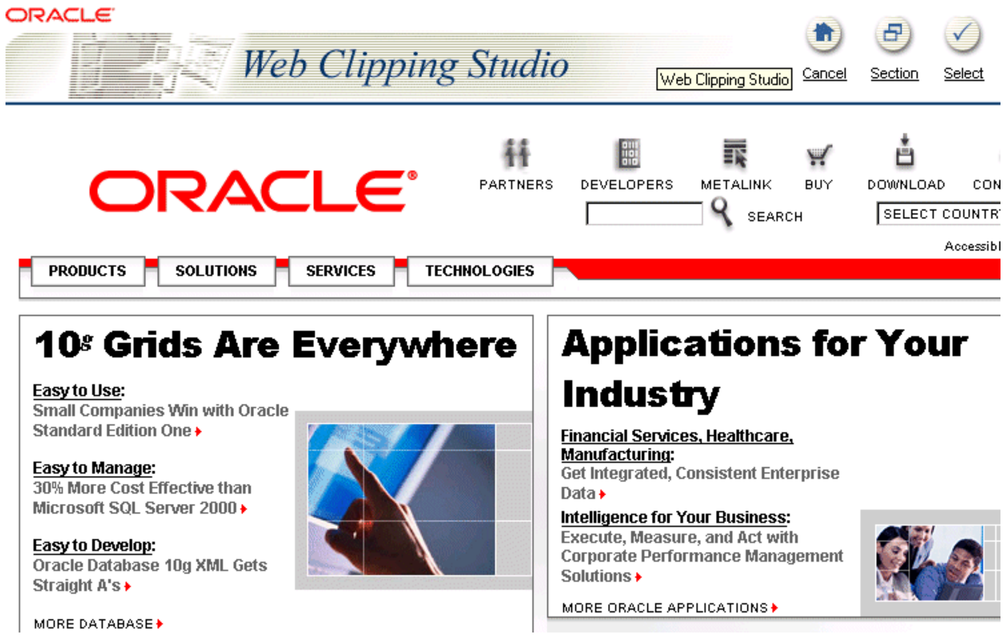
Find a Web clipping.

No Web clipping has been chosen for this Portlet yet. To start, enter your starting URL and click "Start" to start browsing the Web in search of a Web clipping for this Portlet.

URL Location: http://www.oracle.com Start

The Web Clipping Studio displays the page you specified, as shown in Figure 4–6.

Figure 4–6 Browsing to a Page Containing Content for a Web Clipping



10g Grids Are Everywhere

Easy to Use:
Small Companies Win with Oracle Standard Edition One

Easy to Manage:
30% More Cost Effective than Microsoft SQL Server 2000

Easy to Develop:
Oracle Database 10g XML Gets Straight A's

MORE DATABASE

Applications for Your Industry

Financial Services, Healthcare, Manufacturing:
Get Integrated, Consistent Enterprise Data

Intelligence for Your Business:
Execute, Measure, and Act with Corporate Performance Management Solutions

MORE ORACLE APPLICATIONS

Note that the URL in the browser bar changes from:

```
http://hostname:port/portal/page?_dad=portal&_schema=PORTAL...
```

To:

```
http://hostname:port/portalTools/webClipping...
```

4. Browse to the page that contains the content you want to clip.

As you click hyperlinks in the Web page, your navigation links are recorded.

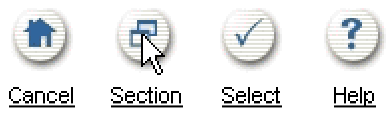
Note: Any browsing operations that do not contribute to the eventual Web clipping will be discarded. Only the significant browsing operations are recorded for later playback during the show mode; any discarded links are not visited.

For any Web sites that require HTTP Basic or Digest Authentication, a form is displayed that requests user name and password information. This encoded authentication information is recorded as part of the browsing information.

---

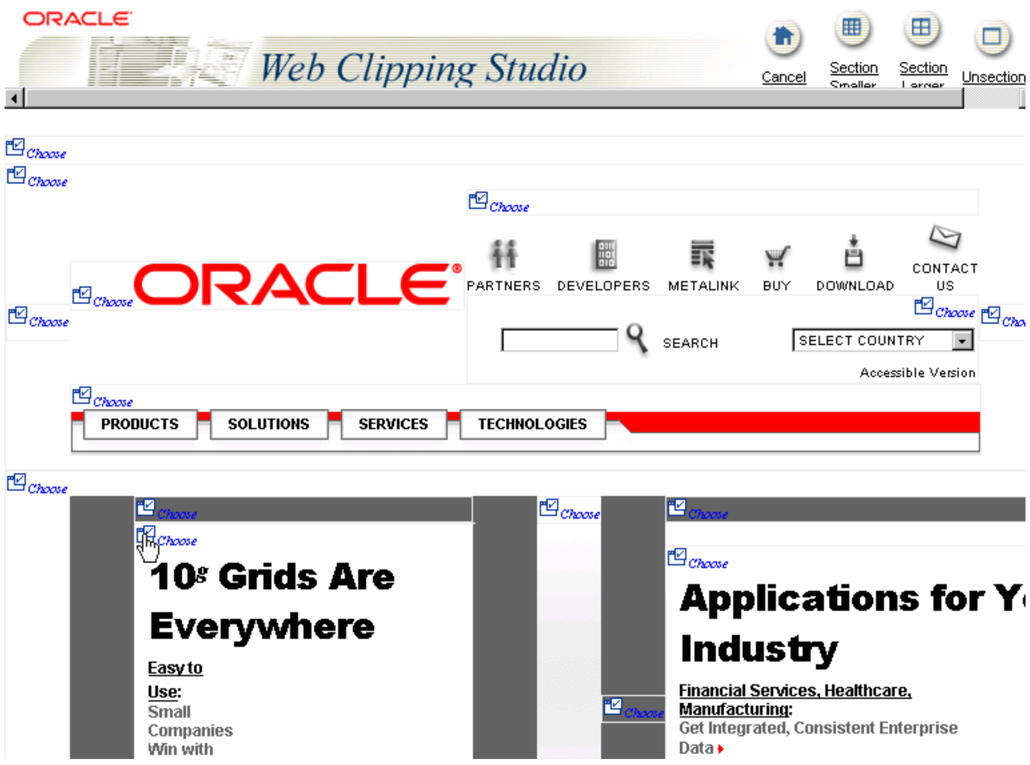
5. Once you display the page that contains the content you want to clip, in the Web Clipping Studio banner, click **Section**, as shown in Figure 4–7.

Figure 4–7 Sectioning the Target Web Page



Sectioning divides the target Web page into its clippable sections, as shown in Figure 4–8. After you click **Section**, you are no longer able to browse links in the displayed page. If you want to continue navigation, click **Unsection** in the Web Clipping Studio banner.

Figure 4–8 Sectioned Target Web Page



6. At the top left of the section of the Web content you want to clip, click **Choose**.

   You can choose only one section as a clipping at a time.

Note: To increase the number of sections available from which to choose, click **Section Smaller** in the Web Clipping Studio banner. For example, you would click **Section Smaller** to drill down one level of nested tables. To decrease the number of sections available from which to choose, click **Section Larger**.

7. Web Clipping Studio displays a preview of your chosen section. If it is the section you want, click **Select** in the Web Clipping Studio banner. The Web Clipping Studio displays the Find a Web Clipping page, with the properties of the clipping.

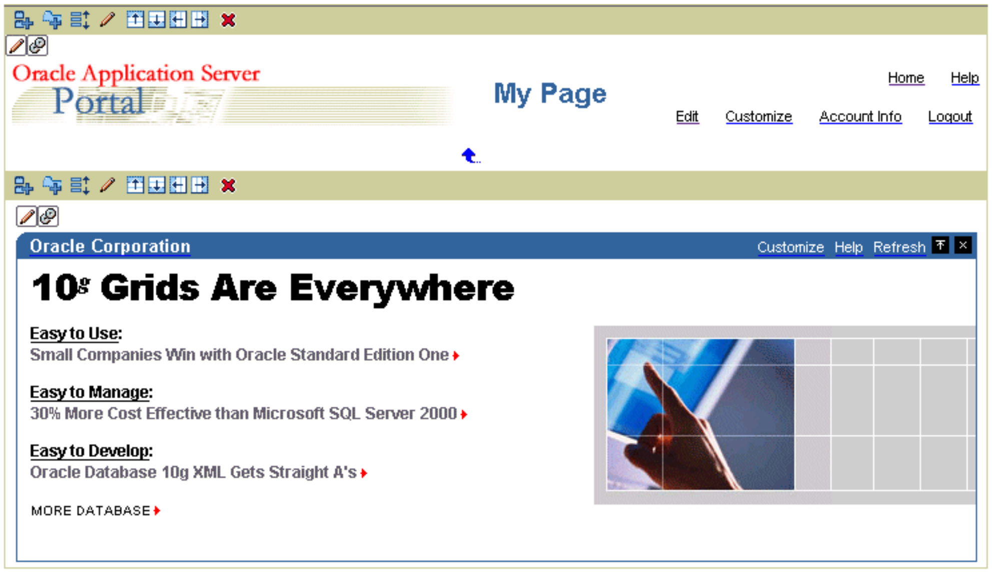
   If you do not want to use the section you clipped in your portlet, click **Unselect** to return to the page containing the section. You can choose another section on the page, or click **Unsection** to navigate to another page.

   Some sections may contain no data, only whitespace. For example, a Web page may contain an HTML <DIV> tag that contains no text or images. If you click **Choose** on a section that contains no data, Web Clipping displays a preview, but the preview correctly shows only whitespace. In this case, click **Unselect** in the preview page to return to the sectioned page. Then, select a section containing data.

8. In the Find a Web Clipping page, click **OK** to display the selected Web clipping in the Web Clipping portlet on your page. (You can edit default properties in the page. See Section 4.2.3 for more information.)

   Figure 4–9 shows the content added to the Web Clipping portlet.

Figure 4–9 Clipped Content Added to the Web Clipping Portlet on a Portal Page



10g Grids Are Everywhere

Easy to Use:
Small Companies Win with Oracle Standard Edition One

Easy to Manage:
30% More Cost Effective than Microsoft SQL Server 2000

Easy to Develop:
Oracle Database 10g XML Gets Straight A's

MORE DATABASE

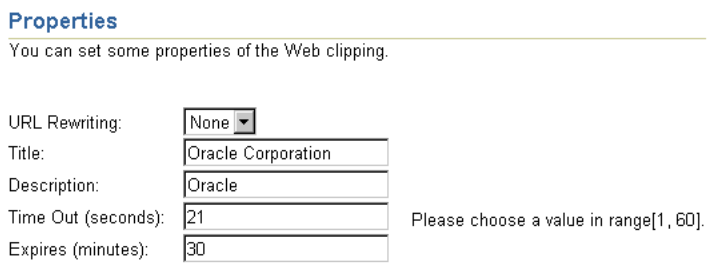
4.2.3 Setting Web Clipping Portlet Properties

You can edit various portlet settings to change the appearance of the Web Clipping portlet and to specify how end users can interact with the portlet.

To set Web Clipping portlet properties:

1. Above the Web Clipping portlet, click the **Edit Defaults** icon. Web Clipping Studio displays the Find a Web Clipping page with a Properties section, as shown in Figure 4–10.

Figure 4–10 Properties Section of Find a Web Clipping Page



Properties

You can set some properties of the Web clipping.

URL Rewriting:	None	
Title:	Oracle Corporation	
Description:	Oracle	
Time Out (seconds):	21	Please choose a value in range[1, 60].
Expires (minutes):	30	

2. From the **URL Rewriting** list in the **Properties** section, choose **Inline** if you want link targets to be displayed inside the portlet, or choose **None** if you want link targets to be displayed in a new browser window.
3. In the **Title** field, enter a title to display in the portlet banner.
4. In the **Description** field, enter a description of the portlet.
5. In the **Time Out (seconds)** field, enter the amount of time (in seconds) for the Web Clipping provider to attempt to contact the Web page from which the content was clipped.
6. In the **Expires (minutes)** field, enter the amount of time (in minutes) that cached content is valid. Any requests for portlet content that occur within the time period you specify will be satisfied from the cache.

   Once the cache period is exceeded, requests for portlet content will be satisfied by retrieving content from the portlet's Web Clipping data source. The cache will also be refreshed with this content.
7. If you entered any information in a form while clipping content for the Web Clipping portlet, the **Parameterize Inputs** section is available. Select the **Click to start parameterizing** check box to customize parameters associated with the Web Clipping portlet content. Then:
   a. From the **Parameters** list, choose the parameters that you want to customize.
   b. From the **Customizable** list, select a parameter if you want to allow end users to provide their own values for the parameters when they customize the portlet. Select **None** if you do not want to allow this.
   c. In the **Display Name** field, enter a name to be displayed for the parameter.
   d. In the **Default Value** field, enter a value to use by default for the parameter.

   Section 4.4.3 provides an example of customizing parameters.
8. Click **OK**.

4.3 Integrating Authenticated Web Content Using Single Sign-On

You can integrate an external application into a Web Clipping portlet, leveraging Oracle Application Server Single Sign-On to clip content from authenticated external Web sites. For example, if you have an account with My Oracle (an external application) that requires a login to access a particular page, you can incorporate clips from that page into a Web Clipping portlet.

To integrate an external application, take the following steps:

1. Set up the external application in OracleAS Portal, specifying the authentication information. Refer to the *Oracle Application Server Portal Configuration Guide* for more detail.

   a. Log in to OracleAS Portal as the **orcladmin** user.

   b. Navigate to the Administer External Applications portlet. (Select the **Administer** tab, then select the **Portal** subtab. In the **SSO Server Administration** section which is in the middle column, select **Administer External Applications.**)

   c. Click **Add External Application.**

   d. In the Create External Application page, in the **Application Name** field, enter a name for the application. For example, **My.Oracle.Com.**

   e. For **Login URL,** enter the URL to log on to the application, for example, **http://my.oracle.com/portal/page?_pageid=0,53&_dad=moc&_schema=MOC.** To determine the URL, navigate to the desired URL in a browser and note the URL.

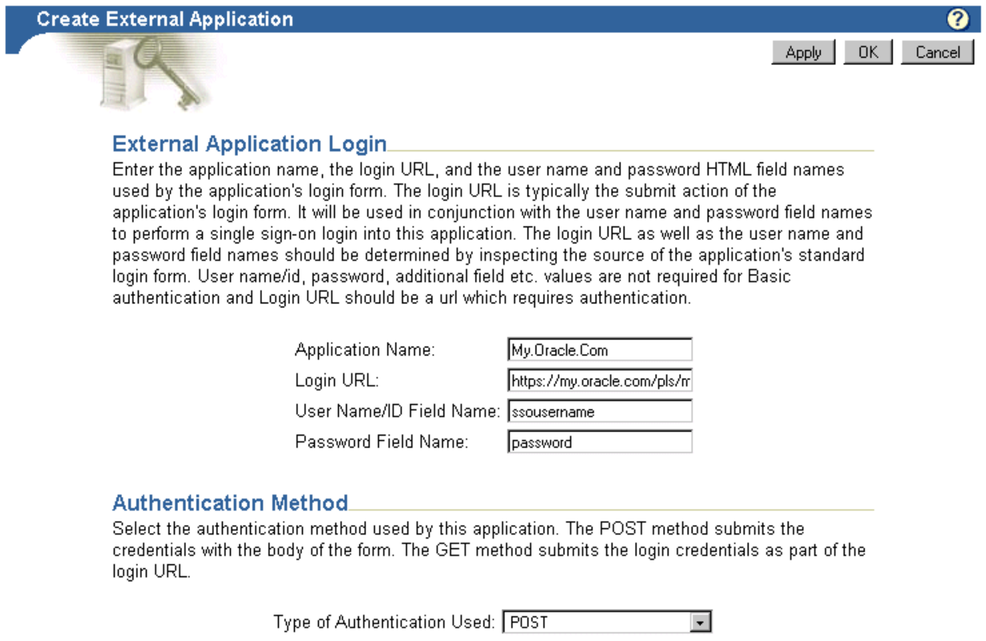
   f. For **User Name/ID Field Name,** enter the field name that the external application uses for the user name. Determine the field name by viewing the source for the desired page. In this case, enter **ssousername.**

   g. For **Password Field Name,** enter the field name that the external application uses for the password. Determine the field name by viewing the source for the desired page. In this case, enter **password.** OracleAS Portal uses this information in connecting to the application.

   h. Select **POST** as the authentication method.

   Figure 4–11 shows a portion of the Create External Application page:

Figure 4–11 Creating an External Application



External Application Login

Enter the application name, the login URL, and the user name and password HTML field names used by the application's login form. The login URL is typically the submit action of the application's login form. It will be used in conjunction with the user name and password field names to perform a single sign-on login into this application. The login URL as well as the user name and password field names should be determined by inspecting the source of the application's standard login form. User name/id, password, additional field etc. values are not required for Basic authentication and Login URL should be a url which requires authentication.

Application Name:	My.Oracle.Com
Login URL:	https://my.oracle.com/pls/m
User Name/ID Field Name:	ssousername
Password Field Name:	password

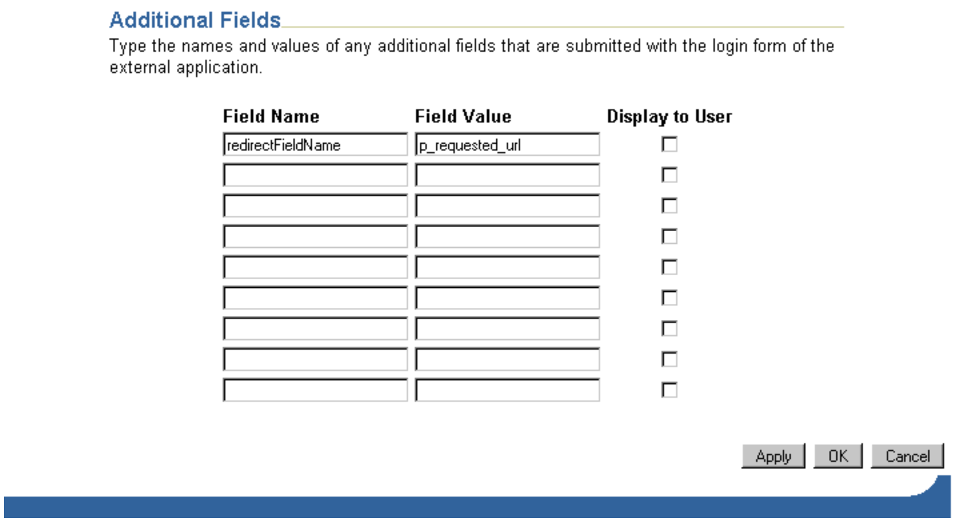
Authentication Method

Select the authentication method used by this application. The POST method submits the credentials with the body of the form. The GET method submits the login credentials as part of the login URL.

Type of Authentication Used: POST

i. In the **Additional Fields** section, you can enter names and values of any additional fields that are submitted with the login form of the external application. To specify the page to be redirected to after you log in, enter **redirectFieldName** for **Field Name.** For example, for My Oracle, enter a **Field Value** of **p_requested_url.** Figure 4–12 shows the Additional Fields section.

Figure 4–12 Specifying Redirection



Additional Fields

Type the names and values of any additional fields that are submitted with the login form of the external application.

Field Name	Field Value	Display to User
redirectFieldName	p_requested_url	☐
		☐
		☐
		☐
		☐
		☐
		☐
		☐
		☐

j. Click **OK.**

k. To test your credentials with My Oracle, in the Administer External Applications page, click the name of the application you just created. Then, in the External Application Login page, log on to the application My Oracle using your My Oracle user name and password.

In the Administer External Applications page click **Close.**

For more information about OracleAS Single Sign-On and External Applications, see the *Oracle Application Server Single Sign-On Administrator's Guide*.

2. For the Web Clipping portlet, create a new Web Clipping Provider:

a. Select the **Administer** tab, then select the **Portlets** tab.

b. Select **Register a Provider.**

c. In the Register Provider page, enter **webClippingMyOracle** for the **Name** and **Web Clipping MyOracle** for the **Display Name.** Enter a Timeout and Timeout Message. Select **Web** for the **Implementation Style.**

d. Click **Next.**

e. In the **General Properties** section of the Define Connection page, for the **URL,** enter:

```
http://server:port/portalTools/webClipping/providers/webClipping
```

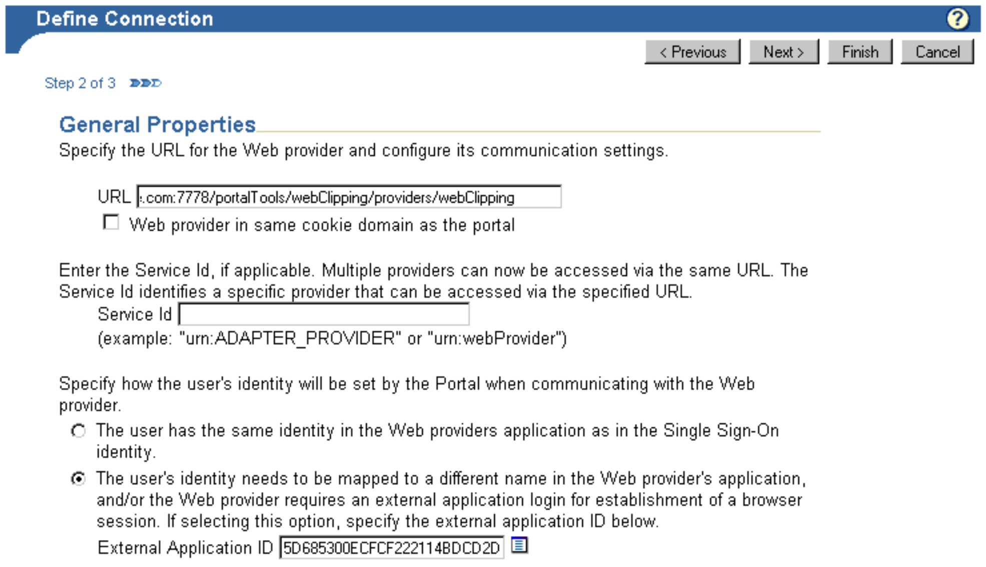
Note the *server:port* refer to the host and port where the providers are located.

f. For the user's identity, select **The user's identity needs to be mapped to a different name in the Web provider's application, and/or the Web provider requires an external application login for establishment of a browser session. If selecting this option, specify the external application ID below.**

g. For **External Application ID**, click the **List of Values** icon and select the external application you registered.

   Figure 4–13 shows the top part of the Define Connections page.

Figure 4–13 Specifying an External Application for a Web Clipping Provider



Step 2 of 3

General Properties

Specify the URL for the Web provider and configure its communication settings.

URL .com:7778/portalTools/webClipping/providers/webClipping

☐ Web provider in same cookie domain as the portal

Enter the Service Id, if applicable. Multiple providers can now be accessed via the same URL. The Service Id identifies a specific provider that can be accessed via the specified URL.

Service Id

(example: "urn:ADAPTER_PROVIDER" or "urn:webProvider")

Specify how the user's identity will be set by the Portal when communicating with the Web provider.

○ The user has the same identity in the Web providers application as in the Single Sign-On identity.

● The user's identity needs to be mapped to a different name in the Web provider's application, and/or the Web provider requires an external application login for establishment of a browser session. If selecting this option, specify the external application ID below.

External Application ID 5D685300ECFCF222114BDCD2D

h. In the **User/Session Information** section, select **User** to send user specific information to the provider. For **Login Frequency,** select **Once Per User Session.**

i. Check that the proxy settings are correct. If you use a proxy server to contact the Web providers from the middle tier, enter the proxy server for **Middle Tier.**

   If you use a proxy server to contact the Web providers from the portal repository, enter the proxy server for **Portal Repository.**

j. Click **Finish.**

k. In the Registration Confirmation page, if the registration was successful, click **OK.**

3. Add a portlet to a page, using the **Web Clipping MyOracle** provider that you just created:

   a. In OracleAS Portal, navigate to the page in which you want to add the portlet (See Section 4.2.1 for information about navigating to a page.)

   b. In the region in which you want to add the Web Clipping portlet, click the **Add Portlet** icon.

   c. In the Add Portlets page, search for **Web Clipping MyOracle.** Click it to move it to the **Selected Portlets** box.

   d. Click **OK.**

   Section 4.2.1 describes in detail how to add a portlet.

4. Select a section of a page to display in the Web Clipping portlet:

   a. Above the Web Clipping portlet, click the **Edit Defaults** icon.

      The Find a Web Clipping page is displayed.

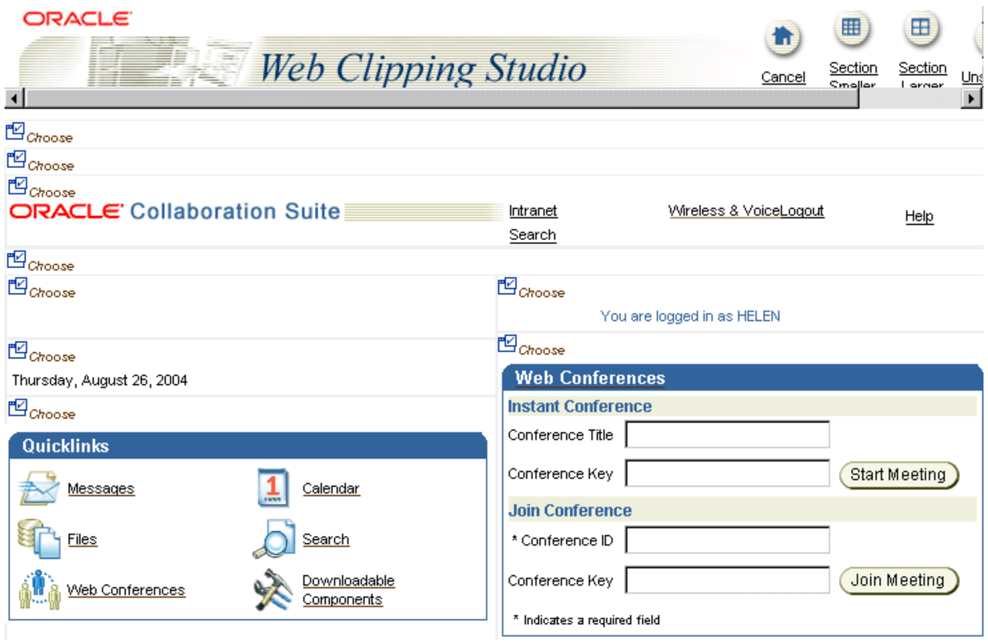
b. In the **URL Location** field, enter the location of the starting Web page that links to the content you want to clip. In this case, enter **http://my.oracle.com.**

c. Click **Start.** The Web Clipping Studio displays the page you specified.

d. Enter the login information for My Oracle.

e. Browse to the page that contains the content you want to clip. After you display the page that contains the content you want to clip, click **Section** in the Web Clipping Studio banner. Figure 4–14 shows the external application displayed in Web Clipping Studio.

Figure 4–14 External Application in Web Clipping Studio

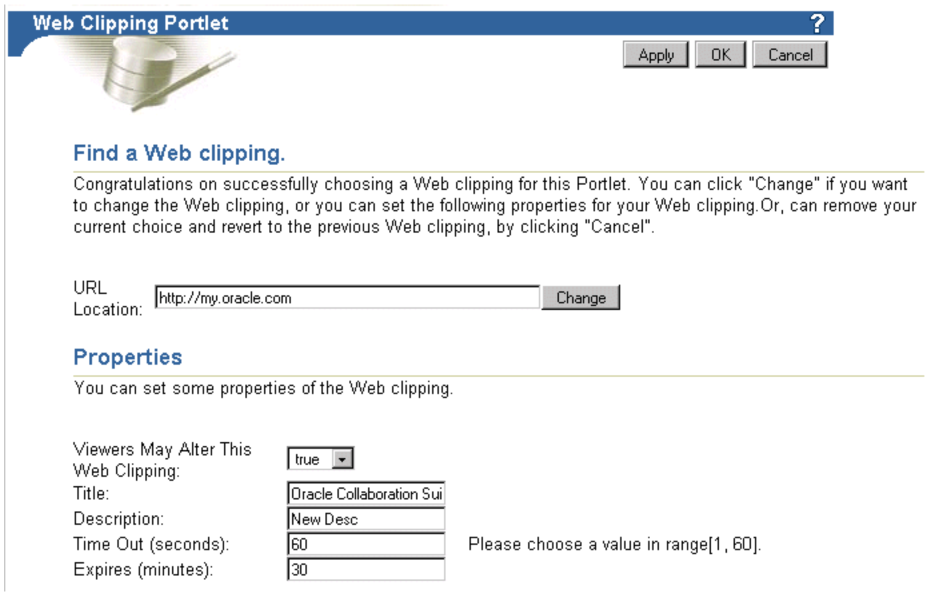


f. At the top left of the section of the Web content you want to clip, click **Choose.**

g. Web Clipping Studio displays a preview of your chosen section. If it is the section you want, click **Select** in the Web Clipping Studio banner.

The Web Clipping Studio displays the Find a Web Clipping page, with the properties of the clipping, as shown in Figure 4–15.

Figure 4–15 Properties of the External Application



Find a Web clipping.

Congratulations on successfully choosing a Web clipping for this Portlet. You can click "Change" if you want to change the Web clipping, or you can set the following properties for your Web clipping.Or, can remove your current choice and revert to the previous Web clipping, by clicking "Cancel".

URL Location: http://my.oracle.com Change

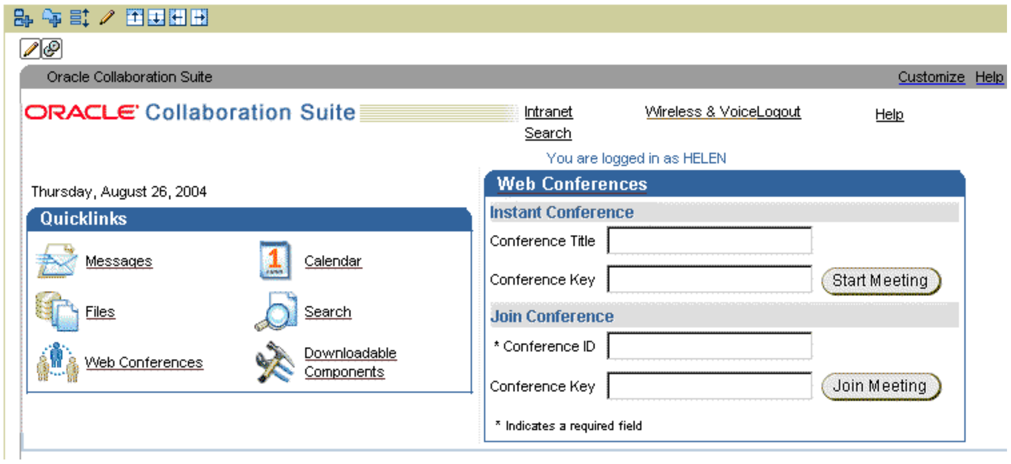
Properties

You can set some properties of the Web clipping.

Viewers May Alter This Web Clipping:	true	
Title:	Oracle Collaboration Sui	
Description:	New Desc	
Time Out (seconds):	60	Please choose a value in range[1, 60].
Expires (minutes):	30	

h. In the Find a Web Clipping page, click **OK** to display the selected Web clipping in the Web Clipping portlet on your page, as shown in Figure 4–16.

Figure 4–16 External Application Displayed in Portlet



Now, the Web clipping, even though it is from a page requiring authentication, is available in your portlet.

Note that you can only associate one external application with a provider. For each external application, you must register a new provider. Each portal user accesses the authenticated content using their user name and password for that system, not the page designer's credentials.

4.4 Example: Adding a Web Clipping That Users Can Customize

In this example, you use the Web Clipping portlet to add information from Portal to a portal page and you allow end users to customize their own views.

This example contains the following exercises:

- Exercise: Adding a Web Clipping Portlet to a Personal Page
- Exercise: Selecting a Clipping in OTN
- Exercise: Customizing a Web Clipping Portlet

4.4.1 Exercise: Adding a Web Clipping Portlet to a Personal Page

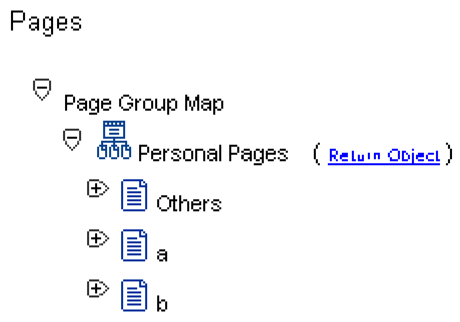
Administrators can set up personal pages for all users. This exercise assumes that the administrator has enabled this functionality. In this exercise, you add the Web Clipping portlet to your personal page.

1. In the **Edit a Page** section of the Page Groups portlet, click the **Browse Pages** icon.

   By default the Page Groups portlet is located on the Build tab of the Portal Builder page.

2. In the Page Group Map, expand the **Personal Pages** node, then expand the node for the first letter of your user name. Figure 4–17 shows the Personal Pages node.

Figure 4–17 Expanding Page Group Map Nodes



3. Click Return Object next to your user name. Your personal page is displayed.
4. Click **Edit Page.**
5. In any portlet region, click the **Add Portlets** icon.
6. In the Add Portlets page, click the **Web Clipping Portlet** link.

   By default, the Web Clipping portlet is located in the Portal Tools page of the Portlet Repository. If you cannot find this page, use the **Search** field to find the portlet.

7. The Web Clipping portlet is added to the Selected Portlets list. Click **OK.**

4.4.2 Exercise: Selecting a Clipping in OTN

In this exercise, you navigate to Oracle Technology Network (OTN) and search for specific information, then select the results as the clipping for your portlet.

1. Above the Web Clipping portlet, click the **Edit Defaults** icon.
2. In the Web Clipping Studio's Find a Web Clipping page, in the **URL Location** field, enter

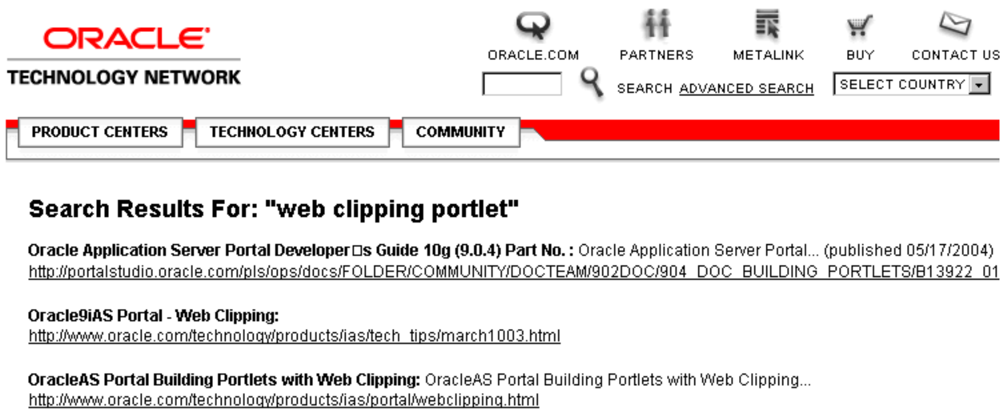
   `http://www.oracle.com/technology/products/ias/portal/index.html`

   Click **Start.** OTN displays the Oracle Application Server Portal page.

3. Enter a search string in the **Search** field at the top of the page, as shown in Figure 4–18. For this exercise, enter **"web clipping portlet"**, then click **Go.**

   The Search result is displayed in the Web Clipping Studio.

Figure 4–18 Searching for Information on OTN



Search Results For: "web clipping portlet"

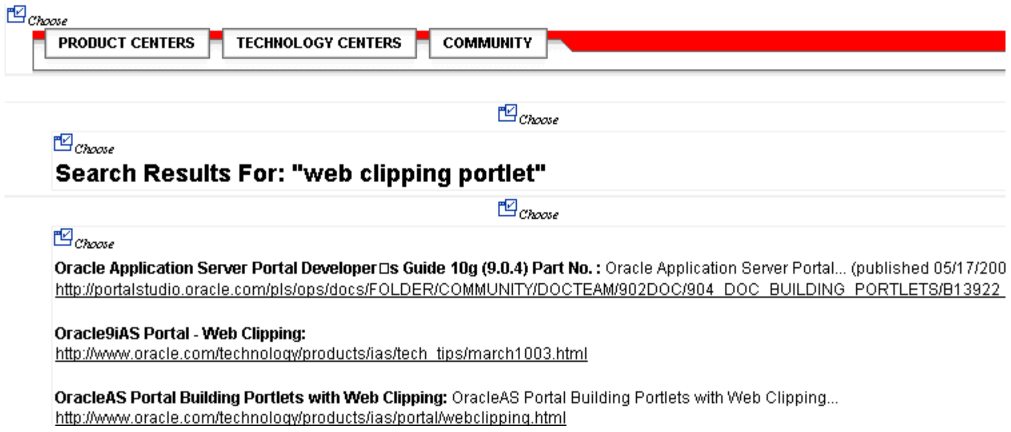
Oracle Application Server Portal Developer□s Guide 10g (9.0.4) Part No. : Oracle Application Server Portal... (published 05/17/2004)
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/DOCTEAM/902DOC/904_DOC_BUILDING_PORTLETS/B13922_01

Oracle9iAS Portal - Web Clipping:
http://www.oracle.com/technology/products/ias/tech_tips/march1003.html

OracleAS Portal Building Portlets with Web Clipping: OracleAS Portal Building Portlets with Web Clipping...
http://www.oracle.com/technology/products/ias/portal/webclipping.html

4. Click **Section**. Web Clipping Studio divides the target Web page into its clippable sections, as shown in Figure 4–19.

Figure 4–19 Sectioning the Target Web Page



Search Results For: "web clipping portlet"

Oracle Application Server Portal Developer□s Guide 10g (9.0.4) Part No. : Oracle Application Server Portal... (published 05/17/200
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/DOCTEAM/902DOC/904_DOC_BUILDING_PORTLETS/B13922_

Oracle9iAS Portal - Web Clipping:
http://www.oracle.com/technology/products/ias/tech_tips/march1003.html

OracleAS Portal Building Portlets with Web Clipping: OracleAS Portal Building Portlets with Web Clipping...
http://www.oracle.com/technology/products/ias/portal/webclipping.html

5. At the top left corner of the search result, click **Choose.**

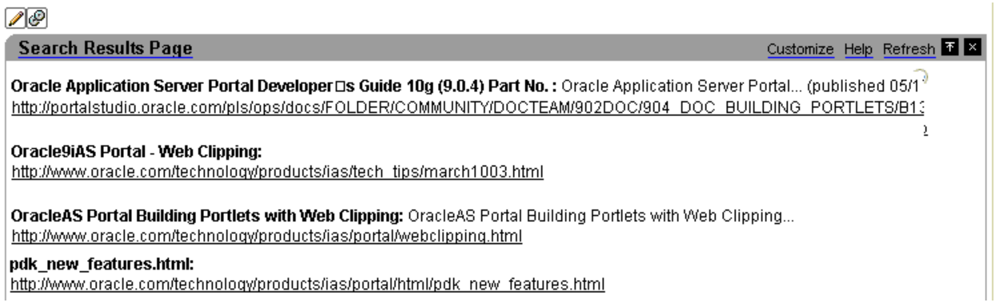
   A preview of the search result section displays.

   Some sections may contain no data, only whitespace. For example, a Web page may contain an HTML <DIV> tag that contains no text or images. If you click **Choose** on a section that contains no data, Web Clipping displays a preview, but the preview correctly shows only whitespace. In this case, click **Unselect** in the preview page to return to the sectioned page. Then, select a section containing data.

6. Click **Select** to confirm that the search result section is the one you want to clip.

7. In the Find a Web Clipping page, click **OK** to display the selected Web clipping in the Web Clipping portlet on your page. Figure 4–20 shows the Web Clipping displayed in the page.

Figure 4–20 Selected Web Clipping Displayed in Web Clipping Portlet



Oracle Application Server Portal Developer□s Guide 10g (9.0.4) Part No. : Oracle Application Server Portal... (published 05/1
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/DOCTEAM/902DOC/904_DOC_BUILDING_PORTLETS/B1

Oracle9iAS Portal - Web Clipping:
http://www.oracle.com/technology/products/ias/tech_tips/march1003.html

OracleAS Portal Building Portlets with Web Clipping: OracleAS Portal Building Portlets with Web Clipping...
http://www.oracle.com/technology/products/ias/portal/webclipping.html

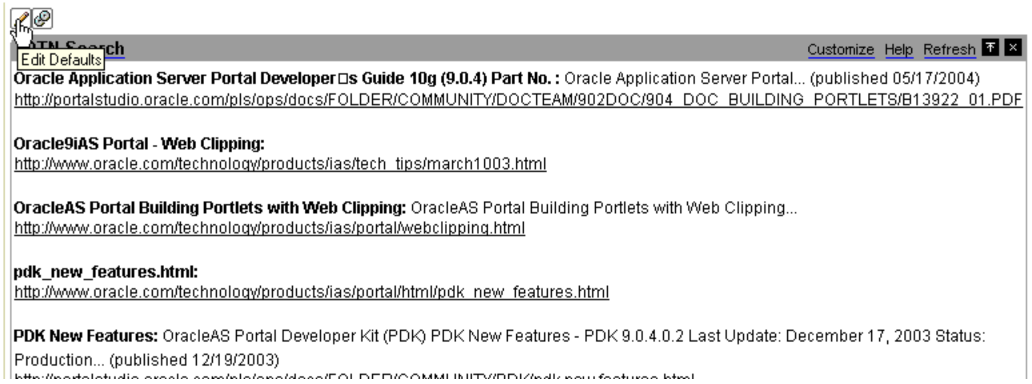
pdk_new_features.html:
http://www.oracle.com/technology/products/ias/portal/html/pdk_new_features.html

4.4.3 Exercise: Customizing a Web Clipping Portlet

In this exercise, you edit the properties of the Web Clipping portlet to allow end users to display a different product in the portlet:

1. Above the Web Clipping portlet you just added, click the **Edit Defaults** icon, as shown in Figure 4–21.

Figure 4–21 Clicking Edit Defaults for the Web Clipping Portlet



Oracle Application Server Portal Developer□s Guide 10g (9.0.4) Part No. : Oracle Application Server Portal... (published 05/17/2004)
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/DOCTEAM/902DOC/904_DOC_BUILDING_PORTLETS/B13922_01.PDF

Oracle9iAS Portal - Web Clipping:
http://www.oracle.com/technology/products/ias/tech_tips/march1003.html

OracleAS Portal Building Portlets with Web Clipping: OracleAS Portal Building Portlets with Web Clipping...
http://www.oracle.com/technology/products/ias/portal/webclipping.html

pdk_new_features.html:
http://www.oracle.com/technology/products/ias/portal/html/pdk_new_features.html

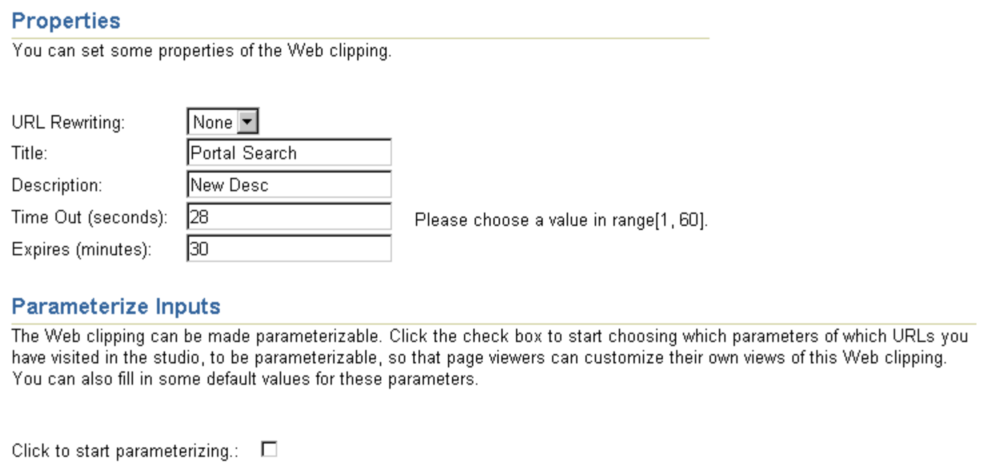
PDK New Features: OracleAS Portal Developer Kit (PDK) PDK New Features - PDK 9.0.4.0.2 Last Update: December 17, 2003 Status: Production... (published 12/19/2003)

2. In the Find a Web Clipping page, modify the following items in the **Properties** section:

   - From the **URL Rewriting** list, choose **Inline** to specify that you want link targets displayed inside the portlet, rather than in a new browser window.
   - In the **Title** field, enter **Portal Search.** This title displays in the banner of your Web Clipping portlet, as well as the pages where users can customize parameters for the Web clipping.

   Figure 4–22 shows the **Properties** and **Parameterize Inputs** sections of the Find a Web Clipping page.

Figure 4–22 Setting Properties for a Web Clipping


Properties

You can set some properties of the Web clipping.

URL Rewriting:	None	
Title:	Portal Search	
Description:	New Desc	
Time Out (seconds):	28	Please choose a value in range[1, 60].
Expires (minutes):	30	

Parameterize Inputs

The Web clipping can be made parameterizable. Click the check box to start choosing which parameters of which URLs you have visited in the studio, to be parameterizable, so that page viewers can customize their own views of this Web clipping. You can also fill in some default values for these parameters.

Click to start parameterizing.: ☐

3. Because the content displayed in the portlet was reached by entering information in the **Search** field on OTN, you can customize the parameters used by the search to allow end users to specify their own search string.

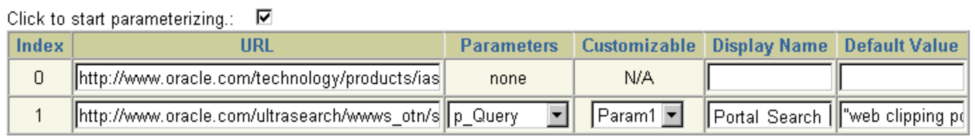
   In the **Parameterize Inputs** section, select the **Click to start parameterizing** check box.

4. On the last line of the parameters table, make the following changes:

   - In the **Parameters** column, choose **p_Query** from the list.
   - In the **Customizable** column, choose **Param1** from the list.
   - In the **Display Name** column, enter **Portal Search.**
   - Make sure that **Default Value** displays **"web clipping portlet"** to be sure you have selected the right parameter.

   Figure 4–23 shows the parameters table.

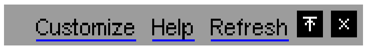
Figure 4–23 Specifying Parameters for User Input


Click to start parameterizing.: ☑

Index	URL	Parameters	Customizable	Display Name	Default Value
0	http://www.oracle.com/technology/products/ias	none	N/A		
1	http://www.oracle.com/ultrasearch/wwws_otn/s	p_Query	Param1	Portal Search	"web clipping p

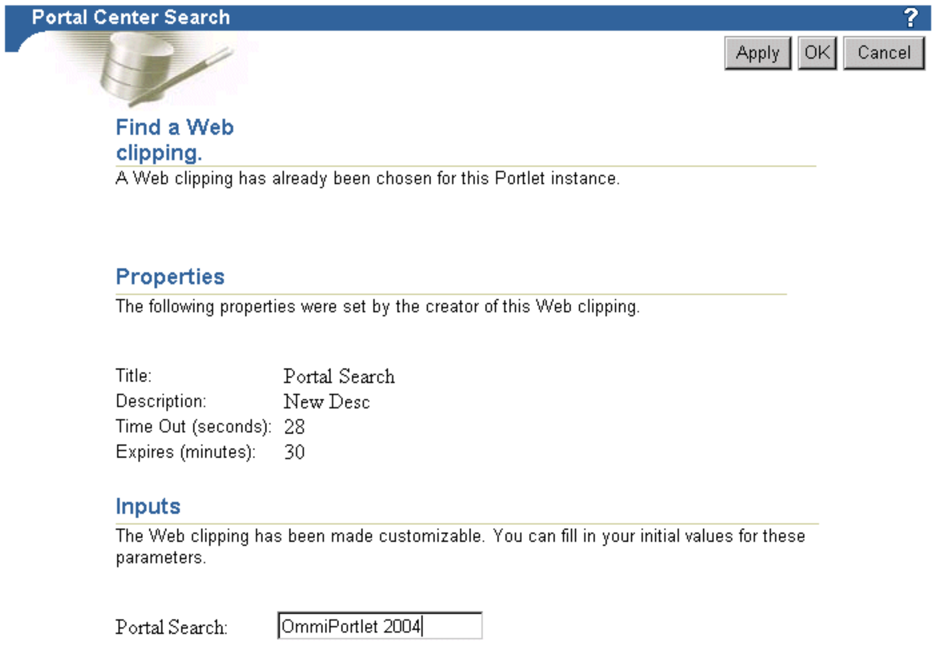
5. Click **OK** to display the default search results in the Web Clipping portlet on your page.
6. In the Web Clipping portlet banner, click **Customize**, as shown in Figure 4–24.

Figure 4–24 Clicking Customize in the Web Clipping Portlet Banner



7. In the page that displays, scroll down to the **Inputs** section. Notice that the parameter field for the search string is labeled Portal Search, as you specified for the Display Name for this parameter. In the **Portal Search** field, enter a different search string. For example, enter **OmniPortlet 2004**, as shown in Figure 4–25.

Figure 4–25 Specifying Input for Parameters



Find a Web clipping.

A Web clipping has already been chosen for this Portlet instance.

Properties

The following properties were set by the creator of this Web clipping.

Title:	Portal Search
Description:	New Desc
Time Out (seconds):	28
Expires (minutes):	30

Inputs

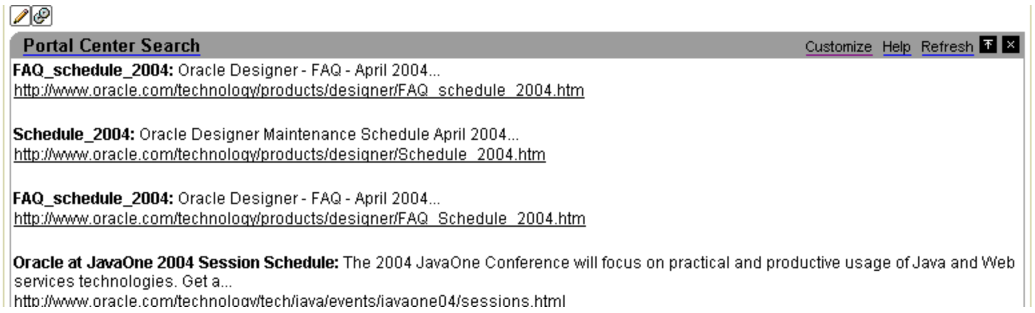
The Web clipping has been made customizable. You can fill in your initial values for these parameters.

Portal Search: OmmiPortlet 2004

8. Click **OK.**

   The Web Clipping portlet now displays the results of performing a search on OTN for OmniPortlet 2004 information, as shown in Figure 4–26.

Figure 4–26 New Web Clipping Result Based on Customer Input Parameter



FAQ_schedule_2004: Oracle Designer - FAQ - April 2004...
http://www.oracle.com/technology/products/designer/FAQ_schedule_2004.htm

Schedule_2004: Oracle Designer Maintenance Schedule April 2004...
http://www.oracle.com/technology/products/designer/Schedule_2004.htm

FAQ_schedule_2004: Oracle Designer - FAQ - April 2004...
http://www.oracle.com/technology/products/designer/FAQ_Schedule_2004.htm

Oracle at JavaOne 2004 Session Schedule: The 2004 JavaOne Conference will focus on practical and productive usage of Java and Web services technologies. Get a...
http://www.oracle.com/technology/tech/java/events/javaone04/sessions.html

4.5 Current Limitations for Web Clipping

This section describes current limitations for Web Clipping. For information about the latest features and limitations in a release, be sure to read the *Oracle Application Server Release Notes*.

- If the site to which you are connecting uses a lot of JavaScript to manipulate cookies or uses the JavaScript method `document.write` to modify the HTML document being written out, you may not be able to clip content from the site.
- URL-based portlets that have been migrated to Web Clipping do not support proxy authentication by default. This is because URL-based portlets inherently do not support proxy authentication and Web Clipping preserves the edit mode of the portlets. Their edit mode does not provide an opportunity to enter authentication information. To work around this restriction, add an empty Web Clipping portlet

to the same portal and use the Web Clipping portlet Customize link to enter the user name and password for proxy authentication.

- When you integrate with partner applications (through the use of mod_osso), you cannot clip directly through those partner applications in an authenticated manner. However, you can use the partner applications through the external application framework.
- You cannot use the Web Clipping Portlet to clip OracleAS Portal pages. As a workaround, examine the portlet that is supplying the data and take the appropriate action:
  - For database provider portlets, use export/import to copy pages across portals.
  - For Web provider portlets, re-register the same provider in the destination portal and edit the portal manually.

For troubleshooting information, see Appendix B, "Troubleshooting OracleAS Portal".

5 Building Java Portlets

This chapter explains how to create Java portlets based on the Java Portlet Specification, how to build Java portlets using Oracle Application Server Portal Developer Kit-Java (PDK-Java), and how to make a portlet out of your struts application:

- Guidelines for Creating Java Portlets
- Introduction to Java Portlet Specification and WSRP
  - The Relationship Between WSRP and JPS
- Building JPS-Compliant Portlets with Oracle JDeveloper
  - Installing the Oracle JDeveloper Portal Add-In
  - Building JPS-compliant Portlets
- Building PDK-Java Portlets with Oracle JDeveloper
  - Installing the Oracle JDeveloper Portal Add-In
  - Building PDK-Java Portlets
  - Adding Render Modes
  - Customizing Portlets
  - Passing Parameters and Submitting Events
  - Accessing Session Information
  - Implementing Portlet Security
  - Controlling the Export/Import of Portlet Customizations
  - Enhancing Portlet Performance with Caching
  - Writing Multi-Lingual Portlets
- Building Struts Portlets with Oracle JDeveloper
  - OracleAS Portal and the Apache Struts Framework
  - Creating a Struts Portlet

More On
Portal Center

The source code for many of the examples referenced in this chapter are available as part of PDK-Java. You can download PDK-Java from the Oracle Application Server Portal Developer Kit (PDK) page on OTN, `http://www.oracle.com/technology/products/ias/portal/pdk.html`.

When you unzip PDK-Java, you will find the examples in:

```
../pdk/jpdk/v2/src/oracle/portal/sample/v2/devguide
```

5.1 Guidelines for Creating Java Portlets

When you write your portlets in Java for either the Java Portlet Specification (JPS) or PDK-Java, you should follow the best practices described in this section. All of these guidelines pertain specifically to the Show modes of your portlet. A Show mode is an area of functionality provided by a portlet.

Note: JPS offers some modes not offered by OracleAS Portal and vice versa. If you are coding portlets to JPS, you can declare custom portlet modes that map to the extra modes offered by OracleAS Portal.

An OracleAS Portal portlet may have the following Show modes, each with its own visualization and behavior. JPS portlets can define custom portlet modes in `portlet.xml`. Defining custom modes is especially useful if the portlet must interoperate with portal implementations from other vendors.

- Shared Screen Mode (View Mode for JPS)
- Edit Mode (JPS and OracleAS Portal)
- Edit Defaults Mode (JPS and OracleAS Portal)
- Preview Mode (JPS and OracleAS Portal)
- Full Screen Mode (OracleAS Portal)
- Help Mode (JPS and OracleAS Portal)
- About Mode (JPS and OracleAS Portal)
- Link Mode (OracleAS Portal)

5.1.1 Shared Screen Mode (View Mode for JPS)

A portlet uses Shared Screen mode (known as View mode in JPS) to appear on a page with other portlets. This is the mode most people think about when they envision a portlet. Portlets are rendered inside HTML table cells when in Shared Screen mode. This means a portlet can display any content that can be rendered within a table cell, including, among other technologies, HTML, plug-ins, and Java applets. The actual size of the table cell is variable depending on user settings, the browser width, and the amount and style of content in the portlet. When developing portlets, remember that your portlet will share a page with others and you cannot completely control its dimensions and placement.

5.1.1.1 HTML Guidelines for Rendering Portlets

Plain HTML is the most basic way to render portlets and provides a great deal of flexibility to portlet developers. You can use almost any standard HTML paradigm, such as links, forms, images, tables, as long as it can display within an HTML table cell. Improperly written HTML may appear inconsistently across different browsers and, in the worst case, could cause parts of your page not to appear at all. Ensure that you adhere to the following rules:

- **Use standard HTML.** The official HTML specification is available from the W3C (more information available at: `http://www.w3.org/MarkUp/`).
- **Avoid unterminated and extraneous tags.** The behavior of pages with improperly terminated tags is unpredictable because it depends on what the browser chooses to do. Tools like weblint (`http://www.weblint.org/`) and HTML Tidy

(`http://www.w3.org/People/Raggett/tidy/`) can help detect and fix hanging and unnecessary tags.

- **Avoid elements that cannot be rendered properly in an HTML table cell.** Some constructs cannot be used simply because they do not display correctly in a table cell. Frames, for example, do not appear when inserted in a table.
- **Keep portlet content concise.** Do not try to take full screen content and expose it through a small portlet. You end up with portlet content too small or cramped for smaller monitors.
- **Do not create fixed-width HTML tables in portlets.** You have no way to tell how wide of a column your portlet will have on a user's page. If your portlet requires more room than given, it might overlap with another portlet in certain browsers.
- **Avoid long, unbroken lines of text.** The result is similar to what happens with wide fixed-width tables. Your portlet might overlap other portlets in certain browsers.
- **Check behavior when resizing the page.** Test your portlet's behavior when the browser window is resized to ensure that it works in different browser window sizes.
- **Check behavior when the default browser font changes.** People may choose whatever font size they wish and they can change it at any point in time. Your portlet should handle these situations gracefully.

The HTML you use also impacts the perceived performance of your site. Users judge performance based on how long it takes for them to see the page they requested, and browsers require time to interpret and display HTML. Given that, you should:

- **Avoid deeply nested tables.** Deeply nested tables slow performance dramatically in some older browser versions. OracleAS Portal draws several levels of tables to render portlets. If your portlets use tables within tables, your users may have to wait quite a while for those pages to render.
- **Avoid lengthy, complex HTML.** Portlets share a page with other portlets. Thus, portlet generation times can significantly effect the overall performance of the page. If portlets must render complex HTML or wait for external resources, such as third party applications, it can greatly slow the rendering of the page.

5.1.1.2 Cascading Style Sheet Guidelines for Rendering Portlets

The fonts and colors of every portlet on a page should match the style settings chosen by the user. To accomplish this goal, these style selections are embedded automatically using a Cascading Style Sheet (CSS) on each OracleAS Portal page. The portlets access these settings for their fonts and colors, either directly or using the API.

While different browsers have implemented varying levels of the full CSS specification, OracleAS Portal uses a very basic subset of this specification to allow for consistent fonts and colors. CSS implementation levels should not affect the consistency of your pages across browsers. Follow these guidelines for using CSS:

- **Use CSS instead of hard coding.** Hard coding fonts and colors is extremely dangerous. If you hard code fonts and colors, your portlet may look out of place when the user changes the page style settings. Since you have no way of knowing the user's font and color preference choices, you might also choose to hard code a font color that turns out to be the same as the user's chosen background color, in which case your portlet appears to be invisible to that user.
- **Use the CSS APIs to format your text.** The stylesheet definition is available at the top of OracleAS Portal pages, but you should not call it directly. Instead, use the

APIs provided to format your text appropriately. This method ensures that your portlets work even if the stylesheet changes in the future.

- **Avoid using CSS for absolute positioning.** Since users can customize their portal pages, you cannot guarantee that your portlet can appear in a particular spot.

5.1.2 Edit Mode (JPS and OracleAS Portal)

A portlet uses Edit mode to allow users to customize the behavior of the portlet. Edit mode provides a list of settings that the user can change. These customizable settings may include the title, type of content, formatting, amount of information, defaults for form elements, and anything that affects the appearance or content of the portlet.

Portal users typically access a portlet's Edit mode by clicking **Customize** on the portlet banner. When you click **Customize**, a new page appears in the same browser window. The portlet typically creates a Web page representing a dialog box to choose the portlet's settings. Once you apply the settings, you automatically return to the original page.

5.1.2.1 Guidelines for Edit Mode Options

The following guidelines should govern what you expose to users in Edit mode:

- **Allow users to customize the title of the portlet.** The same portlet may be added to the same portal page several times. Allowing the user to customize the title helps alleviate confusion.
- **If using caching, invalidate the content.** If customizations cause a change in portlet display or content, you must ensure that the portlet content is regenerated and not returned from the cache. Otherwise, the user may see incorrect content.
- **Do not use Edit mode as an administrative tool.** Edit mode is meant to give users a way of changing the behavior of their portlets. If you need to change provider settings or do other administrative tasks, you should create secured portlets specifically for those tasks.

5.1.2.2 Guidelines for Buttons in Edit Mode

For consistency and user convenience, Edit mode should implement the following buttons in the following order:

- **OK** saves the user customizations and redirects the browser back to the calling portal page
- **Apply** saves the user customizations and reloads the current page.
- **Cancel** redirects the browser to the calling portal page without saving changes.

5.1.2.3 Guidelines for Rendering Customization Values

When you show the forms used to change customization settings, you should default the values such that the user does not have to constantly re-enter settings. When rendering the customization values, follow a specific sequence to provide consistent behavior. Following is the sequence for rendering customization values:

1. **User preference:** Query and display this user's customizations, if available.
2. **Instance defaults:** If no user customizations are found, query and display system defaults for the portlet instance. These are set in Edit Defaults mode and only apply to this portlet instance.

3. **Portlet defaults:** If no system default customizations are found, display general portlet defaults, which may be blank. General portlet defaults are sometimes hard coded into the portlet but should be overridden if either of the two previous conditions apply.

This logic allows the customizations to be presented in a predictable way, consistent with the other portlets in the portal. PDK-Java makes this type of logic easy to implement.

5.1.3 Edit Defaults Mode (JPS and OracleAS Portal)

A portlet uses the Edit Defaults mode to allow page designers to customize the default behavior of a particular portlet instance. Edit Defaults mode provides a list of settings that the page designer can change. These customizable settings may include the title, type of content, formatting, amount of information, defaults for form elements, and anything that affects the appearance or content of the portlet.

These default customization settings can change the appearance and content of that individual portlet for all users. Because Edit Defaults mode defines the system level defaults for what a portlet displays and how it displays it, this mode should not be used as an administrative tool or for managing other portlets.

Page designers access Edit Defaults mode from the Customize Page when they choose **Customize for Others**. The link is labeled **Edit Defaults** on the banner of the wire frame diagram of that portlet.

When you click **Edit Defaults**, a new page appears in the same browser window. The portlet typically creates a Web page representing a dialog box to customize the portlet instance settings. Once the settings are applied, you are automatically returned to the original page.

5.1.3.1 Guidelines for Edit Defaults Mode Options

The following guidelines should govern what you expose to page designers in Edit Defaults mode:

- **If using caching, invalidate the cache.** If customizations cause a change in portlet display or content, you must ensure that the portlet content is regenerated and not returned from the cache. Otherwise, the user may see incorrect content.
- **Do not use Edit Defaults mode as an administrative tool.** Edit Defaults mode gives users a way of changing the behavior of their portlets. If you need to change provider settings or do other administrative tasks, you should create secured portlets specifically for those tasks.

5.1.3.2 Guidelines for Buttons in Edit Defaults Mode

For consistency and user convenience, Edit mode should implement the following buttons in the following order:

- **OK** saves the user customizations and redirects the browser back to the calling portal page.
- **Apply** saves the user customizations and reloads the current page.
- **Cancel** redirects the browser to the calling portal page without saving changes.

5.1.3.3 Guidelines for Rendering Customization Values

When you show the forms used to change customization settings, you should default the values so that the page designer does not have to constantly re-enter settings.

When rendering the customization values, follow a specific sequence to provide consistent behavior. Following is the sequence for rendering customization values:

1. **Instance preferences:** Query and display system defaults for the portlet instance.
2. **Portlet defaults**: If no system default customizations are found, display general portlet defaults, which may be blank. General portlet defaults are sometimes hard coded into the portlet but should be overridden by system defaults.

This logic allows the customizations to be presented in a predictable way, consistent with the other portlets in the portal.

5.1.4 Preview Mode (JPS and OracleAS Portal)

A portlet uses Preview mode to show the user what the portlet looks like before adding it to a page. Preview mode visually represents what the portlet can do.

Portal users typically access a portlet's Preview mode by clicking on its Preview icon from the Add Portlet page. A window then displays the preview of the chosen portlet. The user then has the option to add that portlet to the page. Portal administrators may access Preview mode from the Portlet Repository.

Note that the portal does not draw the portal banner when rendering the portlet in this mode.

5.1.4.1 Guidelines for Preview Mode

The following guidelines should govern what you expose to users in Preview mode:

- **Provide an idea of what the portlet does.** Preview mode should generate enough content for the user to get an idea of the actual content and functionality of the portlet.
- **Keep your portlet previews small.** The amount of data produced in this mode should not exceed a few lines of HTML or a screen shot. Preview mode appears in a small area and exceeding the window's size looks unprofessional and forces users to scroll.
- **Do not use live hyperlinks.** Links may not behave as expected when rendered on the Add Portlet page or the Portlet Repository. Hyperlinks can be simulated using the underline font.
- **Do not use active form buttons.** Forms may not behave as you expect them to when rendered on the Add Portlet page or the Portlet Repository. If you decide to render form elements, do not link them to anything.

5.1.5 Full Screen Mode (OracleAS Portal)

Portlets use Full Screen mode to show more details than possible when sharing a page with other portlets. Full Screen mode lets a portlet have the entire window to itself.

For example, if a portlet displays expense information, it could show a summary of the top ten spenders in Shared Screen mode and the spending totals for everyone in Full Screen mode. Portlets can also provide a shortcut to Web applications. If a portlet provided an interface to submitting receipts for expenses in Shared Screen mode, it could link to the entire expense application from Full Screen mode.

Portal users access a portlet's Full Screen mode by clicking the title of the portlet.

Technically, JPS portlets do not have Full Screen mode. However, you can implement the equivalent of Full Screen mode for a JPS portlet with View mode (Shared Screen mode) and a maximized state for the window.

5.1.5.1 Guidelines for Full Screen Mode

The following guidelines should govern what you expose to users in Full Screen mode:

- **Avoid elements that cannot be rendered properly in an HTML table cell.** Even though Full Screen mode has a window to itself, the portlet is still formatted in an HTML table cell. Hence, you should avoid constructs that do not display correctly in a table cell. Frames, for example, do not appear when inserted in a table.
- **Provide a way to navigate to the previous page.** You should provide users with a link or button that takes them back to the original portal page.

5.1.6 Help Mode (JPS and OracleAS Portal)

A portlet uses Help mode to display information about the functionality of the portlet and how to use it. The user should be able to find useful information about the portlet, its content, and its capabilities with this mode.

Portal users access a portlet's Help mode by clicking **Help** in the portlet banner.

5.1.6.1 Guidelines for Help Mode

The following guidelines should govern what you expose to users in Help mode:

- **Describe how to use the portlet.** Users may not know all the features your portlet provides just from its interface. Describe the features and how to get the most out of them.

5.1.7 About Mode (JPS and OracleAS Portal)

Users should be able to see what version of the portlet is currently running, its publication and copyright information, and how to contact the author. Portlets that require registration may link to Web-based applications or contact information from this mode, as well.

Portal users access a portlet's About mode by clicking **About** on the portlet banner. A new page appears in the same browser window. The portlet can either generate the content for this new page or take the user to an existing page or application.

5.1.7.1 Guidelines for About Mode

The following guidelines should govern what you expose to users in About mode:

- **Display relevant copyright, version, and author information.** Users want to know what portlet they are using and where they can get more information. The about page may become important when supporting your portlets.

5.1.8 Link Mode (OracleAS Portal)

A portlet uses Link mode to render a link to itself that displays on a mobile page. When the user clicks the link, the portlet is called in Show mode but with a different content type.

For JPS portlets that declare support of the Oracle Mobile XML content type, OracleAS Portal renders the link in one of two ways:

- Call the portlet's View mode with the MINIMIZED window state, if the portlet declares support for it.
- Otherwise, render a link using the portlet’s title.

5.1.8.1 Guidelines for Link Mode

The following guidelines should govern what you expose to users in Link mode:

- **Limit content.** Users should see the same data as Shared Screen mode but without all of the graphic images and unnecessary content that would be difficult or impossible to view effectively on the tiny screen of a mobile device.

5.2 Introduction to Java Portlet Specification and WSRP

Organizations engaged in enterprise portal projects have found application integration to be a major issue. Until now, users developed portlets using proprietary APIs for a single portal platform and often faced a shortage of available portlets from a particular portal vendor. All this changes with the introduction of the following standards:

- Web Services for Remote Portlets (WSRP)
- Java Portlet Specification (JPS)[1] based on JSR 168

These two standards enable the development of portlets that interoperate with different portal products, and therefore widen the availability of portlets within an organization. This wider availability can, in turn, dramatically increase an organization's productivity when building enterprise portals.

WSRP is a Web services standard that allows the plug-and-play of visual, user-facing Web services with portals or other intermediary Web applications. Being a standard, WSRP enables interoperability between a standards enabled container and any WSRP portal. WSRP defines:

- Web Services Definition Language (WSDL) interface for the invocation of WSRP services
- Markup fragment rules for markup emitted by WSRP services
- The method to publish, find, and bind WSRP services and metadata

JPS is a specification that defines a set of APIs to enable interoperability between portlets and portals, addressing the areas of aggregation, personalization, presentation, and security. JPS defines container services which provide:

- A portlet API for coding portlet functionality
- The URL-rewriting mechanism for creating user interaction within a portlet container
- The security and personalization of portlets

Oracle actively participates in the WSRP committee and is also a member of the expert group for JPS. Oracle is committed to supporting these standards and is working on a production release of a WSRP-enabled portal. Today, Oracle Technology Network (OTN) members can:

- View a hosted, pre-release version of the WSRP portal. On this hosted portal, users can also view a set of WSRP sample portlets, register a provider (also known as a producer), and add portlets to a page.
- View a developer’s preview version of the WSRP portal.
- View a preview release of Oracle’s JPS-compliant portlet container, which exposes Java portlets as WSRP services.
- Verify the interoperability of their WSRP-enabled portlets.

[1] The Java Portlet Specification 1.0 arose from Java Specification Request 168 and the JSR168 Expert Group.

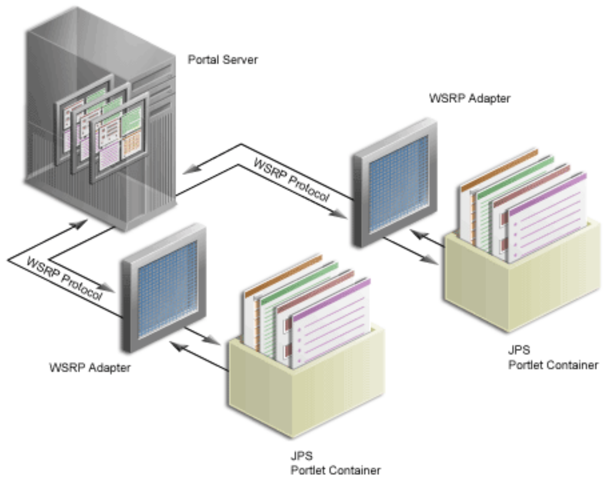
- Download tools to build interoperable portlets based on these standards.

5.2.1 The Relationship Between WSRP and JPS

WSRP is a communication protocol between portal servers and portlet containers, while JPS describes the Java Portlet API for building portlets. Combining these standards enables developers to integrate their applications from any internal or external source as portlets with WSRP portals. Building portal pages becomes as simple as selecting portlets from the OracleAS Portal repository. Figure 5–1 shows the architecture of the WSRP specification.

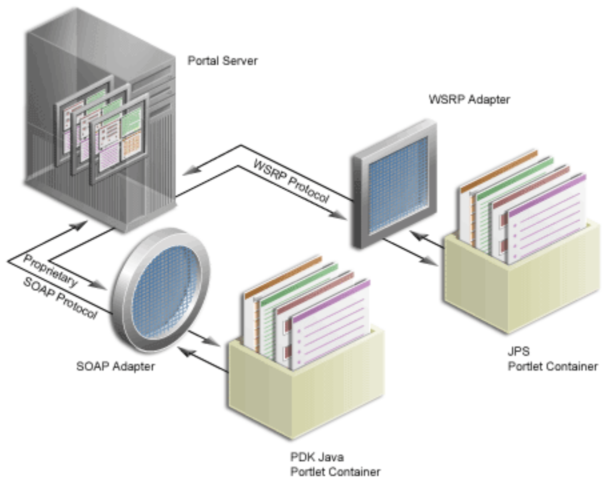
Note: Figure 5–1 illustrates the use of JPS portlets with WSRP, but it should be noted that WSRP can also work with non-JPS portlets.

Figure 5–1 WSRP Specification Architecture



Since OracleAS Portal's existing architecture is so similar to the one specified by the WSRP committee, OracleAS Portal is able to support communication between our portal and both the new Java Portlet APIs as well as our existing APIs (PDK-Java). Figure 5–2 shows the architecture of the WSRP portal. Notice that the JPS-compliant portlet container uses the WSRP protocol for communication and the PDK-Java portlet container uses Oracle's proprietary protocol for communication.

Figure 5–2 OracleAS Portal's WSRP Architecture



5.3 Building JPS-Compliant Portlets with Oracle JDeveloper

Using a convenient download from OTN, you can add extensions to Oracle JDeveloper for building portlets. To use this extension, you perform the steps in the following procedures:

- Installing the Oracle JDeveloper Portal Add-In
- Building JPS-compliant Portlets

Note: OracleAS Portal is not WSRP-enabled in Release 9.0.4. You can, however, build and test your standards-based portlets today. To build standards-based portlets, use Oracle JDeveloper with the OracleAS Portal Add-In as described Section 5.3.2, "Building JPS-compliant Portlets". To test your standards-based portlets against a WSRP-enabled OracleAS Portal, you must use the Portlet Standards Developer's Preview release or `portalstandards.oracle.com`.

This section assumes the following:

- You are familiar with portlet terminology such as portlet Show modes. Refer to Chapter 1, "Understanding Portlets" and Section 5.1, "Guidelines for Creating Java Portlets".
- You have already downloaded and installed the Java Portlet Container and have an Oracle Application Server Containers for J2EE 9.0.4 container to which you may deploy your portlets.
- You are already familiar with Oracle JDeveloper and know how to build and deploy Java components using it. You can download Oracle JDeveloper from OTN.

5.3.1 Installing the Oracle JDeveloper Portal Add-In

The OracleAS Portal Developer Kit (PDK) provides you with the necessary libraries to install an add-in for Oracle JDeveloper that dramatically increases your flexibility and productivity when developing portlets. This extension includes two wizards, one for building JPS-compliant portlets and one for building PDK-Java portlets. Both wizards guide you through the steps of creating the portlet skeleton and all you need do then is implement your own business logic.

To obtain the add-in:

1. Visit `http://www.oracle.com/technology/products/ias/portal/index.html`.
2. On the left side of the page, click **Integration/Utilities**.
3. Click **Portal Add-In for Oracle JDeveloper** to download portal-addin.zip.
4. Click **Install Instructions** and read and follow the instructions.

Note: You can also just go to `http://www.oracle.com/technology/products/ias/portal/index.html` and search for `portal-addin.zip`.

5.3.2 Building JPS-compliant Portlets

Once you have successfully installed the Portlet Wizard for Java, you can begin your interoperability portlet development quickly and easily with Oracle JDeveloper:

- Creating a Portlet: Use the wizard to create your basic portlet code and the necessary configuration files for the framework.
- Adding Portlet Logic: Extend the example code with your own business logic.
- Deploying Your Portlet to an Application Server: Use Oracle JDeveloper to deploy your application to your application server.
- Registering and Viewing Your Portlet: Register and view your portlet with your local OracleAS Portal instance or, if you do not have a local instance, http://portalstandards.oracle.com.

5.3.2.1 Creating a Portlet

This section walks you through the Portlet Wizard. You can choose which portlet Show modes you want to implement and the implementation method (JSP, HTTP servlet, Java class, or HTML). The wizard then creates a simple sample implementation for each of the selected modes.

1. After you open Oracle JDeveloper, click the workspace where you want to create this project. If you do not have a workspace, you can create one as follows:
   a. Right-click the **Applications** node in the Applications - Navigator and choose **New** from the context menu.
   b. Choose **Workspace** from the Items list in the New Gallery dialog box.
   c. Click **OK**.
   d. Enter a **Workspace Name** and **Directory Name**, and clear **Add a New Empty Project** in the Create Workspace dialog box.
   e. Click **OK**.
2. Right-click the name of the workspace in the Applications - Navigator and choose **New Project** from the context menu.
3. Click **Empty Project** in the Items list in the New Gallery dialog box.
4. Click **OK**.

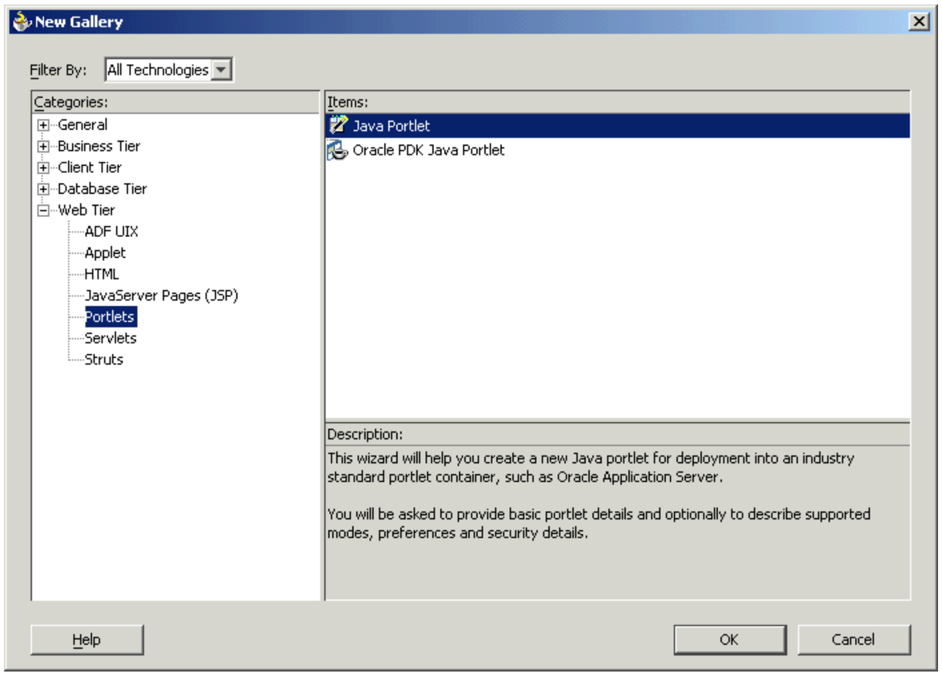
5. Enter the **Project Name** and **Directory Name** in the Create Project dialog box.
6. Click **OK**.
7. Right-click your project and select **New** from the context menu.
8. In the Categories list, expand the Web Tier category and click **Portlets**.

> **Note:** If you cannot find **Portlets**, refer to Section 5.3.1, "Installing the Oracle JDeveloper Portal Add-In" to ensure that you have installed the add-in correctly.

9. In the Items list, click **Java Portlet**.

> **Note:** Clicking **Java Portlet** opens the Portlet Wizard for creating JPS-compliant portlets. Clicking **Oracle PDK Java Portlet** opens the Portlet Wizard for creating PDK-Java portlets.

Figure 5–3 New Dialog Box

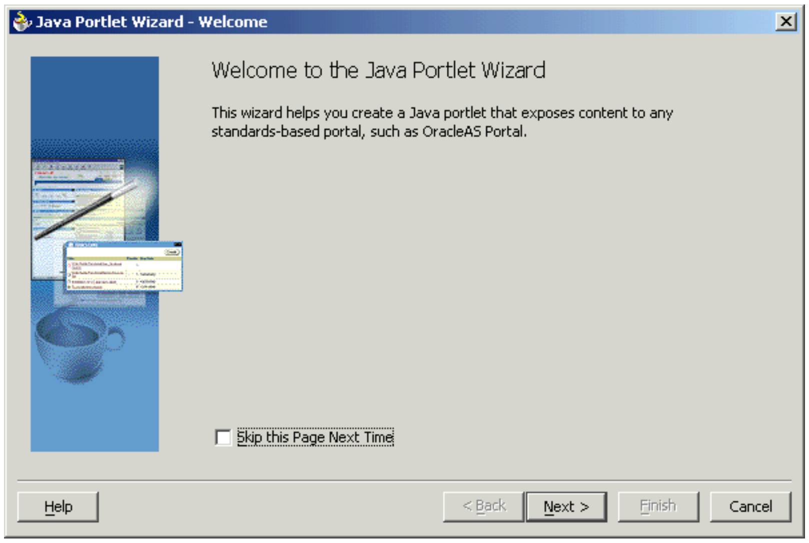


This wizard will help you create a new Java portlet for deployment into an industry standard portlet container, such as Oracle Application Server.

You will be asked to provide basic portlet details and optionally to describe supported modes, preferences and security details.

10. Click **OK**. The Portlet Wizard appears.
11. If you are on the Welcome page of the wizard, click **Next**.

Figure 5–4 Welcome Page



Welcome to the Java Portlet Wizard

This wizard helps you create a Java portlet that exposes content to any standards-based portal, such as OracleAS Portal.

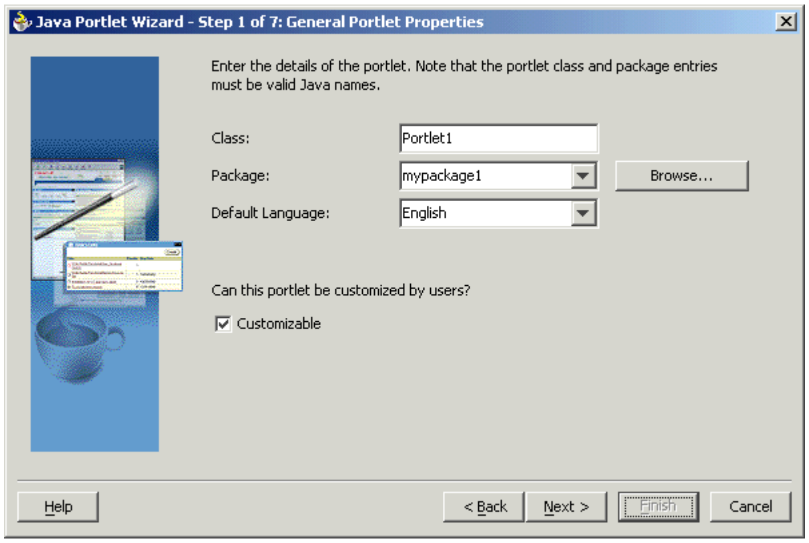
☐ Skip this Page Next Time

12. On the General Portlet Properties page, enter the values as described in Table 5–1 and shown in Figure 5–5.

Table 5–1 General Portlet Properties Values

Property	**Value**
Class	Enter the name of the class the wizard will create. The field is primed with a default class name that you may accept or change.
Package	Browse the packages in the project and select the one in which the class will reside. If you do not select a package, the wizard uses the default package of the project.
Default Language	Select the default language that your portlet will support. The wizard uses English by default.
Customizable	Select whether your portlet will be customizable by end users. The wizard allows customization by default.

Figure 5–5 General Portlet Properties Page


Java Portlet Wizard - Step 1 of 7: General Portlet Properties
Browse...
Help
< Back
Next >
Finish
Cancel

Enter the details of the portlet. Note that the portlet class and package entries must be valid Java names.

Class: Portlet1

Package: mypackage1

Default Language: English

Can this portlet be customized by users?

☑ Customizable

13. Click **Next**.

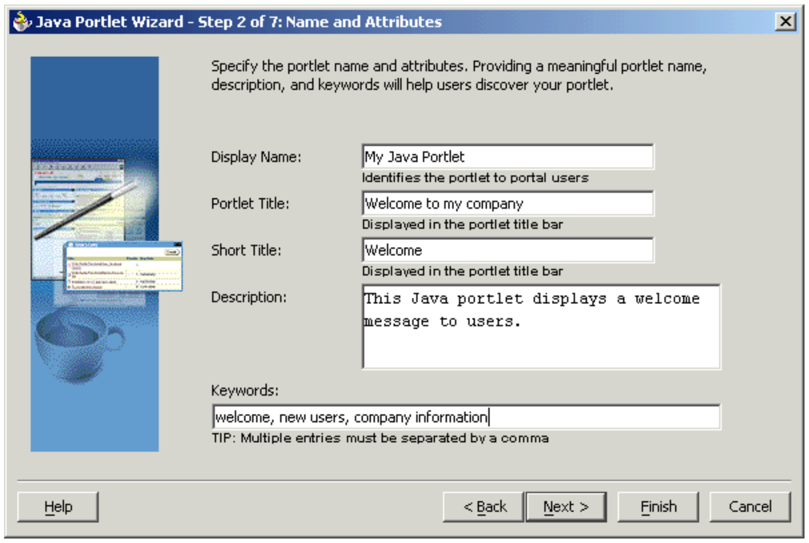
Note: The Finish button is not enabled until after the second step of the wizard has been completed.

14. On the Name and Attribution page, enter the values as described in Table 5–2 and shown in Figure 5–6.

Table 5–2 Name and Attribution Values

Property	Value
Display Name (required)	Enter the name that will be displayed in the OracleAS Portal catalog or repository. Enter `My Java Portlet` for this example.
Portlet Title (required)	Enter the title that will appear on the portlet header (on a portal page). Enter `Welcome to my company` for this example.
Short Title	Enter the title that will appear on the portlet header for mobile devices. Enter `Welcome` for this example.
Description	Enter a description of your portlet. Enter `This Java portlet displays a welcome message to users.` for this example.
Keywords	Enter keywords to help users find the portlet in the portal. Enter `welcome, new users, company information` for this example.

Figure 5–6 Name and Attribution Page



Specify the portlet name and attributes. Providing a meaningful portlet name, description, and keywords will help users discover your portlet.

Display Name: My Java Portlet
Identifies the portlet to portal users

Portlet Title: Welcome to my company
Displayed in the portlet title bar

Short Title: Welcome
Displayed in the portlet title bar

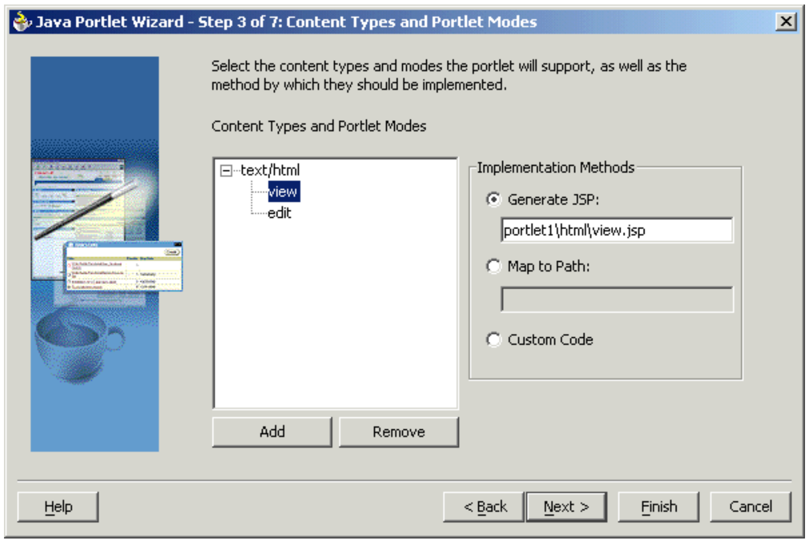
Description: This Java portlet displays a welcome message to users.

Keywords: welcome, new users, company information
TIP: Multiple entries must be separated by a comma

15. Click **Next** to continue specifying the portlet's properties or click **Finish**. If you click **Finish** at this point, the wizard chooses the default values for all remaining settings.

16. On the Content Types and Portlet Modes page shown in Figure 5–7, select the content types for your portlet and map the portlet modes to an implementation. By default, your portlet will display text/html as the content type, and a portlet mode of View. If **Customizable** was selected on the General Portlet Properties page, then edit also appears as a portlet mode for text/html.

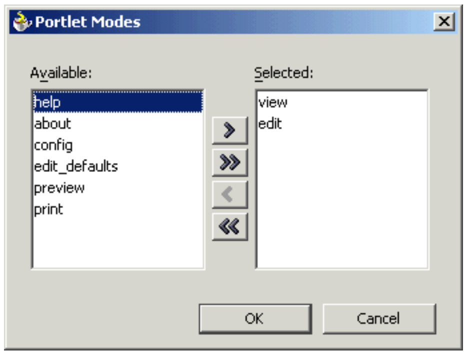
Figure 5–7 Content Types and Portlet Modes Page


Java Portlet Wizard - Step 3 of 7: Content Types and Portlet Modes
Select the content types and modes the portlet will support, as well as the method by which they should be implemented.
Content Types and Portlet Modes
text/html
view
edit
Implementation Methods
Generate JSP:
portlet1\html\view.jsp
Map to Path:
Custom Code
Add
Remove
Help
< Back
Next >
Finish
Cancel

a. If you need to add content types other than `text/html`, click `text/html` and then click **Add**. For the purposes of this example, you do not need to add any other content types.

b. If you need to add additional portlet modes, click an existing portlet mode (for example, view) and click **Add**. The list of available portlet modes displays, as shown in Figure 5–8, and you can add portlet modes by moving the desired portlet modes from the Available Modes list to the Selected Modes list. For the purposes of this example, you do not need to select any other portlet modes. When you are finished with the Portlet Modes dialog box, click **OK**.

Figure 5–8 Portlet Modes Dialog Box



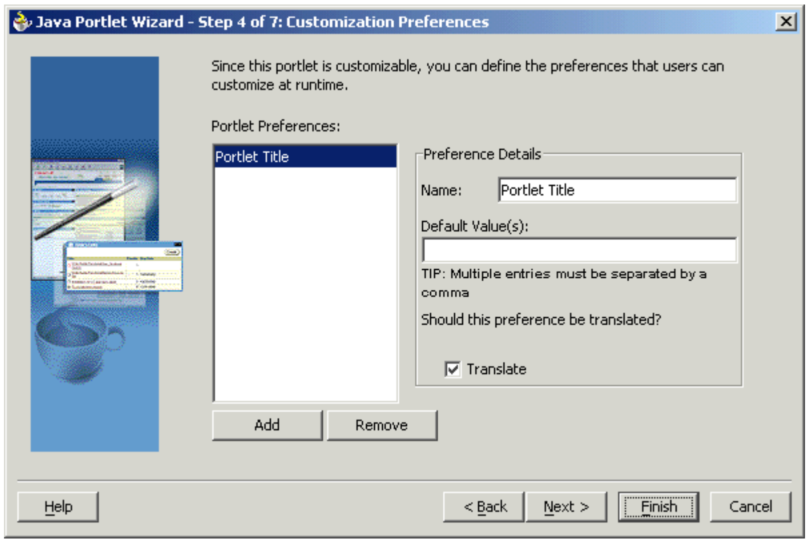
c. Once you have added all of the desired portlet modes, you need to choose the function to be performed for each mode. For each portlet mode, click the portlet mode and choose the desired function from the radio group on the right. For the purposes of this example, choose **Generate JSP** for both the edit and view portlet modes, and leave the default path and file name of the generated JSP. For more information on portlet modes, refer to Section 5.1, "Guidelines for Creating Java Portlets".

Note: Generate JSP and Custom Code generate code for you in the specified location. Map to Path routes the request through to an existing Web resource that you will create separately yourself.

---

17. Click **Next**.
18. If you selected **Customizable** on the General Portlet Properties page earlier in the wizard, the Customization Preferences page, shown in Figure 5–9, now displays and you can declare preferences for your portlet. If you did not select **Customizable** earlier, then this page is skipped. On the Customization Preferences page, you specify a preference name, default value, and whether the preference value should be translated:

Figure 5–9 Customization Preferences Page



Java Portlet Wizard - Step 4 of 7: Customization Preferences

Since this portlet is customizable, you can define the preferences that users can customize at runtime.

Portlet Preferences:

Portlet Title

Preference Details

Name: Portlet Title

Default Value(s):

TIP: Multiple entries must be separated by a comma

Should this preference be translated?

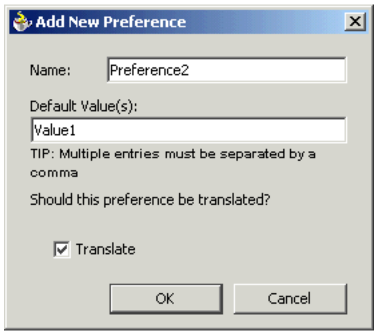
☑ Translate

   a. To add a preference, click **Add** and fill in the Add New Preference dialog box, show in Figure 5–10, (Name, Default Value(s), and whether it should be translated).

Note: The Name is always translated, but there is not always a need to translate the Default Value. For example, if the value is an integer, no translation is needed

---

Figure 5–10 Add New Preference Dialog Box



Add New Preference

Name: Preference2

Default Value(s):

Value1

TIP: Multiple entries must be separated by a comma

Should this preference be translated?

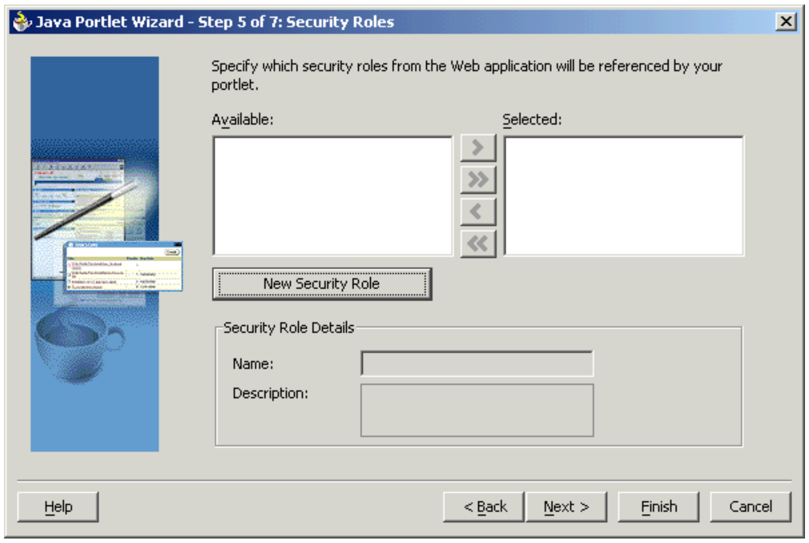
☑ Translate

  b. To delete a preference, choose it in the Portlet Preferences list and click **Remove**.

19. Click **Next**.

20. On the Security Roles page, shown in Figure 5–11, you can add security roles to your portlet. The wizard has no predefined security roles. It parses the `web.xml` for security roles and enables them to be referenced by your portlet. You will not need to do anything for this step as a new project has no security roles defined. You can manually create the security roles according to JPS later.

Figure 5–11 Security Roles Page



Specify which security roles from the Web application will be referenced by your portlet.

21. Click **Next**.

22. On the Caching page, you specify whether to enable caching of your portlet by default. The portlet itself may choose to cache content for any given response. The settings on this page come into force when the portlet itself does not specify a caching condition for a response. For the purposes of this example, fill out this page as described below and shown in Figure 5–12:

  a. Click **Cache Portlet**.

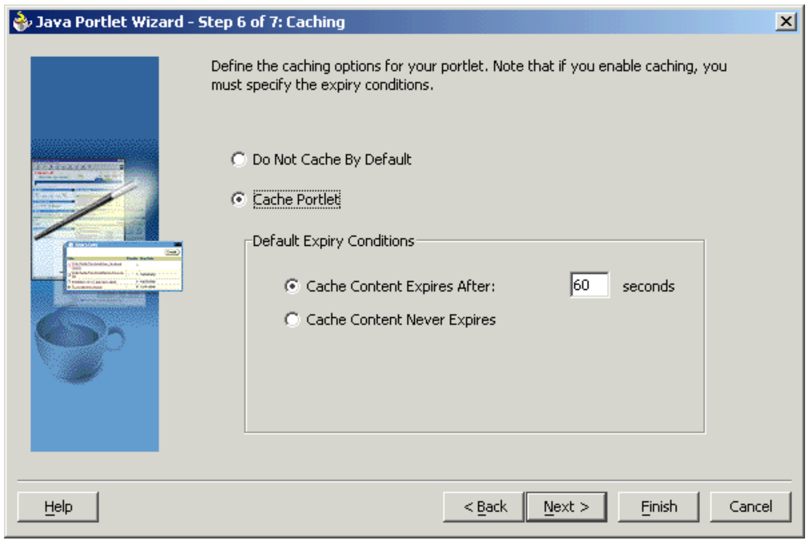
  b. Click **Cache Content Expires After**.

  c. Accept the default duration of the cached copy (`60` seconds).

Note: If you did not want any default caching for this portlet, you would choose **Do Not Cache By Default**. In this case, the wizard actually sets a cache duration of 0 seconds. As stated above, this cache setting only comes into play when the portlet itself does not specify a caching condition for a response.

If you choose no caching here and you later decide that you want default caching for the portlet, you can easily go back and change the cache duration value in the `portlet.xml` file, which is generated by the wizard, to some number greater than zero.

Figure 5–12 Caching Page


Java Portlet Wizard - Step 6 of 7: Caching

Define the caching options for your portlet. Note that if you enable caching, you must specify the expiry conditions.

Do Not Cache By Default [✓ Cache Portlet]

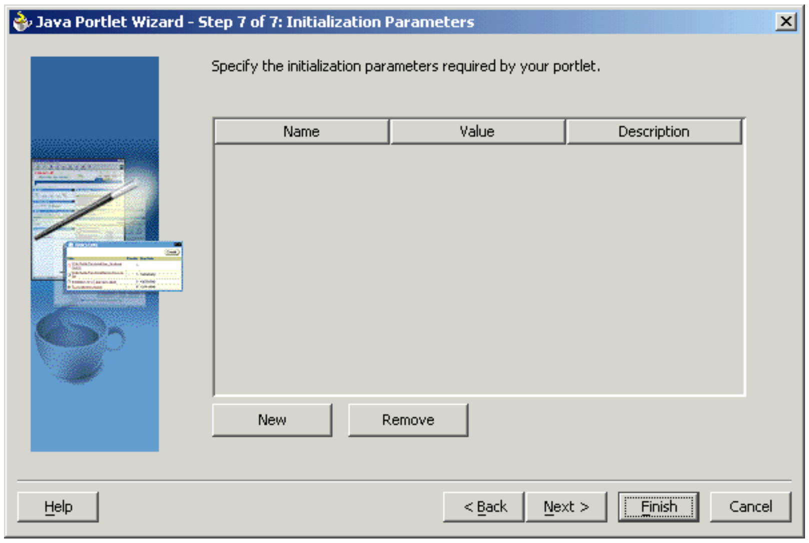
Default Expiry Conditions

[✓ Cache Content Expires After:] 60 seconds

Cache Content Never Expires

23. Click **Next**.
24. On the Initialization Parameters page, shown in Figure 5–13, you can add any required initialization parameters for the portlet. Initialization parameters provide the Web application developer, who decides what goes into the `.war` file, an alternative to JNDI variables for configuring the behavior of all of the different components of the Web application (for example, servlets and portlets) in a compatible way. For more information on initialization parameters, refer to the Java Portlet Specification. For the purposes of this example, no initialization parameters are needed.

Figure 5–13 Initialization Parameters Page


Java Portlet Wizard - Step 7 of 7: Initialization Parameters

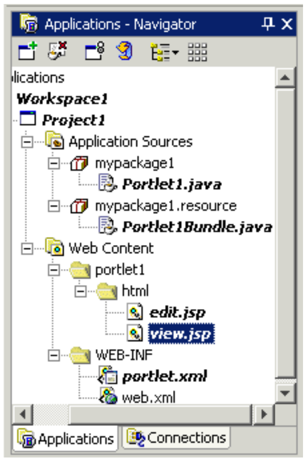
Specify the initialization parameters required by your portlet.

Name	Value	Description

25. Click **Next**.
26. Click **Finish** to generate the files for your portlet. In this case, the following files should be generated for your project node in the Applications - Navigator (see Figure 5–14):

- Generated code for each View mode, assuming that you selected **Generate JSP** on the Content Types and Portlet Modes page. If you selected **Custom Code** instead, that code will reside in the portlet's Java class.
- Two Java classes
- `portlet.xml`
- `web.xml`

Figure 5–14 Applications - Navigator



5.3.2.2 Adding Portlet Logic

Once you create the default implementation, you can extend the sample code with your own business logic to implement the desired functionality and features. Refer to the JavaDoc or JPS for more information on adding functionality and features. For the purposes of this example, you do not need to perform this step and can proceed directly to the deployment procedure.

5.3.2.3 Deploying Your Portlet to an Application Server

After you finish the wizard and successfully generate your portlet, you are ready to deploy it to an application server. Because you chose to create a JPS-compliant portlet, you can deploy it using the wizard for any vendor's JPS-compliant container. The following procedures describe how to deploy a JPS-compliant portlet to Oracle's WSRP container running on Oracle Application Server Containers for J2EE:

- Creating a Connection to Oracle Application Server Containers for J2EE
- Deploying the WAR File

5.3.2.3.1 Creating a Connection to Oracle Application Server Containers for J2EE To establish a connection to your application server, perform the following steps:

Note: The steps that follow describe the procedure for deploying to a standalone instance of Oracle Application Server Containers for J2EE. For information about deploying to a full Oracle Application Server instance, please refer to the Oracle JDeveloper online help system.

---

1. If it is not still open, start Oracle JDeveloper and open the project you created in the previous sections.
2. In the Navigator, click the **Connections** tab.
3. Right-click the **Connections** node and select **New Application Server Connection** from the context menu. Complete the wizard that displays as follows:
    a. If the Welcome page appears, click **Next**. If you do not want the Welcome page to appear in future, be sure to select **Skip this Page Next Time**.
    b. Enter a meaningful name for the connection (for example, `PDKStandardsOC4J`), and choose **Standalone OC4J** as the connection type.
    c. Click **Next**.
    d. Enter the administrator's user name and password. This password was set during the installation of the PDK Standards Oracle Application Server Containers for J2EE.
    e. Click **Next**.
    f. Enter the information in Table 5–3.

Table 5–3 Settings for New Application Server Connection

Setting	Value

URL	Enter the full RMI URL for this setting. For example:
	`ormi://my.machine.com:23791/`
	The RMI port number may be found in:
	`OC4J_HOME/j2ee/home/config/rmi.xml`

Target Web Site	Enter the name of the target Web site containing your deployed J2EE application files. For the purposes of this example, you can accept the default value, `http-web-site`.
Local Directory Where `admin.jar` for OC4J Is Installed	Enter the path to the local admin.jar that is version-compatible with the remote servers specified in the URL setting above. For JPS-compliant portlets built with Oracle JDeveloper 9.0.3, you need to change the default path in this setting to point to an instance of Oracle Application Server Containers for J2EE Release 9.0.4. For example: `OC4J_HOME\j2ee\home`

    g. Click **Next**.
    h. Verify the connection details by clicking **Test Connection**. A success message should appear if everything is correct. If the test fails, you may need to revise your connection information.
    i. Click **Finish**.

5.3.2.3.2 Deploying the WAR File To create and deploy a WAR file, perform the following steps:

1. Go to the **Applications - Navigator**.
2. In your current project, right-click `web.xml` and choose **Create WAR Deployment Profile** from the context menu.
3. In the Save Deployment Profile dialog box, change the name to something meaningful (for example, `jsrportlet1.deploy`).

4. Click **OK**.
5. In the Profile Settings dialog box, perform the following steps:
   a. Click **Specify J2EE Web Context Root** and enter `my-portlet` in the adjacent field.
   b. Click **OK**.
6. Right-click the deployment profile (for example, `jsrportlet1.deploy`) and choose **Deploy to >** the application server connection (for example, `PDKStandardsOC4J`) from the context menu.
7. Await the Deployment Finished message in the **Deployment Log** at the bottom of Oracle JDeveloper and verify that no errors occurred.
8. Take the URL provided in the log (for example, `http://myserver.com:8888/my-portlet`[2]) and append `/portlets?WSDL` to construct the URL you use to register your JPS-compliant portlet with OracleAS Portal. For example:

   ```
   http://myserver.com:8888/my-portlet/portlets?WSDL
   ```

   Note: In some cases, you may get a message in the log stating that Oracle JDeveloper was unable to determine the HTTP port number of the remote server. Typically, you can determine the port number yourself by looking at the URL that takes you to your Oracle Application Server Containers for J2EE home page (for example, `http://myserver.com:8888`).

5.3.2.4 Registering and Viewing Your Portlet

After you've created and deployed the provider and its portlet(s), you must register the provider with OracleAS Portal. Registering your provider gives OracleAS Portal the information it needs to locate and communicate with your provider. After you register a provider, the provider and its portlets become available in the Portlet Repository. They are also listed in the OracleAS Portal Navigator.

Note: When you build portlets and providers with built-in tools, such as the Portlet Builder, OracleAS Portal automatically registers the provider for you. Once you've created your portlet, it automatically displays in the Portlet Repository. OracleAS Portal also offers built-in portlets that are contained in a preconfigured provider. For example, OmniPortlet and Web Clipping are portlets that you can use out of the box, and are already registered with OracleAS Portal. You can view these portlets in the Add Portlets list. However, if you build the portlets and providers programmatically, you must then register these providers in order to make them available to the portal user.

This section describes how to register your provider and add your portlet to pages:

- Registering on a Local OracleAS Portal Instance
- Registering on portal.standards.com
- Adding Your Portlet

[2] The examples in this chapter typically use a port of 8888, which is the default port. Of course, you may choose to use different port numbers in your own installation.

5.3.2.4.1 Registering on a Local OracleAS Portal Instance

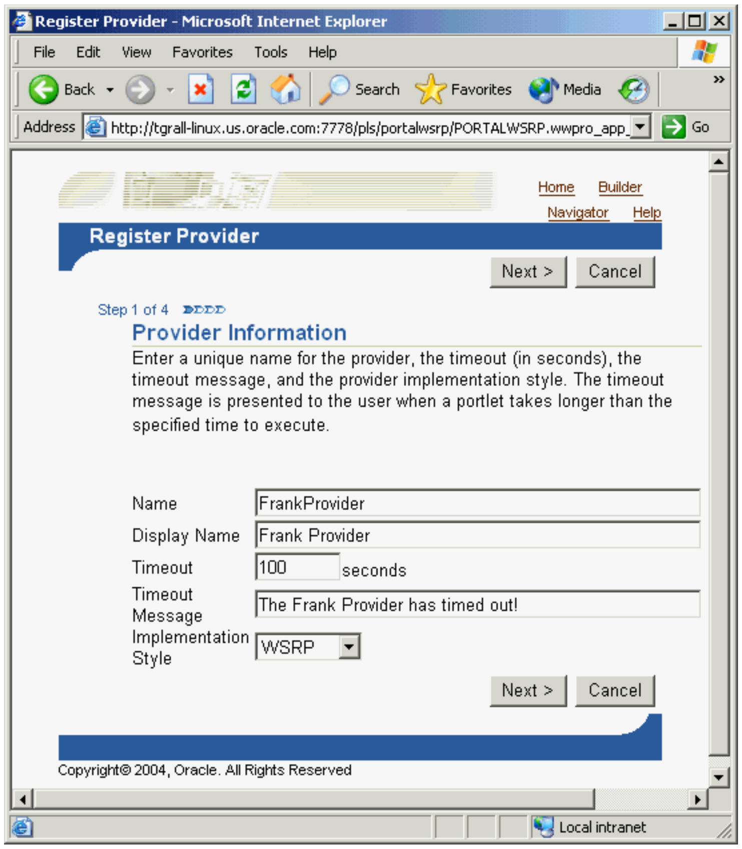
To register your standards-based portlets against a WSRP-enabled OracleAS Portal, you must have installed and configured the Portlet Standards Developer's Preview release. You can obtain this preview release from the Oracle Technology Network. Assuming you have a local instance of the WSRP-compliant OracleAS Portal, you can register your portlet as follows:

1. Open OracleAS Portal and log in as normal. Note that to register your provider, you need to have Manage or Edit privileges on providers. If you do not have these privileges, you need to request them from your administrator.
2. If you are not already on the Builder page, click **Builder** in the upper right corner.
3. Click the **Administer** tab.
4. Click the **Portlets** subtab.
5. In the Remote Providers portlet, click **Register a Provider**.
6. On the Register Provider page, shown in Figure 5–15, enter the values as described in Table 5–4.

Table 5–4 Register Provider Page Values

Setting	Value
Name	`*your_name*Provider`
Display Name	`*your_name* Provider`
Timeout	`100`
Timeout Message	`The *your_name* Provider has timed out!`
Implementation Style	`WSRP`

Figure 5–15 Register Provider Page


Register Provider - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://tgrall-linux.us.oracle.com:7778/pls/portalwsrp/PORTALWSRP.wwpro_app_ Go
Home Builder
Navigator Help
Register Provider
Next > Cancel
Step 1 of 4

Provider Information

Enter a unique name for the provider, the timeout (in seconds), the timeout message, and the provider implementation style. The timeout message is presented to the user when a portlet takes longer than the specified time to execute.

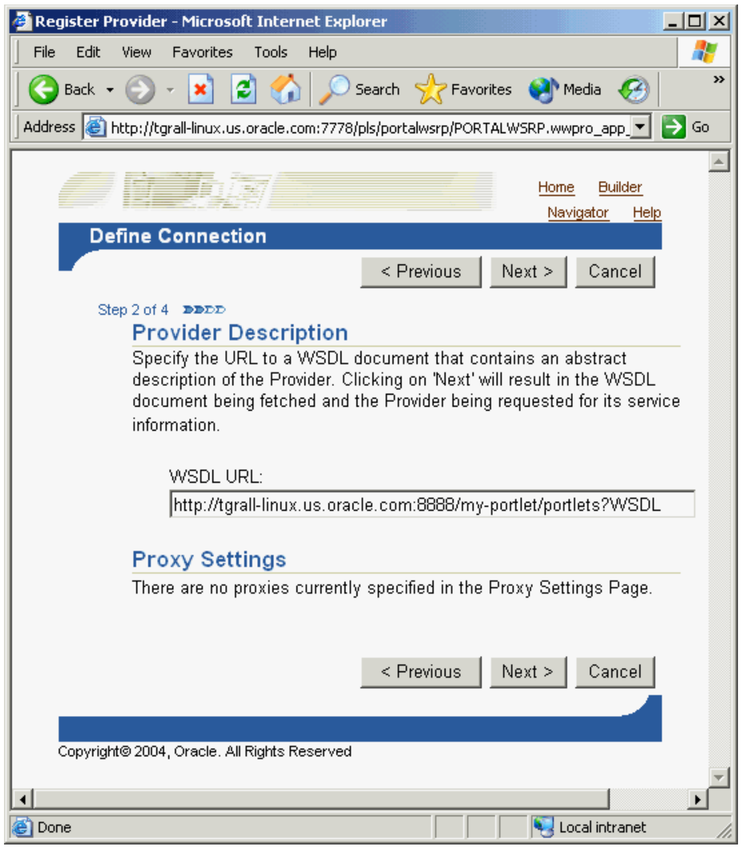
Name	FrankProvider
Display Name	Frank Provider
Timeout	100 seconds
Timeout Message	The Frank Provider has timed out!
Implementation Style	WSRP

7. Click **Next**.

8. On the Define Connection page, shown in Figure 5–16, enter the WSDL URL for your provider in the WSDL URL field. This URL is the one that you created in step 8 of Section 5.3.2.3.2, "Deploying the WAR File". For example:

```
http://myserver.com:8888/my-portlet/portlets?WSDL
```

Figure 5–16 Define Connection Page


Register Provider - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://tgrall-linux.us.oracle.com:7778/pls/portal/wsrp/PORTALWSRP.wwpro_app_
Go
Home Builder
Navigator Help
< Previous Next > Cancel
Done
Local intranet

Define Connection

Step 2 of 4

Provider Description

Specify the URL to a WSDL document that contains an abstract description of the Provider. Clicking on 'Next' will result in the WSDL document being fetched and the Provider being requested for its service information.

WSDL URL: http://tgrall-linux.us.oracle.com:8888/my-portlet/portlets?WSDL

Proxy Settings

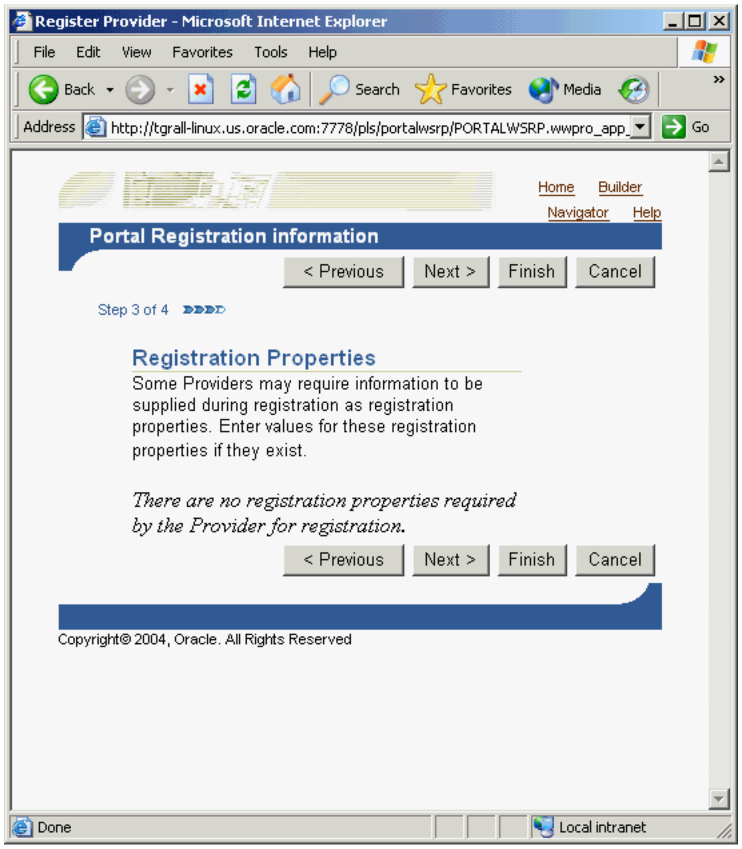
There are no proxies currently specified in the Proxy Settings Page.

Copyright© 2004, Oracle. All Rights Reserved

9. Click **Next**.

10. On the Portal Registration Information page, shown in Figure 5–17, you fill in any registration properties required by the provider. If there are none, you can proceed to the next step.

Figure 5–17 Portal Registration Information Page


Register Provider - Microsoft Internet Explorer

Address http://tgrall-linux.us.oracle.com:7778/pls/portalwsrp/PORTALWSRP.wwpro_app_

Portal Registration information

Step 3 of 4

Registration Properties

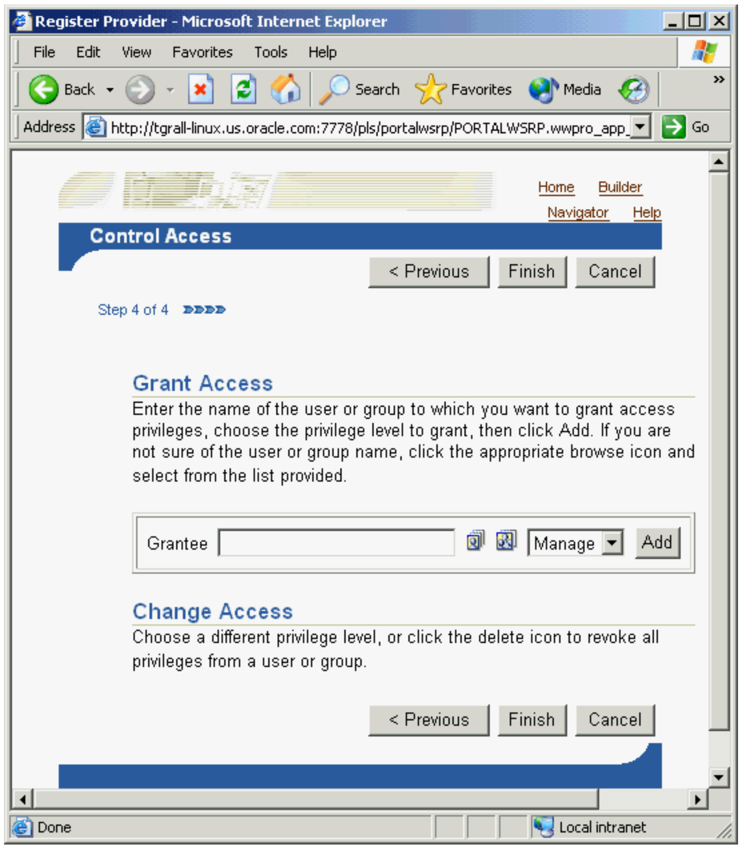
Some Providers may require information to be supplied during registration as registration properties. Enter values for these registration properties if they exist.

There are no registration properties required by the Provider for registration.

Copyright© 2004, Oracle. All Rights Reserved

11. Click **Next**.
12. On the Control Access page, shown in Figure 5–18, assign privileges for this provider as desired. For the purposes of this example, you can enter a group (for example, PORTAL_ADMINISTRATORS) for Grantee and click **Add**.

Figure 5–18 Control Access Page


Register Provider - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://tgrall-linux.us.oracle.com:7778/pls/portalwsrp/PORTALWSRP.wwpro_app_ Go
Home Builder
Navigator Help

Control Access

< Previous | Finish | Cancel

Step 4 of 4

Grant Access

Enter the name of the user or group to which you want to grant access privileges, choose the privilege level to grant, then click Add. If you are not sure of the user or group name, click the appropriate browse icon and select from the list provided.

Grantee [ ] Manage Add

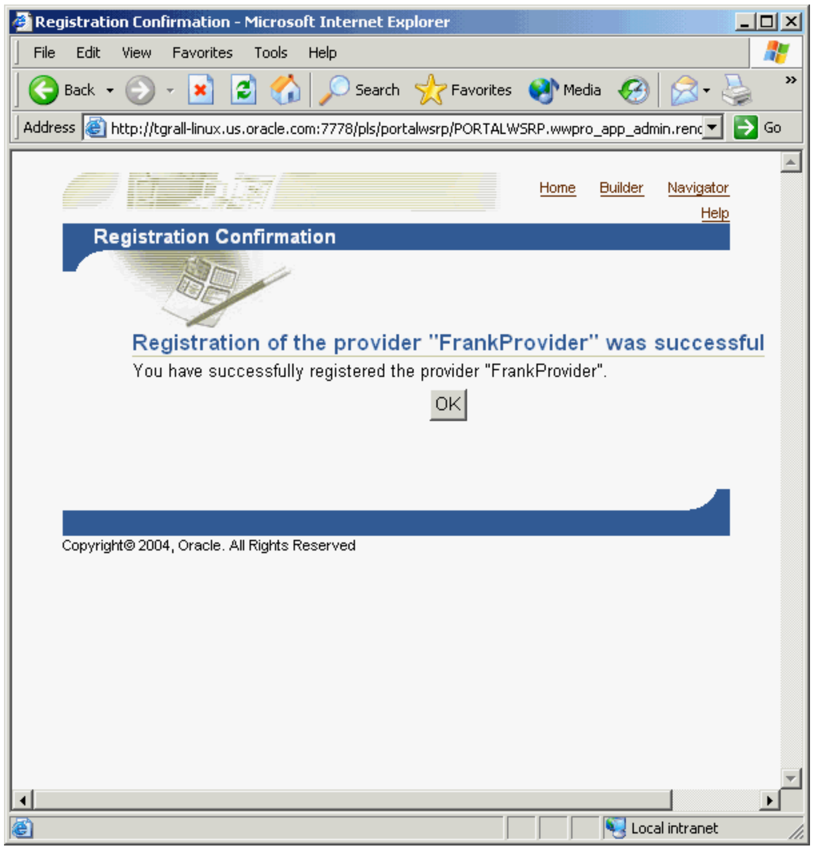
Change Access

Choose a different privilege level, or click the delete icon to revoke all privileges from a user or group.

< Previous | Finish | Cancel

13. Click **Finish**. You should see a Registration Confirmation page similar to the one in Figure 5–19.

Figure 5–19 Registration Confirmation Page


Registration Confirmation - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://tgrall-linux.us.oracle.com:7778/pls/portalwsrp/PORTALWSRP.wwpro_app_admin.renc
Go
Home Builder Navigator
Help
OK
Local intranet

Registration Confirmation

Registration of the provider "FrankProvider" was successful

You have successfully registered the provider "FrankProvider".

Copyright© 2004, Oracle. All Rights Reserved

5.3.2.4.2 Registering on portal.standards.com If you do not have a local instance of the WSRP-compliant OracleAS Portal, you can register your portlet as follows:

1. Go to the OracleAS Portal Verification service on OTN:

   ```
   http://portalstandards.oracle.com/
   ```

2. Follow the instructions in the document, *How to Test Interoperability*, located on OTN:

   ```
   http://portalstandards.oracle.com/pls/wsrp/docs/PAGE/WSRPPORTALPAGE/
   VERIFICATIONSERVER/TAB31866/HOW.TO.REGISTER.WSRP.PROVIDERS.HTML
   ```

5.3.2.4.3 Adding Your Portlet Your portlet should now be available for adding to pages like any other portlet in the Portlet Repository. To add your portlet to a page, follow the instructions in Section 7.6.2, "Adding Portlets," of the *Oracle Application Server Portal User's Guide*.

5.4 Building PDK-Java Portlets with Oracle JDeveloper

Using a convenient download from OTN, you can add extensions to Oracle JDeveloper for building portlets. To use this extension, you perform the steps in the following procedures:

- Installing the Oracle JDeveloper Portal Add-In
- Building PDK-Java Portlets

This section assumes the following:

- You are familiar with portlet terminology such as portlet Show modes. Refer to Chapter 1, "Understanding Portlets" and Section 5.1, "Guidelines for Creating Java Portlets".
- You have already downloaded and installed the Java Portlet Container and have an Oracle Application Server Containers for J2EE 9.0.4 container to which you may deploy your portlets.
- You are already familiar with Oracle JDeveloper and know how to build and deploy Java components using it. You can download Oracle JDeveloper from OTN.

5.4.1 Installing the Oracle JDeveloper Portal Add-In

For instructions on how to install the Oracle JDeveloper Portal add-in, refer to Section 5.3.1, "Installing the Oracle JDeveloper Portal Add-In".

5.4.2 Building PDK-Java Portlets

Once you have successfully installed the Oracle JDeveloper Portal add-in, you can begin your portlet development quickly and easily with Oracle JDeveloper:

- Creating a Portlet and Provider: Use the Portlet Wizard to create your basic portlet code and the necessary configuration files for the provider framework.
- Adding Portlet Logic: Extend the example code with your own business logic.
- Validating Your Portlet and Provider: Using the built-in J2EE server of Oracle JDeveloper, you can validate the configuration of your provider and its portlets.
- Deploying to an Application Server: Use Oracle JDeveloper to deploy your application to your application server.
- Registering and Viewing Your Portlet: Register and view your portlet with your local OracleAS Portal instance.

5.4.2.1 Creating a Portlet and Provider

This section walks you through the Portlet Wizard. You can choose which portlet Show modes you want to implement and the implementation method (JSP, HTTP servlet, Java class, or HTML). The wizard then creates a simple sample implementation for each of the selected modes.

1. After you open Oracle JDeveloper, click the workspace where you want to create this project. If you do not have a workspace, you can create one as follows:
   a. Right-click the **Applications** node in the Applications - Navigator and choose **New** from the context menu.
   b. Choose **Workspace** from the Items list in the New Gallery dialog box.
   c. Click **OK**.
   d. Enter a **Workspace Name** and **Directory Name**, and clear **Add a New Empty Project** in the Create Workspace dialog box.
   e. Click **OK**.
2. Right-click the name of the workspace in the Applications - Navigator and choose **New Project** from the context menu.
3. Click **Empty Project** in the Items list in the New Gallery dialog box.
4. Click **OK**.

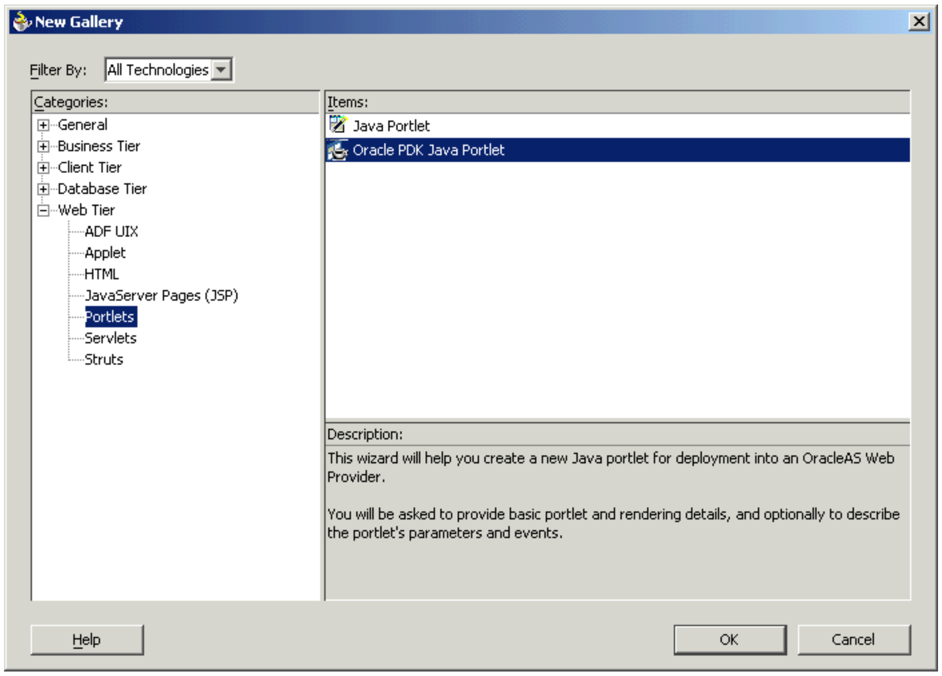
5. Enter the **Project Name** and **Directory Name** in the Create Project dialog box.
6. Click **OK**.
7. Right-click your project and select **New** from the context menu.
8. In the Categories list, expand the Web Tier category and click **Portlets**.

> **Note:** If you cannot find Portlets, refer to Section 5.3.1, "Installing the Oracle JDeveloper Portal Add-In" to ensure that you have installed the add-in correctly.

9. In the Items list, click **Oracle PDK Java Portlet**.

> **Note:** Clicking **Java Portlet** opens the Portlet Wizard for creating JPS-compliant portlets. Clicking **Oracle PDK Java Portlet** opens the Portlet Wizard for creating PDK-Java portlets.

Figure 5–20 New Gallery Dialog Box for Oracle PDK Java Portlet

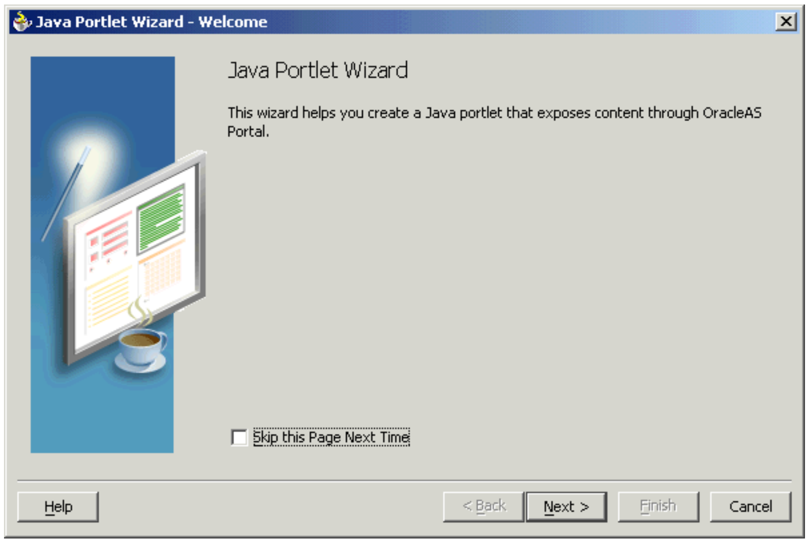


This wizard will help you create a new Java portlet for deployment into an OracleAS Web Provider.

You will be asked to provide basic portlet and rendering details, and optionally to describe the portlet's parameters and events.

10. Click **OK**. The Portlet Wizard displays.
11. If you are on the Welcome page of the wizard, click **Next**.

Figure 5–21 Welcome Page



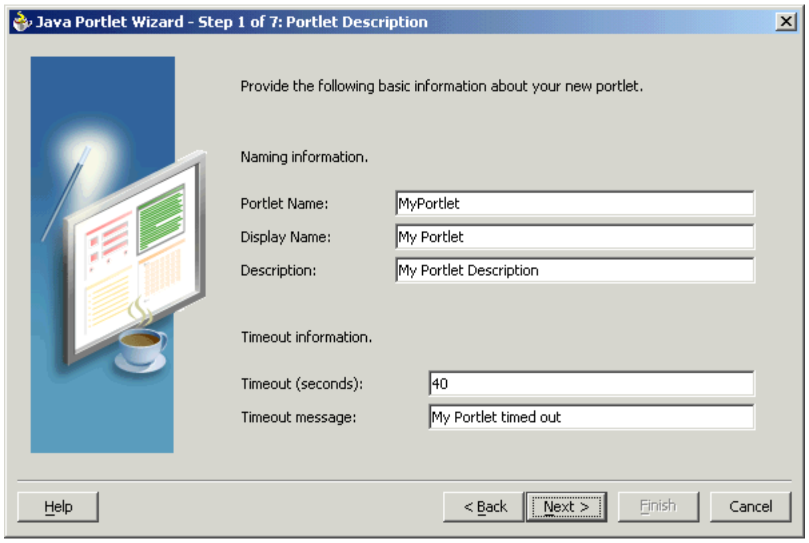
Java Portlet Wizard

This wizard helps you create a Java portlet that exposes content through OracleAS Portal.

☐ Skip this Page Next Time

12. On the Portlet Description page, shown in Figure 5–22, enter the names, description, and timeout settings. For the purposes of this example, you can accept the default values on this page.

Figure 5–22 Portlet Description Page



Provide the following basic information about your new portlet.

Naming information.

Portlet Name: MyPortlet
Display Name: My Portlet
Description: My Portlet Description

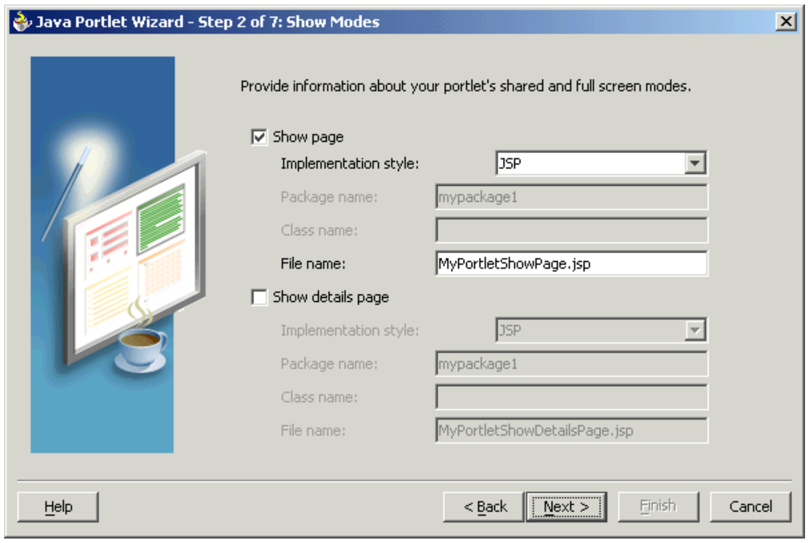
Timeout information.

Timeout (seconds): 40
Timeout message: My Portlet timed out

13. Click **Next**.
14. The next few pages of the wizard enable you to define the portlet modes for this portlet. The Show Modes page, shown in Figure 5–23, enables you to choose the implementation style for the Show page and whether you want to implement Show details page:
    a. Show page is selected by default. Choose an Implementation style from the list. For the purposes of this example, choose **JSP**. Enter the File name for the JSP to be generated by the Portlet Wizard. For the purposes of this example, you may accept the default file name.

b. If your portlet requires a details page, select Show details page and enter the Implementation style and File name as appropriate. For the purposes of this example, we do not need Show details.

Figure 5–23 Show Modes Page



Java Portlet Wizard - Step 2 of 7: Show Modes

Provide information about your portlet's shared and full screen modes.

[✓] Show page
Implementation style: JSP
Package name: mypackage1
Class name:
File name: MyPortletShowPage.jsp

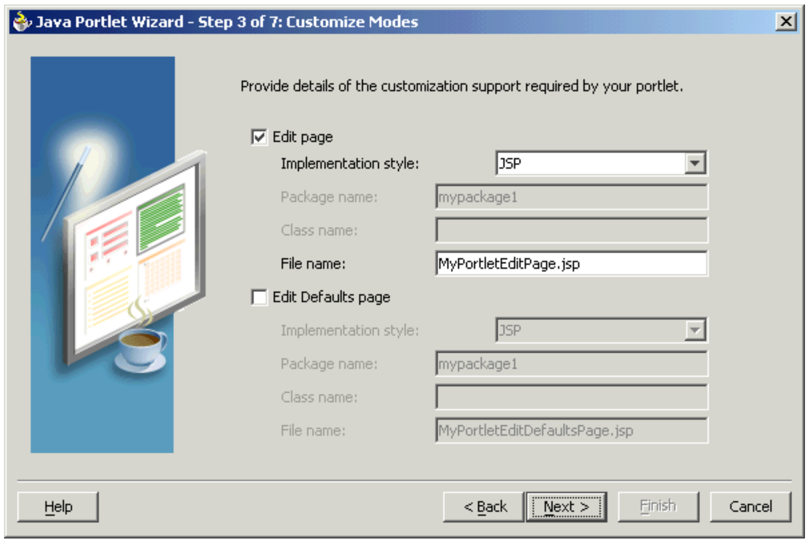
[ ] Show details page
Implementation style: JSP
Package name: mypackage1
Class name:
File name: MyPortletShowDetailsPage.jsp

15. Click **Next**.

16. On the Customize Modes page, shown in Figure 5–24, Edit page should already be selected. For the purposes of this example, choose an Implementation style of **JSP** and accept the default File name.

17. Select Edit Defaults page and, for the purposes of this example, choose an Implementation style of **JSP** and accept the default File name.

Figure 5–24 Customize Modes Page



Java Portlet Wizard - Step 3 of 7: Customize Modes

Provide details of the customization support required by your portlet.

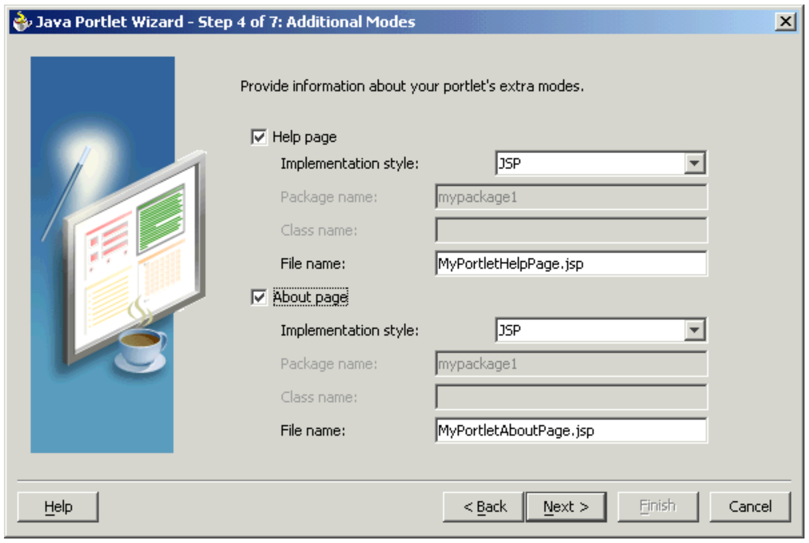
[✓] Edit page
Implementation style: JSP
Package name: mypackage1
Class name:
File name: MyPortletEditPage.jsp

[ ] Edit Defaults page
Implementation style: JSP
Package name: mypackage1
Class name:
File name: MyPortletEditDefaultsPage.jsp

18. Click **Next**.

19. On the Additional Modes page, shown in Figure 5–25, nothing is selected by default. For the purposes of this example, select **Help page**, choose an Implementation style of **JSP**, and accept the default File name.
20. Select **About page**, choose an Implementation style of **JSP**, and accept the default File name.

Figure 5–25 Additional Modes Page


Java Portlet Wizard - Step 4 of 7: Additional Modes

Provide information about your portlet's extra modes.

[✓] Help page
Implementation style: JSP
Package name: mypackage1
Class name:
File name: MyPortletHelpPage.jsp

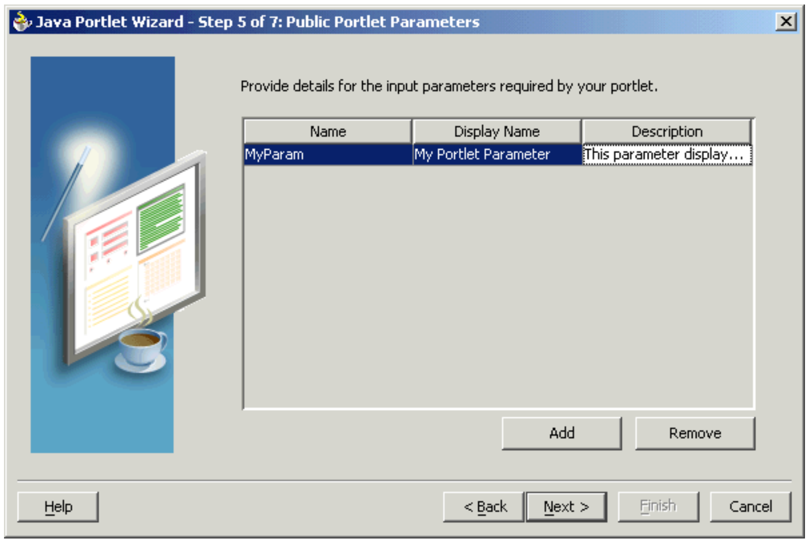
[✓] About page
Implementation style: JSP
Package name: mypackage1
Class name:
File name: MyPortletAboutPage.jsp

21. Click **Next**.
22. On the Public Portlet Parameters page, shown in Figure 5–26, click **Add**.
23. Enter the values as described in Table 5–5 and shown in Figure 5–26.

Table 5–5 Public Portlet Parameter

Name	Display Name	Description
`MyParam`	`My Portlet Parameter`	This parameter displays a value.

Figure 5–26 Public Portlet Parameters Page


Java Portlet Wizard - Step 5 of 7: Public Portlet Parameters

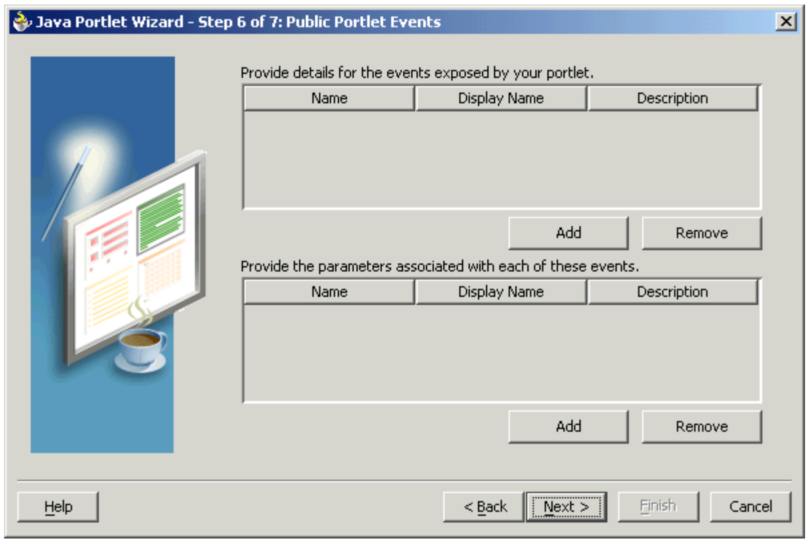
Provide details for the input parameters required by your portlet.

Name	Display Name	Description
MyParam	My Portlet Parameter	This parameter display...

24. Click **Next**.

25. On the Public Portlet Events page, shown in Figure 5–27, you can map parameters to events. For the purposes of this example, leave this page empty and click **Next**.

Figure 5–27 Public Portlet Events Page


Java Portlet Wizard - Step 6 of 7: Public Portlet Events

Provide details for the events exposed by your portlet.

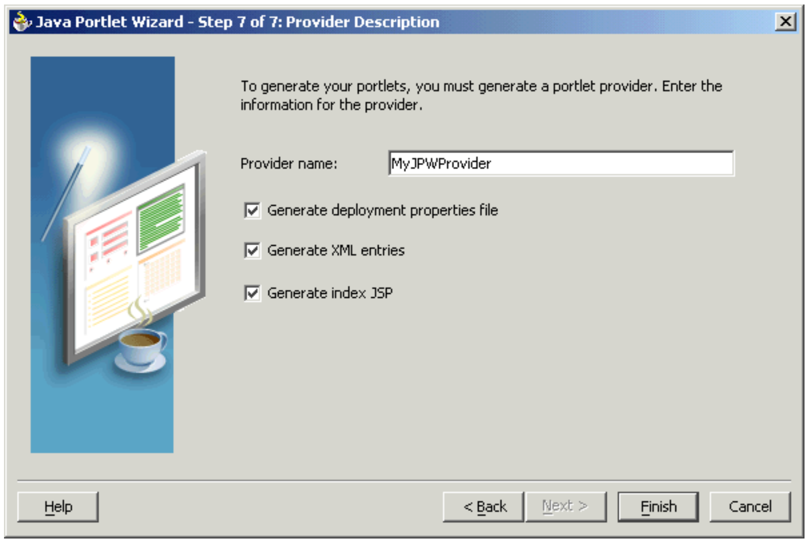
Name	Display Name	Description

Provide the parameters associated with each of these events.

Name	Display Name	Description

26. On the Provider Description page, shown in Figure 5–28, enter `MyJPWProvider` as the Provider name. Ensure that all of the check boxes are selected.

Figure 5–28 Provider Description Page



To generate your portlets, you must generate a portlet provider. Enter the information for the provider.

Provider name: MyJPWProvider

[✓] Generate deployment properties file

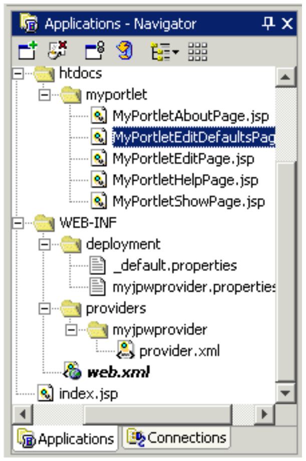
[✓] Generate XML entries

[✓] Generate index JSP

27. Click **Finish** to generate the files for your portlet. In this case, the following files should be generated for your project in the Applications - Navigator (see Figure 5–29):
    - Files for each portlet mode you selected.
    - `provider.xml`
    - `web.xml`
    - `index.jsp`
    - `_default.properties`
    - `myjpwprovider.properties`

    All of the above files are required to deploy and run the portlet successfully, except for `index.jsp`, which is used by Oracle JDeveloper for testing purposes.

Figure 5–29 Applications - Navigator



```
htdocs
  myportlet
    MyPortletAboutPage.jsp
    MyPortletEditDefaultsPag
    MyPortletEditPage.jsp
    MyPortletHelpPage.jsp
    MyPortletShowPage.jsp
WEB-INF
  deployment
    _default.properties
    myjpwprovider.properties
  providers
    myjpwprovider
      provider.xml
  web.xml
index.jsp
```

5.4.2.2 Adding Portlet Logic

Once you create the default implementation, you can extend the sample code with your business logic to implement the desired functionality and features. For the purposes of this example, you do not need to perform this step and can proceed directly to the testing and registration procedures.

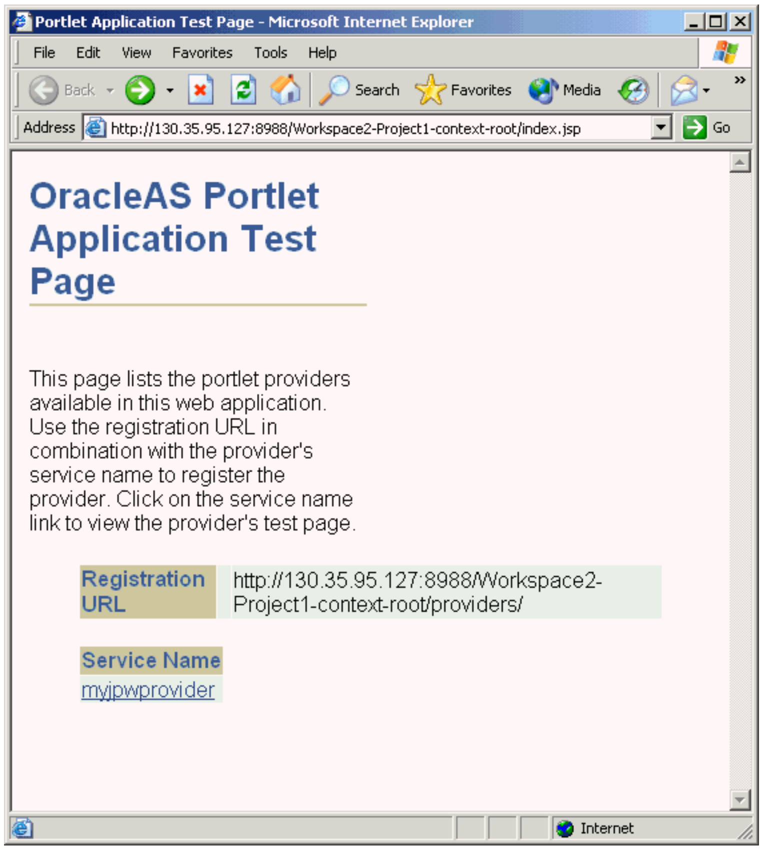
5.4.2.3 Validating Your Portlet and Provider

After you have built your portlet, you need to check the configuration to ensure that the portlet and its provider operate correctly.

Note: This procedure is for testing purposes only. After this procedure, you still need to register your provider as described in Section 5.4.2.5, "Registering and Viewing Your Portlet". For development and production, you should always deploy your portlet to an application server as described in Section 5.4.2.4, "Deploying to an Application Server".

1. If it is not already open, open Oracle JDeveloper and the project your created in the previous sections.
2. Find the `index.jsp` file for your portlet in the Navigator and right-click it.
3. Choose **Run** from the context menu. Your browser should open with a page similar to the one shown in Figure 5–30.

Figure 5–30 Portlet Application Test Page


Portlet Application Test Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
Go
Internet

http://130.35.95.127:8988/Workspace2-Project1-context-root/index.jsp

OracleAS Portlet Application Test Page

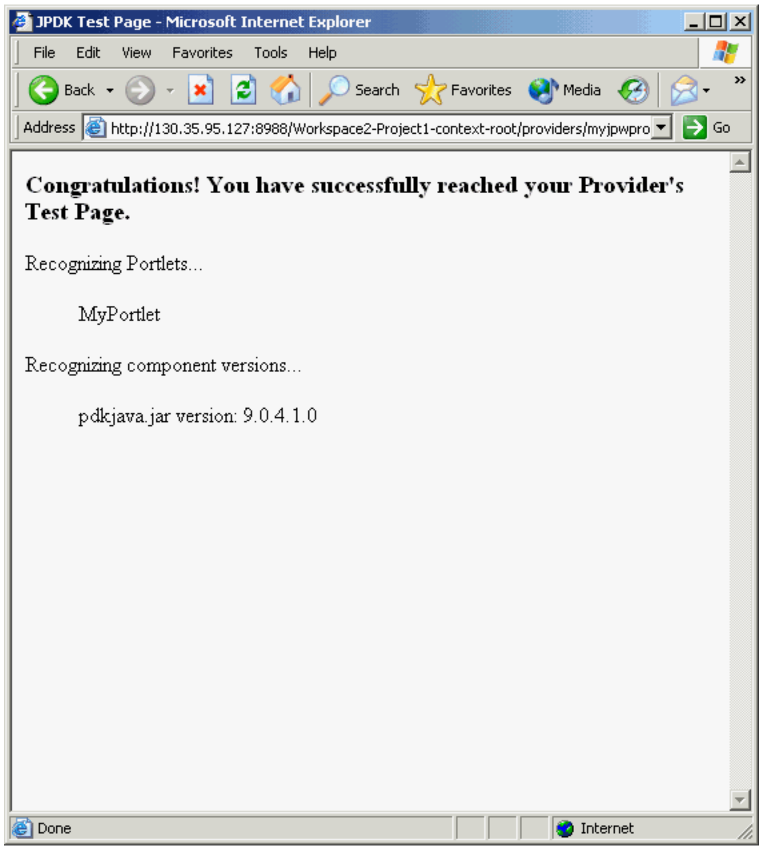
This page lists the portlet providers available in this web application. Use the registration URL in combination with the provider's service name to register the provider. Click on the service name link to view the provider's test page.

Registration URL	http://130.35.95.127:8988/Workspace2-Project1-context-root/providers/

Service Name
myjpwprovider

4. Click the link underneath Service Name. Your browser should open with a page similar to the one shown in Figure 5–31. Note that you need the URL from this page to register your provider, which is the next procedure.

Figure 5–31 Provider Test Page


JPDK Test Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
Go
Done
Internet

http://130.35.95.127:8988/Workspace2-Project1-context-root/providers/myjpwpro

Congratulations! You have successfully reached your Provider's Test Page.

Recognizing Portlets...

MyPortlet

Recognizing component versions...

pdkjava.jar version: 9.0.4.1.0

5.4.2.4 Deploying to an Application Server

After you finish the wizard and successfully generate your portlet, you are ready to deploy it to an application server, Oracle Application Server Containers for J2EE. The following procedures describe how to deploy a portlet to Oracle Application Server Containers for J2EE:

- Creating a Connection to Oracle Application Server Containers for J2EE
- Deploying the WAR File

5.4.2.4.1 Creating a Connection to Oracle Application Server Containers for J2EE To establish a connection to your application server, perform the following steps:

> **Note:** The steps that follow describe the procedure for deploying to a standalone instance of Oracle Application Server Containers for J2EE. For information about deploying to a full Oracle Application Server instance, please refer to the Oracle JDeveloper online help system.

1. If it is not still open, start Oracle JDeveloper and open the project you created in Section 5.4.2.1, "Creating a Portlet and Provider".

2. In the Navigator, right-click **Connections** and select **New Application Server Connection** from the context menu. Complete the wizard that displays as follows:

   a. If the Welcome page appears, click **Next**. If you do not want the Welcome page to appear in future, be sure to select **Skip this Page Next Time**.

   b. Enter a meaningful name for the connection (for example, `PDKJavaOC4J`) and choose **Standalone OC4J** as the connection type.

   c. Click **Next**.

   d. Enter the administrator's user name and password. This password was set during the installation of the Oracle Application Server Containers for J2EE.

   e. Click **Next**.

   f. Enter the information in Table 5–6.

Table 5–6 Settings for New Application Server Connection

Setting	Value

URL	Enter the full RMI URL for this setting. For example:
	`ormi://my.machine.com:23791/`
	The RMI port number may be found in:
	`OC4J_HOME/j2ee/home/config/rmi.xml`

Target Web Site	Enter the name of the target Web site containing your deployed J2EE application files. For the purposes of this example, you can accept the default value, `http-web-site`.
Local Directory Where `admin.jar` for OC4J Is Installed	Enter the path to the local `admin.jar` that is version-compatible with the remote servers specified in the URL setting above. For portlets you plan to deploy to Oracle Application Server Containers for J2EE 9.0.3, you can use the default admin.jar included with Oracle JDeveloper 9.0.3. For portlets you plan to deploy to Oracle Application Server Containers for J2EE 9.0.4, you need to change the default path in this setting to point to an instance of Oracle Application Server Containers for J2EE Release 9.0.4. For example: `OC4J_HOME\j2ee\home`

   g. Click **Next**.

   h. Verify the connection details by clicking **Test Connection**. A success message should display if everything is correct. If the test fails, you may need to revise your connection information.

   i. Click **Finish**.

5.4.2.4.2 Deploying the WAR File To create and deploy a WAR file, perform the following steps:

1. Right-click your portlet project and choose **New** from the context menu.
2. In the New dialog box, under Categories, choose **General > Deployment Profiles** and, under Items, choose **WAR File - J2EE Web Module**.
3. In the Save Deployment Profile dialog box, change the name to something meaningful (for example, `myj2eeportlet1.deploy`).
4. Click **OK**.
5. In the Profile Settings dialog box, perform the following steps:

   a. Click S**pecify J2EE Web Context Root** and enter `myj2eeportlet1`.
   b. In the pane on the left, choose **File Groups > WEB-INF/lib > Contributors**.
   c. Check **Portlet Development**.
   d. Click **OK**.
   e. Choose **File > Save All**.

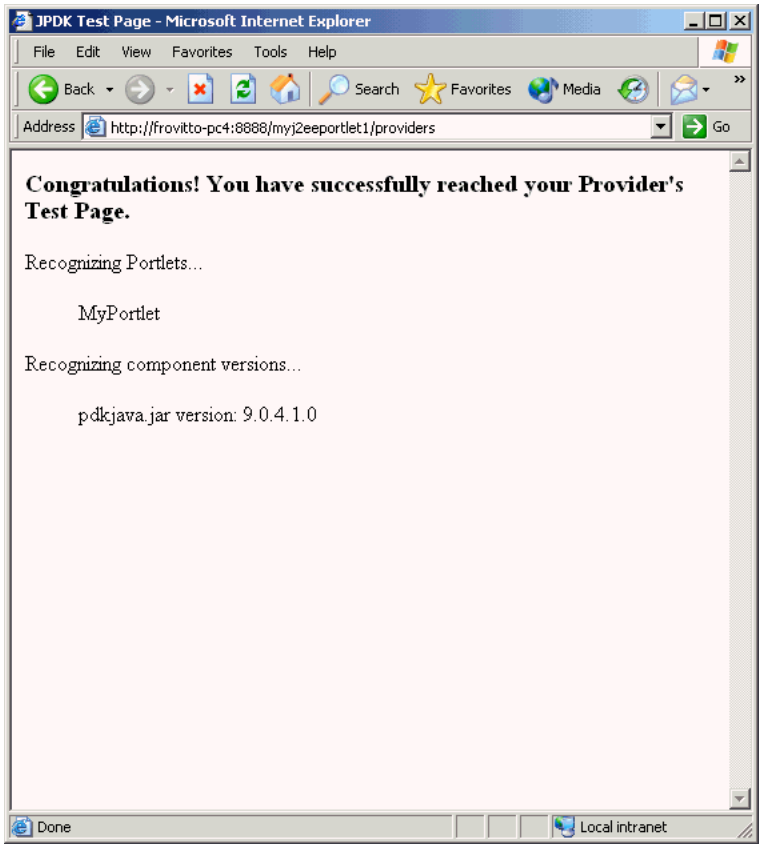
6. Right-click the deployment profile (for example, `myj2eeportlet1.deploy`) and choose **Deploy to >** the application server connection (for example, `PDKJavaOC4J`) from the context menu.
7. Await the Deployment Finished message in the Deployment Log at the bottom of Oracle JDeveloper and verify that no errors occurred.
8. Take the URL provided in the **Deployment Log** (for example, `http://myserver.com:8888/myj2eeportlet1`) and append /providers to construct the URL you use to test and register your J2EE portlet with OracleAS Portal. For example:

```
http://myserver.com:8888/myj2eeportlet1/providers
```

Note: In some cases, you may get a message in the log stating that Oracle JDeveloper was unable to determine the HTTP port number of the remote server. Typically, you can determine the port number yourself by looking at the URL that takes you to your Oracle Application Server Containers for J2EE home page (for example, `http://myserver.com:8888`).

9. Enter the URL you constructed in the preceding step in your browser. You should see a page similar to the one in Figure 5–32.

Figure 5–32 PDK - Java Test Page for Portlets


JPDK Test Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address http://frovitto-pc4:8888/myj2eeportlet1/providers
Go
Done
Local intranet

Congratulations! You have successfully reached your Provider's Test Page.

Recognizing Portlets...

MyPortlet

Recognizing component versions...

pdkjava.jar version: 9.0.4.1.0

5.4.2.5 Registering and Viewing Your Portlet

After you've created and deployed the provider and its portlet(s), you must register the provider with OracleAS Portal. Registering your provider gives OracleAS Portal the information it needs to locate and communicate with your provider. After you register a provider, the provider and its portlets become available in the Portlet Repository. They are also listed in the OracleAS Portal Navigator.

Note: When you build portlets and providers with built-in tools, such as the Portlet Builder, OracleAS Portal automatically registers the provider for you. Once you've created your portlet, it automatically displays in the Portlet Repository. OracleAS Portal also offers built-in portlets that are contained in a preconfigured provider. For example, OmniPortlet and Web Clipping are portlets that you can use out of the box, and are already registered with OracleAS Portal. You can view these portlets in the Add Portlets list. However, if you build the portlets and providers programmatically, you must then register these providers in order to make them available to the portal user.

This section describes how to register your provider and add your portlet to pages:

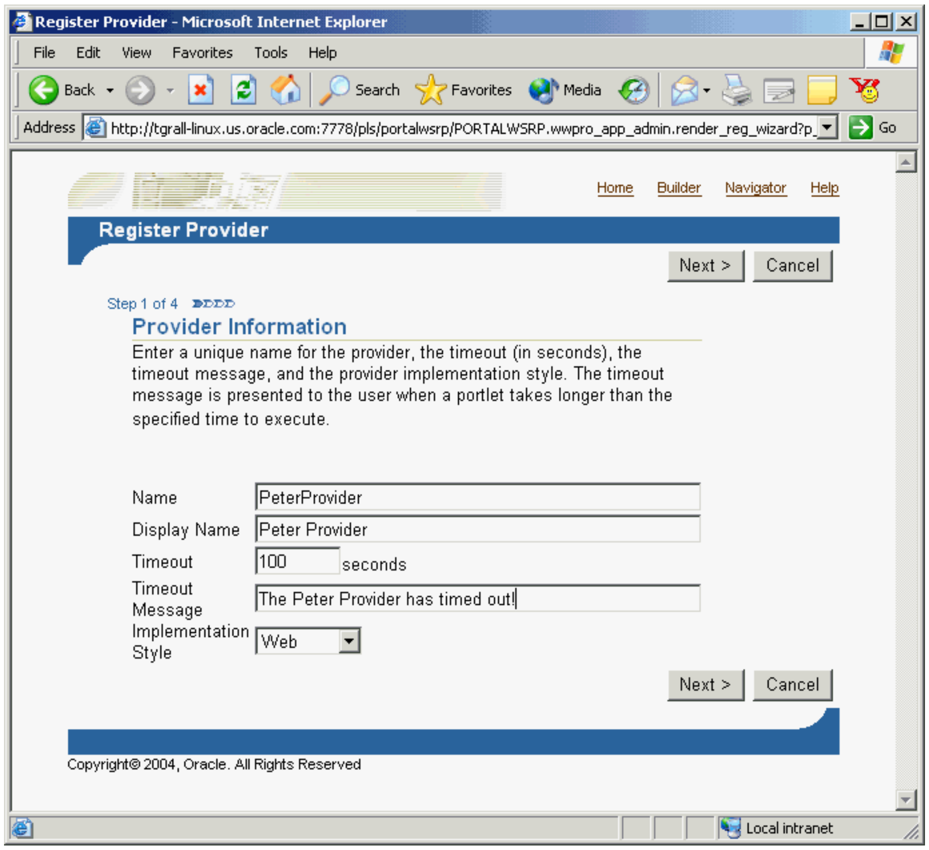
1. Open OracleAS Portal and log in as normal. Note that to register your provider, you need to have Manage or Edit privileges on providers. If you do not have these privileges, you need to request them from your administrator.
2. If you are not already on the Builder page, click **Builder** in the upper right corner.

3. Click the **Administer** tab.
4. Click the **Portlets** sub tab.
5. In the Remote Providers portlet, click **Register a Provider**.
6. On the Provider Information page, shown in Figure 5–33, enter the values as described in Table 5–7.

Table 5–7 Provider Information Page Values

Setting	Value
Name	`*your_name*PDKJProvider`
Display Name	`*your_name* PDKJProvider`
Timeout	100
Timeout Message	`The *your_name* PDKJProvider has timed out!`
Implementation Style	`Web`

Figure 5–33 Register Provider Page


Register Provider - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://tgrall-linux.us.oracle.com:7778/pls/portalwsrp/PORTALWSRP.wwpro_app_admin.render_reg_wizard?p_ Go
Home Builder Navigator Help
Register Provider
Next > Cancel
Step 1 of 4

Provider Information

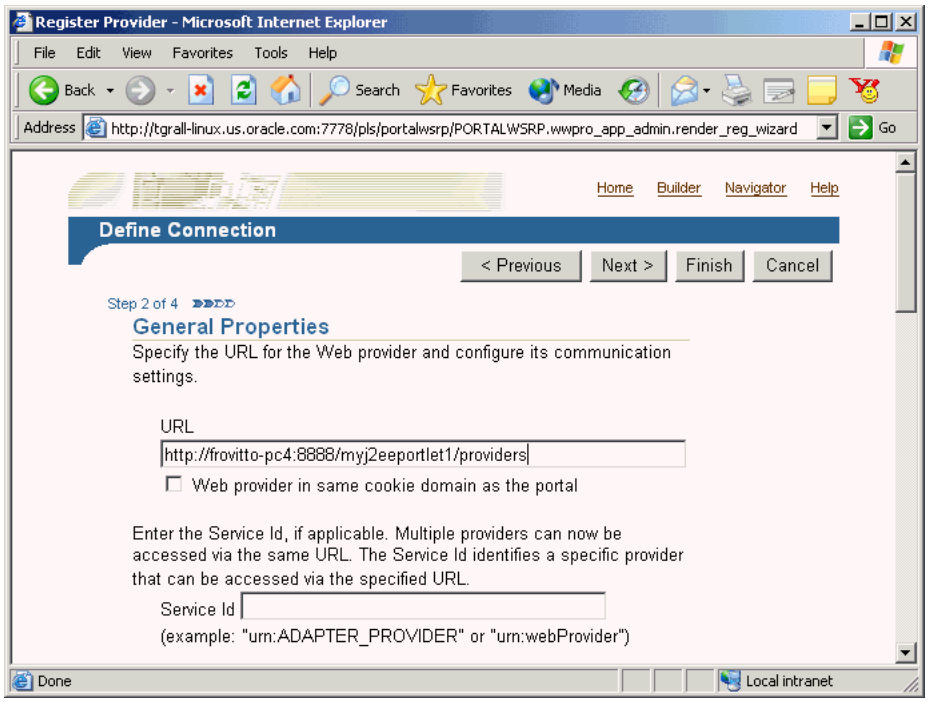
Enter a unique name for the provider, the timeout (in seconds), the timeout message, and the provider implementation style. The timeout message is presented to the user when a portlet takes longer than the specified time to execute.

Name	PeterProvider
Display Name	Peter Provider
Timeout	100 seconds
Timeout Message	The Peter Provider has timed out!
Implementation Style	Web

7. Click **Next**.
8. On the Define Connection page, shown in Figure 5–34, enter the URL for your provider in the URL field. This URL is the one that you created in step 8 of Section 5.4.2.4.2, "Deploying the WAR File". For example:

```
http://myserver.com:8888/myj2eeportlet1/providers
```

Figure 5–34 Define Connection Page


Register Provider - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://tgrall-linux.us.oracle.com:7778/pls/portalwsrp/PORTALWSRP.wwpro_app_admin.render_reg_wizard
Go
Home Builder Navigator Help
< Previous Next > Finish Cancel
Done
Local intranet

Define Connection

Step 2 of 4

General Properties

Specify the URL for the Web provider and configure its communication settings.

URL: http://frovitto-pc4:8888/myj2eeportlet1/providers

☐ Web provider in same cookie domain as the portal

Enter the Service Id, if applicable. Multiple providers can now be accessed via the same URL. The Service Id identifies a specific provider that can be accessed via the specified URL.

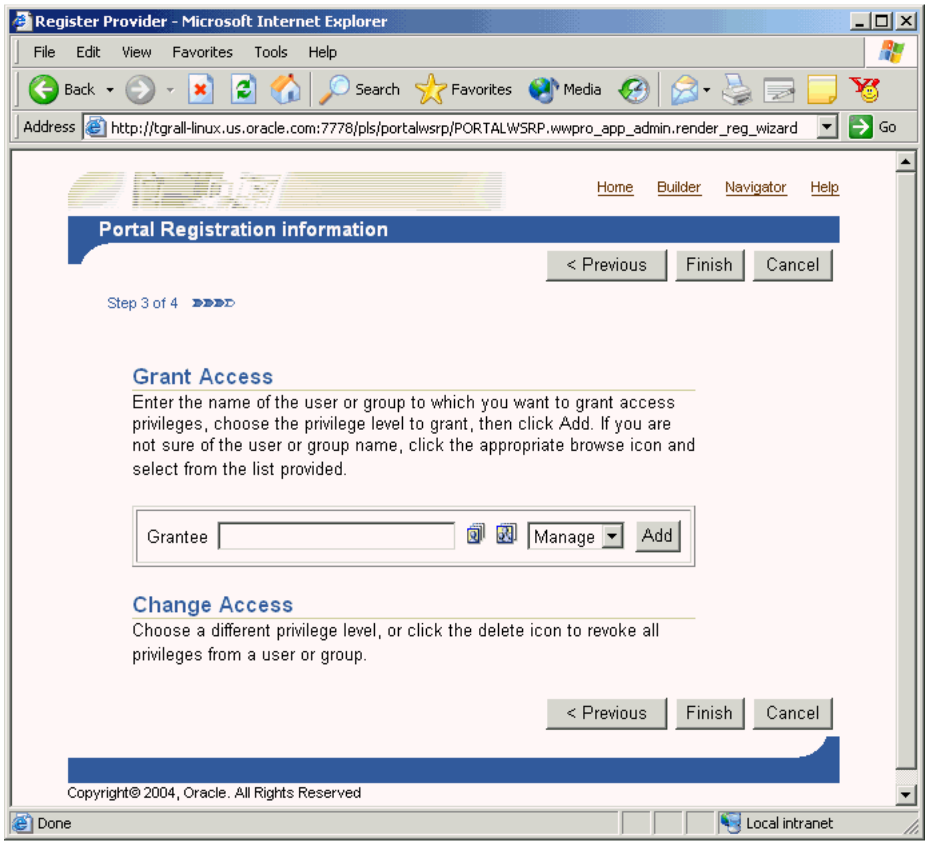
Service Id

(example: "urn:ADAPTER_PROVIDER" or "urn:webProvider")

9. Click **Next**.

10. On the Control Access page, shown in Figure 5–35, assign privileges for this provider as desired. For the purposes of this example, you can enter a group (for example, PORTAL_ADMINISTRATORS) for Grantee and click **Add**.

Figure 5–35 Control Access Page


Register Provider - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://tgrall-linux.us.oracle.com:7778/pls/portalwsrp/PORTALWSRP.wwpro_app_admin.render_reg_wizard
Go
Home Builder Navigator Help
< Previous Finish Cancel
Manage Add
Done
Local intranet

Portal Registration information

Step 3 of 4

Grant Access

Enter the name of the user or group to which you want to grant access privileges, choose the privilege level to grant, then click Add. If you are not sure of the user or group name, click the appropriate browse icon and select from the list provided.

Grantee

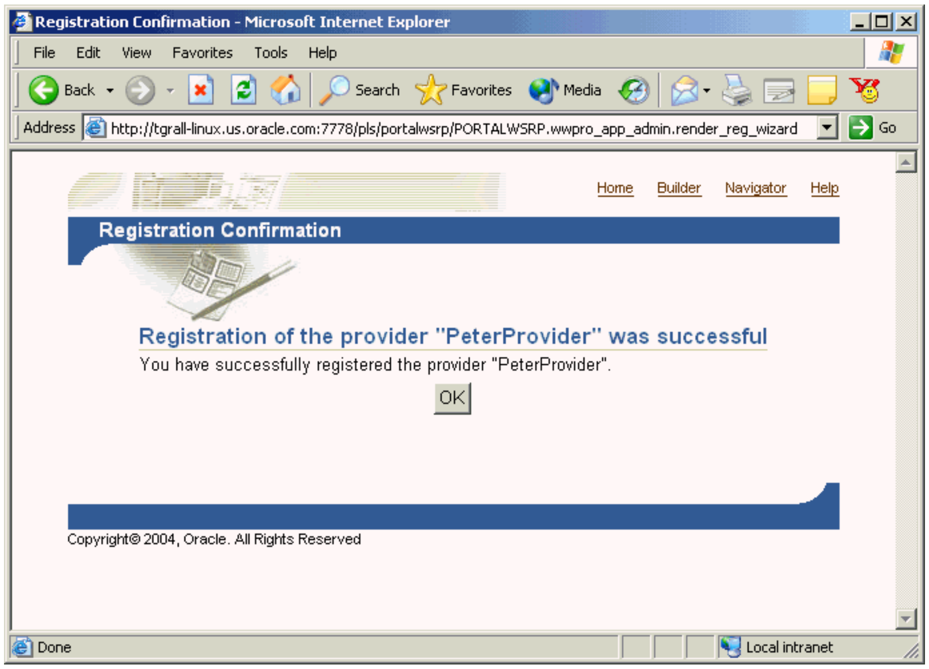
Change Access

Choose a different privilege level, or click the delete icon to revoke all privileges from a user or group.

Copyright© 2004, Oracle. All Rights Reserved

11. Click **Finish**. You should see a Registration Confirmation page similar to the one in Figure 5–36.

Figure 5–36 Registration Confirmation Page


Registration Confirmation - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://tgrall-linux.us.oracle.com:7778/pls/portalwsrp/PORTALWSRP.wwpro_app_admin.render_reg_wizard
Go
Home Builder Navigator Help
OK
Done
Local intranet

Registration Confirmation

Registration of the provider "PeterProvider" was successful

You have successfully registered the provider "PeterProvider".

Copyright© 2004, Oracle. All Rights Reserved

12. Your portlet should now be available for adding to pages just as any other portlet in the Portlet Repository. To add your portlet to a page, follow the instructions in Section 7.6.2, "Adding Portlets," of the *Oracle Application Server Portal User's Guide*.

5.4.3 Adding Render Modes

In the Portlet Wizard, you add Show modes by checking boxes on the wizard pages. Refer to Section 5.4.2, "Building PDK-Java Portlets" for more information about using the wizard. For each Show mode that you select in the wizard, a basic HelloWorld skeleton is created. If you need to add a Show mode after creating the portlet or you are adding one of the modes (preview or link) not available through the wizard, you can do that manually by updating `provider.xml` and HTML or JSPs in Oracle JDeveloper. To add Render modes, you need to perform the following tasks:

- Implementing Extra Show Modes
- Updating the XML Provider Definition
- Viewing the Portlet

Once you have completed this section, you will be able to implement any Render mode using RenderManager because the principles are the same for all modes. For example, even though this section does not describe how to implement the Help mode in detail, you will understand how to do it, as the process is the same as for Preview mode, which is described here.

To learn about the extra requirements for rendering the special **Edit** and **Edit Defaults** Render modes, refer to the article, "Adding Customization to Java Portlets" on the Oracle Technology Center for OracleAS Portal. For more detailed information on the PDK runtime classes used in this article, refer to the **JavaDoc**.

5.4.3.1 Assumptions

- You built a portlet using the wizard and successfully added it to a page.

5.4.3.2 Implementing Extra Show Modes

Your first task when creating Show modes manually is to create an HTML file or JSP for each mode. For example, if you want to implement Preview mode, you need to create an HTML file to provide preview content.

To create an HTML file to preview content, perform the following step:

1. In Oracle JDeveloper, open the project that contains your portlets and select the portlet in the Applications - Navigator to ensure the HTML page is created in the appropriate place. You can use Oracle JDeveloper's design view to easily create HTML pages. For example, the following HTML could serve as a preview page:

```
<p>This is the <i>preview</i> mode of your portlet!</p>
```

Once you have created the HTML file for previewing content, you are ready to update the XML provider definition.

5.4.3.3 Updating the XML Provider Definition

When you want to expose additional Render modes you must update your XML provider definition as follows:

- Set a boolean flag that indicates to the PDK Framework that a link or icon to that mode should be rendered.
- Point to the HTML file or JSP that you created for that mode.

For example, if you want to render Preview mode, perform the following steps:

1. Edit the provider definition file, `provider.xml` and add the tag to activate Preview mode:

   ```
   <showPreview>true</showPreview>
   ```

2. Specify the preview page to be the HTML page that you created in Section 5.4.3.2, "Implementing Extra Show Modes":

   ```
   <previewPage>/htdocs/myportlet/MyPortletPreviewPage.html</previewPage>
   ```

3. Save the updates to `provider.xml`.
4. Redeploy your portlet. Refer to step 6 in Section 5.4.2.4.2, "Deploying the WAR File".

5.4.3.4 Viewing the Portlet

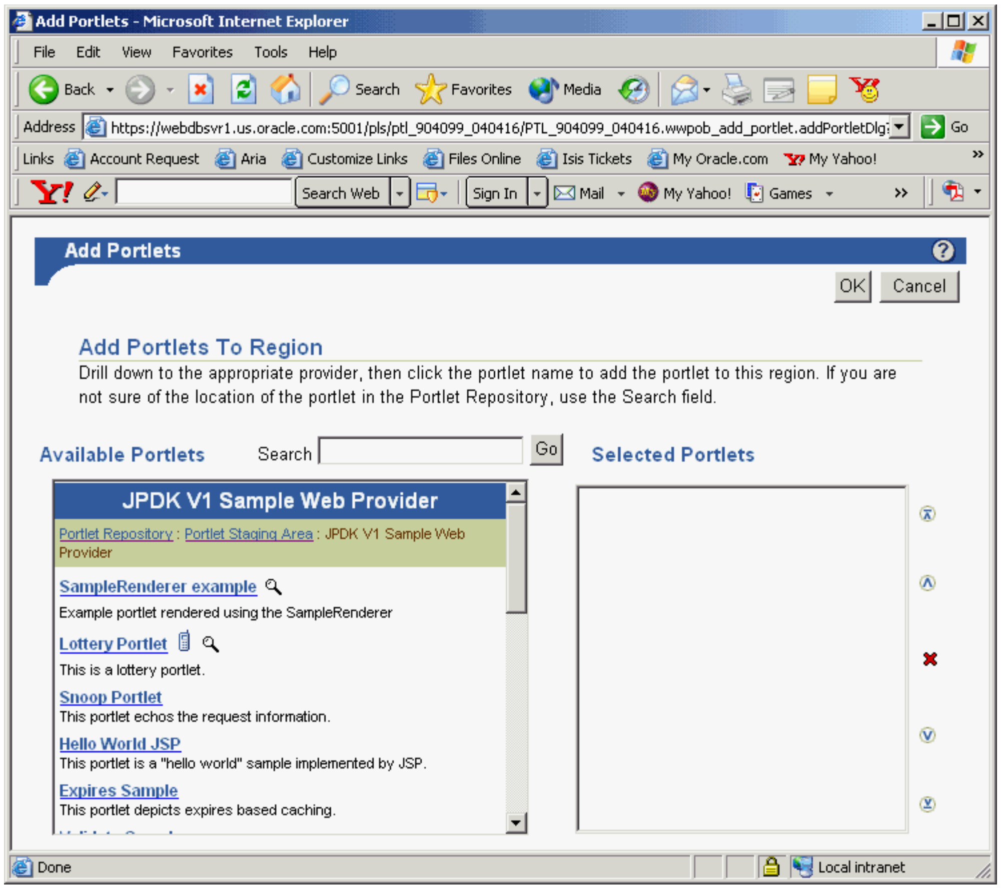
To view the new Render modes, you must ensure that your updated XML provider definition is re-parsed. To do this, perform the following tasks:

1. Copy the HTML file you created in Section 5.4.3.2, "Implementing Extra Show Modes" and `provider.xml` to the Oracle Application Server Containers for J2EE instance where you plan to deploy the portlet.
2. Refresh the provider.
3. Refresh the portal page containing your portlet.

To view Preview mode, do the following:

1. Edit a page or create a new page and choose **Add Portlet**.
2. Navigate to the location of your portlet with a Preview mode (for example, Portlet Staging Area). Note the magnifying glass icon next to the portlet shown in Figure 5–37

Figure 5–37 Add Portlet Page


Add Portlets - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://webdbsvr1.us.oracle.com:5001/pls/ptl_904099_040416/PTL_904099_040416.wwpob_add_portlet.addPortletDlg
Go
Links Account Request Aria Customize Links Files Online Isis Tickets My Oracle.com My Yahoo!
Search Web Sign In Mail My Yahoo! Games
Add Portlets
OK Cancel
Available Portlets
Search
Go
Selected Portlets
Done
Local intranet

Add Portlets To Region

Drill down to the appropriate provider, then click the portlet name to add the portlet to this region. If you are not sure of the location of the portlet in the Portlet Repository, use the Search field.

JPDK V1 Sample Web Provider

Portlet Repository : Portlet Staging Area : JPDK V1 Sample Web Provider

SampleRenderer example
Example portlet rendered using the SampleRenderer

Lottery Portlet
This is a lottery portlet.

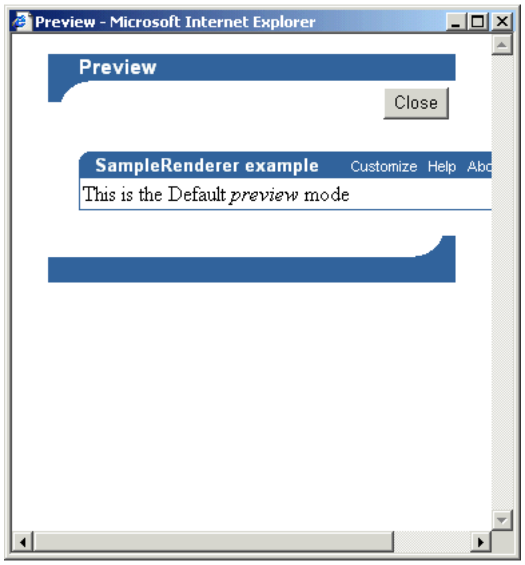
Snoop Portlet
This portlet echos the request information.

Hello World JSP
This portlet is a "hello world" sample implemented by JSP.

Expires Sample
This portlet depicts expires based caching.

3. Click the magnifying glass icon next to the portlet and a preview window similar to the one in Figure 5–38 appears.

Figure 5–38 Preview Window



This is the Default *preview* mode

5.4.4 Customizing Portlets

In Section 5.4.3, "Adding Render Modes" you learned how to use the PDK Provider Framework to activate and render additional Show modes that were either not activated when creating the portlet with the wizard or not available through the wizard (that is, Link and Preview modes). This section describes the two Customization modes (Edit and Edit Defaults) in more detail. When checked in the Java Portlet Wizard, Edit page and Edit Defaults page cause the generation of skeleton code for the two Customization modes. The skeleton code enables you to access the personalization framework with a few lines of code rather than completely hand coding a customization framework and a data store to hold the values.

To add customization to your portlet, you need to do the following:

- Update the Edit page of your portlet to set and retrieve customization changes.
- Update the Edit Defaults page of your portlet to set and retrieve customization changes.
- Update the Show page of your portlets to use the customization set by the user.

The Edit and Edit Defaults modes allow portlet users to change a set of customizable parameters supported by the portlet, which typically drive the way the portlet is rendered in other modes. For a particular instance of a portlet on an OracleAS Portal page, the customizations made in the Edit and Edit Defaults modes apply only to that instance of the portlet.

- **Edit** mode customizations are specific to the individual user making the customizations. This mode is activated by clicking the **Customize** link on the portlet header in show mode.
- **Edit defaults** mode customizations apply to all users in the same locale who have not yet made specific customizations to that portlet instance. This mode is generally only available to page designers, and can be activated by following the **Edit** icon on the page.

When rendering Edit and Edit Defaults modes, a `PortletRenderer` can carry out either of these tasks to support the customization process:

- **Renders the Edit Form**: For each of the portlet's customizable parameters, `PortletRenderer` uses a `PortletPersonalizationManager` to retrieve the current value and renders a control in an HTML form so the current value can be edited.
- **Handles Edit Form actions**: When an **OK** or **Apply** button is clicked on the standard edit form header, `PortletRenderer` uses a PortletPersonalizationManager to store the customized parameters submitted by the edit form and redirects the browser to the appropriate portal page.

Therefore, the purpose of the `PortletPersonalizationManager` controller is to enable a `PortletRenderer` to store and retrieve the current values of customizable parameters that apply to a particular portlet instance and user. The PDK Framework uses the abstraction of a `PersonalizationObject` as a container for a set of customized parameters and a `PortletReference` as the key under which a set of customizations are stored. Thus, a `PortletPersonalizationManager` is simply a mechanism that allows the storage and retrieval of persisted `PersonalizationObjects` under a given `PortletReference`.

A preference store is a mechanism for storing information like user preference data, portlet/provider settings, or even portlet data, while using OracleAS Portal. The information stored in the preference store is persistent in the sense that, even if you log out and log back in later, you could still access previously saved preferences. The preference store maintains the user preference information and invokes the user preferences whenever the user logs in again. PDK-Java provides the `PrefStorePersonalizationManager`, which uses a `PreferenceStore` implementation to persist customized data. Currently, PDK-Java has two `PreferenceStore` implementations: `DBPreferenceStore` and `FilePreferenceStore`. The DBPreferenceStore persists data using a JDBC compatible relational database and FilePreferenceStore persists data using the file system.

For more details of these implementations, consult the **JavaDoc**.

Note: PDK-Java provides the Preference Store Migration/Upgrade Utility to help migrate the preference store from a file system to a database and upgrade customizations from earlier releases. This utility is described more fully on the Oracle Technology Network (`http://www.oracle.com/technology/products/ias/portal/index.html`).

To add customization functionality to your portlet you use `PrefStorePersonalizationManager` in conjunction with `NameValuePersonalizationObject` (that is, the default `PersonalizationObject` implementation). By default, the wizard generates a simple edit form for both the Edit and Edit Defaults modes to enable users to customize the portlet title. This section describes how to update the existing code to enable portal users to customize the portlet greeting.

5.4.4.1 Assumptions

1. You have followed through and understood these sections:
   - Building PDK-Java Portlets
   - Adding Render Modes
2. You built a portlet using the wizard, with **Edit page** and **Edit Defaults page** checked, and successfully added it to a page.

5.4.4.2 Implementing Customization for Edit and Edit Defaults Pages

The Edit page of your portlet is called when a user customizes the portlet. By default, the JSP generated by the wizard includes all of the required code to provide customization of the portlet title. You just need to insert a few lines of code into the Edit page for additional customization.

5.4.4.2.1 Reviewing the Generated Code The wizard creates the following code for you by default:

```
<%@page contentType="text/html; charset=windows-1252"
   import="oracle.portal.provider.v2.render.PortletRenderRequest"
   import="oracle.portal.provider.v2.http.HttpCommonConstants"
   import="oracle.portal.provider.v2.personalize.NameValuePersonalizationObject"
   import="oracle.portal.provider.v2.render.PortletRendererUtil"
%>

<%
 PortletRenderRequest pReq = (PortletRenderRequest)
   request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
%>
```

```
<P>Hello <%=pReq.getUser().getName() %>.</P>
<P>This is the <b><i>Edit</i></b> render mode!</P>
<%-- This page both displays the customization
     form and processes it,. Display the form if
     there is no action parameter, process it
     otherwise --%>

<%
 String actionParam = PortletRendererUtil.getEditFormParameter(pReq);
 String action = request.getParameter(actionParam);
 String title = request.getParameter("my2portlet_title");
 NameValuePersonalizationObject data = (NameValuePersonalizationObject)
    PortletRendererUtil.getEditData(pReq);
 // Cancel automatically redirects to the page, so
 // will only receive OK or APPLY
 if (action !=null)
 {
    data.setPortletTitle(title);
    PortletRendererUtil.submitEditData(pReq, data);
    return;
 }

 // Otherwise just render the form.
 title = data.getPortletTitle();
%>
<table border="0">
  <td width="20%">
   <p align="right">Title:</p>
  </td>
  <td width="80%">
   <input type="TEXT" name="my2portlet_title" value="<%= title %>">
  </td>
</table>
```

5.4.4.2.2 Modifying the Generated Code The JSP contains an input field for the portlet title. This field represents the Customize page of the portlet where users can update the portlet title.

1. Following the table in the generated code, add a second table containing a text field and a prompt, allowing users to enter a new greeting for the portlet:

```
<table border="0">
  <tr>
    <td width="20%">
      <p align="right">Greeting:</p>
    </td>
    <td width="80%">
      <input type="TEXT" name="myportlet_greeting" value="<%= greeting %>">
    </td>
  </tr>
</table>
```

2. The HTML above simply specifies a field to enter a new greeting on the Edit page. This new greeting is displayed in the portlet's Shared Screen mode. Next, you add a string below `String title` that retrieves the value of the greeting:

```
String title = request.getParameter("my2portlet_title");
String greeting = request.getParameter("myportlet_greeting");
```

3. Generating an Edit page from the wizard automatically includes access to the personalization framework in the page code. At the top of the Edit page, you see the `NameValuePersonalizationObject` declared. This form of personalization in OracleAS Portal allows easy storage of name/value pairs.

   The Edit page handles two cases: viewing the page or applying changes to it. The changes we have made so far affect the code for viewing the page. Applying changes to the Edit page is handled in the block of code beginning with `if (action !=null)`.

   In this block of code, you must store the new portlet greeting. You must also account for the case where the user decides to make no changes and you simply retrieve the existing greeting:

   ```
   if (action !=null)
   {
        data.setPortletTitle(title);
        //Put the new greeting.
        data.putString("myportlet_greeting", greeting);
        PortletRendererUtil.submitEditData(pReq, data);
        return;
   }
   //Otherwise just render the form.
   title = data.getPortletTitle();
   //Get the old greeting.
   greeting = data.getString("myportlet_greeting");
   ```

You are now done updating the Edit page.

You can simply duplicate these changes for the Edit Defaults page. The Edit Defaults page is called when a page designer or portal administrator clicks **Edit** on the page and then clicks **Edit Defaults**. This page sets the default customization for this instance of the portlet. Even though the code in the JSP is identical, the PDK Framework and OracleAS Portal automatically handle the customization differently depending on the Show mode (Edit or Edit Defaults).

5.4.4.3 Implementing Customization for Show Pages

To have access to the personalization data in the portlet's Shared Screen mode, you need to add a few lines of code to the Show page. These lines include:

- Adding import statements.
- Declaring the `NameValuePersonalizationObject`.
- Retrieving the customization data.

1. Edit your Show page and import `NameValuePersonalizationObject` and `PortletRendererUtil`. You can copy these from the Edit page if necessary.

   ```
   <%@page contentType="text/html; charset=windows-1252"
      import="oracle.portal.provider.v2.render.PortletRenderRequest"
      import="oracle.portal.provider.v2.http.HttpCommonConstants"
      import="oracle.portal.provider.v2.personalize.
        NameValuePersonalizationObject"
      import="oracle.portal.provider.v2.render.PortletRendererUtil"
   %>
   ```

2. Declare the `NameValuePersonalizationObject` and retrieve the edit data from the portlet render request. You can copy this from the portlet's Edit page.

   ```
   <%
    PortletRenderRequest pReq = (PortletRenderRequest)
   ```

```
        request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
        NameValuePersonalizationObject data = (NameValuePersonalizationObject)
          PortletRendererUtil.getEditData(pReq);
    %>
```

3. Get the string information from the customization framework:

   ```
   String greeting = data.getString("myportlet_greeting");
   ```

4. Add some text to the Show page that displays the greeting in the Shared Screen mode of the portlet.

   ```
   <P>Hello <%= pReq.getUser()getName() %>.</P>
   <P>This is the <b><i>show</i>,</b> render mode!</P>
   <P>Greeting: <%= greeting %></P>
   ```

You have now completed updating the Show page of the portlet.

5.4.4.4 Preference Information Within the XML Provider Definition

The Portlet Wizard generates all of the needed tags for accessing the `PreferenceStore` in the XML provider definition file (`provider.xml`). By default, at the provider level, the wizard uses the `FilePreferenceStore` class to store preferences:

```
<provider class="oracle.portal.provider.v2.DefaultProviderDefinition">
<session>false</session>
<passAllUrlParams>false</passAllUrlParams>
<preferenceStore class="oracle.portal.provider.v2.preference.FilePreferenceStore">
     <name>prefStore1</name>
     <useHashing>true</useHashing>
</preferenceStore>
```

At the portlet level, tags are added to use `PrefStorePersonalizationManager` as the `personalizationManager` class and `NameValuePersonalizationObject` as the data class:

```
<personalizationManager class="oracle.portal.provider.v2.personalize.
   PrefStorePersonalizationManager">
    <dataClass>oracle.portal.provider.v2.NewValuePersonalizationObject</dataClass>
</personalizationManager
```

You need not make any changes or updates to the XML Provider Definition if you choose to continue to use the `FilePreferenceStore` class. However, if you have a global environment for OracleAS Portal (for example, you are running in a load balanced, multi-node cluster of Oracle Application Server Containers for J2EE instances) or would prefer to store preferences in the database, you can change the class from `FilePreferenceStore` to `DBPreferenceStore`.

Note: For more information on using DBPreferenceStore, refer to Section "5.3.6 Step 6: Configure Portal Tools and Web Providers (Optional)" of the *Oracle Application Server Portal Configuration Guide*.

5.4.4.5 Viewing the Portlet

To view the customization changes you made in the preceding sections, you need to deploy the portlet to your application server or Oracle Application Server Containers for J2EE and refresh the page containing your portlet. You should now see that the portlet contains a null greeting. click **Customize** in the portlet title bar and update the greeting. When you return to the page, you should see your changes.

You can also test Edit Defaults by clicking **Edit** on the page and then clicking **Edit Defaults**. Since you have already modified the portlet, the changes will not appear to you in Shared Screen mode unless you view the page as a public user or a different user.

5.4.5 Passing Parameters and Submitting Events

OracleAS Portal and the PDK provide page parameters, public and private portlet parameters, and events to enable portlet developers to easily write reusable, complex portlets. The Portlet Wizard in Oracle JDeveloper creates portlets that are already set up to use parameters and events. This feature enables you to focus solely on adding business logic to your portlets and does not require any changes to `provider.xml`.

For an overview of parameters and events, refer to the following:

- Section 2.12, "Public Portlet Parameters Support"
- Section 2.13, "Private Portlet Parameter Support"
- Section 2.14, "Event Support"

5.4.5.1 Assumptions

1. You have followed through and understood Section 5.4.2, "Building PDK-Java Portlets".
2. You built a portlet using the wizard and successfully added it to a page.

5.4.5.2 Adding Parameters to Your Portlets

Using the wizard in Section 5.4.2, "Building PDK-Java Portlets", you built a basic portlet and specified a parameter called `MyParam`. If you did not create a parameter, you can create a new portlet now by right clicking on `provider.xml` in the Applications - Navigator of Oracle JDeveloper, selecting **Add Portlet**, and following the steps in Section 5.4.2, "Building PDK-Java Portlets".

By default, the wizard creates a portlet to which you can easily map page parameters without updating any code or files. In this section, you will use the default parameter created for you by the wizard.

To use the default parameter, you only need to register the provider and add the portlet to a page. After that, you perform the following tasks:

- Create a page parameter.
- Wire the page parameter to your Java portlet.
- Enter parameter values in the URL or another portlet that passes this page parameter.

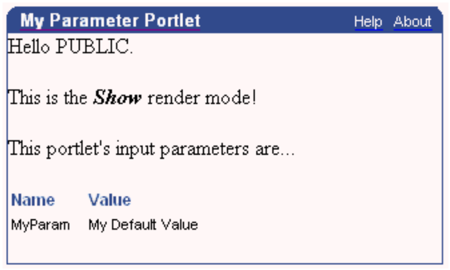
1. Go to the **Parameter** tab of the page properties. Note that parameters should be enabled by default, but, if not, you must enable them before proceeding.
2. Create a page parameter called `MyParameter` with a default value of `My Default Value`.
3. Expand your Java portlet and map the page parameter you just created to the portlet parameter.

   ```
   My Portlet Parameter = Page Parameter MyParameter
   ```

4. Go back to the page. Notice that, in the portlet, a value of My Default Value appears.
5. View the page and enter the parameter and a value at the end of the URL:

```
&MyParameter=This%20portlet%20works
```

Figure 5–39 Parameter Portlet



Hello PUBLIC.

This is the ***Show*** render mode!

This portlet's input parameters are...

Name	Value
MyParam	My Default Value

If you have a portlet, such as the Simple Parameter Form included with OmniPortlet, that can pass parameters, you can easily map parameters from that portlet to your Java portlet using the Events tab.

If you now take a look at the code and tags generated by the wizard, you see that very little code was needed to enable parameters in the Java portlet.

Review `provider.xml`. Note that the wizard added one tag group called `inputParameter`, which includes the name of the parameter for which the portlet listens.

```
<inputParameter class="oracle.portal.provider.v2.DefaultParameterDefinition">
  <name>MyParam</name>
  <displayName>My Portlet Parameter</displayName>
</inputParameter>
```

The wizard also generated code in the JSP that represents your Show page, which receives this parameter, and displays the parameter name and its value.

```
<%
ParameterDefinition params[] =
  pReq.GetPortletDefinition().getInputParameters();
%>

<p>This portlets input parameters are ...</p>
<table align="left" width="50%"><tr><td><span class="PortletHeading1">Value
  </span></td></tr>
<%
  String name = null;
  String value = null;
  String[] values = null;
for (int i = 0; i < params.length; i++)
{
  name = params[i].getName();
  values = pReq.getParameterValues(name);
  if (values != null)
  {
   StringBuffer temp = new StringBuffer();
   for (int j = 0; j < params.length; j++)
    {
       temp.append(values[j]);
```

```
        if (j + 1 != values.length)
        {
          temp.append(", ");
        }
      }
      value = temp.toString();
    }
    else
    {
      value = "No values submitted yet.";
    }
  %>
  <tr>
    <td><span class="PortletText2" <%= name %></span></td>
    <td><span class="PortletText2" <%= value %></span></td>
  </tr>
  <%
  }
  %>
  </table>
```

5.4.5.3 Submitting Events

In the previous section, you created a portlet that received parameters. Now you will create a portlet that passes parameters and events to other portlets on the same page or a different page. Some portlets, like the Simple Parameter Form in OmniPortlet, provide an easy, declarative interface to create a simple form to pass parameters to other portlets. If you want complete control over the events passed and the look of your portlet, though, you can add events to your Java portlet.

The Portlet Wizard does not create all of the code needed to pass parameters to other portlets. The wizard updates the tags in `provider.xml` and requires that you add the necessary business logic to your JSP code. To create a portlet that uses events, you perform the following tasks:

- Create a new portlet with the Portlet Wizard.
- Add code to your JSP page.
- Map this portlet's parameters to the portlet you created in Section 5.4.5.2, "Adding Parameters to Your Portlets".

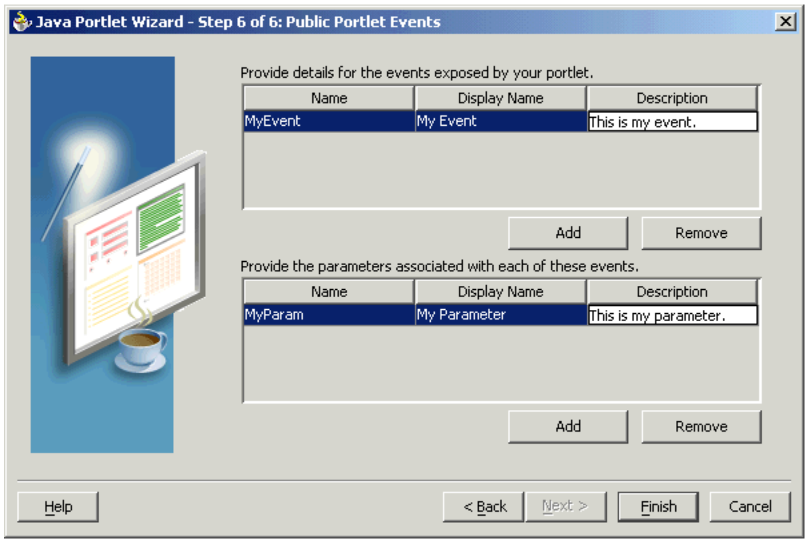
5.4.5.3.1 Creating an Events Portlet To create an events portlet, perform the following steps:

1. Create a new portlet called `MyEventsPortlet` in the same provider by invoking the Portlet Wizard. Go through the wizard as normal. In step 5 of the wizard, create a parameter. In step 6 of the wizard, enter the information shown in Table 5–8.

Table 5–8 Events

Events Area	Name	Display Name	Description
Events Exposed	MyEvent	My Event	This is my event.
Parameters Associated	MyParam	My Parameter	This is my parameter

Figure 5–40 Public Portlet Events Page of Portlet Wizard



Provide details for the events exposed by your portlet.

Name	Display Name	Description
MyEvent	My Event	This is my event.

Provide the parameters associated with each of these events.

Name	Display Name	Description
MyParam	My Parameter	This is my parameter.

The wizard generates the following code in `provider.xml`:

Note: In the following example, notice that the input parameter and the event parameter have the same name, `MyParam`. They are two different parameters, even though they have the same name.

```
<showDetails>false</showDetails>
<inputParameter class="oracle.portal.provider.v2.
  DefaultParameterDefinition">
   <name>MyParam</name>
   <displayName>My Parameter</displayName>
</inputParameter>
<event class="oracle.portal.provider.v2.DefaultEventDefinition">
   <name>MyEvent</name>
   <displayName>My Event</displayName>
   <parameter class="oracle.portal.provider.v2.DefaultParameterDefinition">
     <name>MyParam</name>
     <displayName>My Parameter</displayName>
   </parameter>
</event>
<renderer class="oracle.portal.provider.v2.render.RenderManager">
```

2. Import the needed classes:
   - `oracle.portal.provider.v2.event.EventUtils`
   - `oracle.portal.utils.NameValue`
   - `oracle.portal.provider.v2.url.UrlUtils`
3. Add a link that passes the parameter value to another portlet. As shown in the sample code below, you receive the same page parameter as the previous portlet, but in addition you create a link that passes an event as well:

```
<%@page contentType="text/html; charset=windows-1252"
import="oracle.portal.provider.v2.render.PortletRenderRequest"
import="oracle.portal.provider.v2.http.HttpCommonConstants"
import="oracle.portal.provider.v2.ParameterDefinition"
import="oracle.portal.provider.v2.event.EventUtils"
```

```
import="oracle.portal.utils.NameValue"
import="oracle.portal.provider.v2.url.UrlUtils"
%>
<%
PortletRenderRequest pReq = (PortletRenderRequest)
request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
%>
<%
  NameValue[] parameters = new NameValue[2];
  parameters[0] = new NameValue( EventUtils.eventName("MyEvent"),"");
  parameters[1] = new
NameValue(EventUtils.eventParameter("MyParam"),pReq.getParameter
  ("MyParam"));
%>
<span class="portletText1"><br>

<a href="<%= UrlUtils.constructLink
  (pReq, pReq.getRenderContext().getEventURL(), parameters , true, true)%>">
The value of the stock is <%= pReq.getParameter("MyParam") %>
</a>
<br><br></span>
```

Note: This sample code does not handle NULL values. When the portlet is initially added to the page, you may receive an error, but, after wiring the portlet to the page parameter, it should work fine.

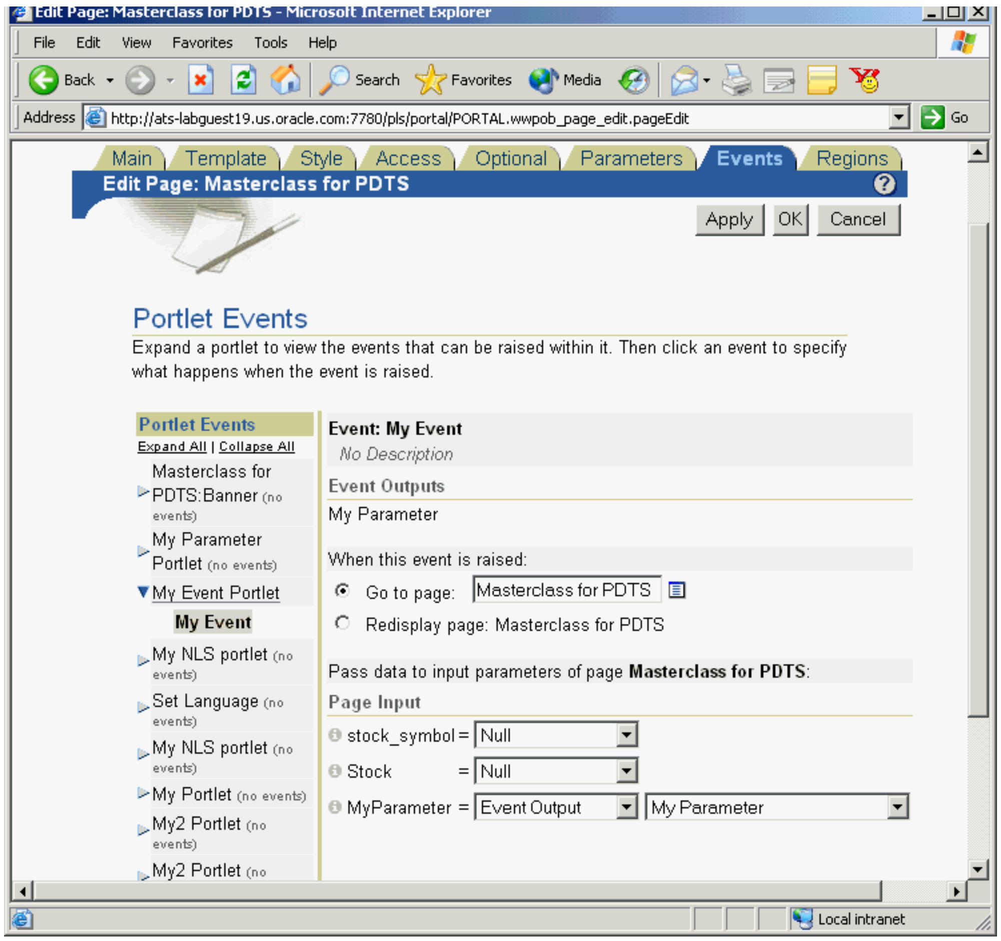
4. Add the portlet to a different page (in the same page group) than the previous portlet (the Parameter Portlet). Expand the portlet and wire it to receive the same parameter as the previous portlet.

```
My Parameter = Page Parameter MyParameter
```

5. Apply your changes on the Parameter tab and go to the Events tab. Expand the Event portlet and select the event. Select **Go to Page** and find the page to which you want to pass the event. Choose the page where the Parameter portlet is located. Configure this portlet to pass an event as the page parameter `MyParameter`.

```
MyParameter = Event Output MyParameter
```

Figure 5–41 Portlet Events in the Edit Page


Edit Page: Masterclass for PDTS - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://ats-labguest19.us.oracle.com:7780/pls/portal/PORTAL.wwpob_page_edit.pageEdit
Go
Main Template Style Access Optional Parameters Events Regions
Edit Page: Masterclass for PDTS
Apply OK Cancel
Local intranet

Portlet Events

Expand a portlet to view the events that can be raised within it. Then click an event to specify what happens when the event is raised.

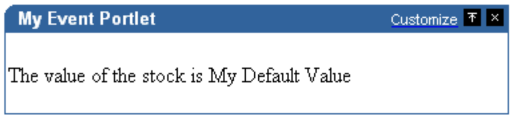
Portlet Events
Expand All | Collapse All
Masterclass for PDTS:Banner (no events)
My Parameter Portlet (no events)
My Event Portlet
My Event
My NLS portlet (no events)
Set Language (no events)
My NLS portlet (no events)
My Portlet (no events)
My2 Portlet (no events)
My2 Portlet (no

Event: My Event
No Description
Event Outputs
My Parameter
When this event is raised:
[✓ Go to page: Masterclass for PDTS]
Redisplay page: Masterclass for PDTS
Pass data to input parameters of page **Masterclass for PDTS**:
Page Input

stock_symbol =	Null	
Stock =	Null	
MyParameter =	Event Output	My Parameter

6. Click **OK** to view the page. Your Event portlet should have a link that displays the value received from the page.

Figure 5–42 My Event Portlet Before Parameter Change



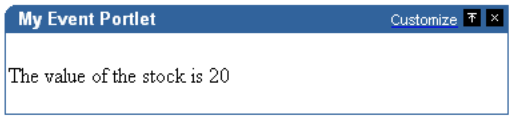
The value of the stock is My Default Value

7. You can append a parameter value to the URL and the portlet displays the value in the link.

```
&MyParameter=20
```

8. When you click the link, that value is passed to the Parameter portlet on its page.

Figure 5–43 My Event Portlet After Parameter Change



The value of the stock is 20

5.4.6 Accessing Session Information

When a user accesses any portal page, OracleAS Portal initiates a public unauthenticated session and maintains a cookie to track information about the session across requests. If the user logs in to OracleAS Portal, this session becomes an authenticated session of the logged-in user. This portal session terminates when:

- The browser session terminates (that is, the user closes all the browser windows).
- The user explicitly logs out.
- The session times out because the user's idle time exceeds the configured limit.

A portal session exists from the time the user first accesses the portal to the time the session ends in one of these ways.

You can utilize the session store to save and retrieve information that persists during the portal session. This information is only available, and useful, to you during the life of the session. You should only store temporary information in the session store. Application developers may use the session store to save information related to the current user session. Data in the session store can be shared across portlets.

If the information you want to store must persist across sessions, you may want to store it in the preference store instead. Some common applications of the session store are:

- to cache data that is expensive to load or calculate (for example, search results).
- to cache the current state of a portlet (for example, the current range, or page, of search results displayed in the portlet, or sequence of events performed by user).

Before you implement session storage, you should carefully consider the performance costs. Because portlets and providers are remote, it can be a relatively expensive operation to create and maintain even a small amount of information in the session store. For this reason, you may want to avoid altogether any session storage for public pages that are accessed frequently by many users.

Furthermore, while using the session store with Web providers, you create a stateful application that needs to track state information in memory. Similarly, you create a stateful application if you use the file-system implementation of preference store.

If scalability is an important concern for you, a stateful application may cause you problems. Stateful applications can impact the load-balancing and failover mechanism for your OracleAS Portal configuration. Even though you may deploy multiple middle-tiers accessing the same OracleAS Portal instance, you must implement sticky routing (where the same node handles subsequent requests in the same session) to track state. Sticky routing may result in lopsided load-balancing or loss of session data in case a node crashes, impacting failover. This issue is one reason why many developers prefer to build stateless applications. However, if scalability is not a concern, then a stateful application should present no problems for you.

In previous sections, you learned how to use the PDK Framework to render portlet content in various Render modes and how to implement features such as customization, and parameters and events. This section describes you how to implement session storage for your portlet.

In this section, session storage is used to count the number of times your portlet has rendered in Shared Screen mode.

5.4.6.1 Assumptions

1. You have followed through and understood Section 5.4.2, "Building PDK-Java Portlets".

2. You built a portlet using the wizard and successfully added it to a page.

5.4.6.2 Implementing Session Storage

The PDK Framework represents the session with a `ProviderSession` object, which is established during the call to the Provider Instance's `initSession` method. This object is associated with the `ProviderUser`. To make data persistent between requests from OracleAS Portal, you need to write data into the session object using the `setAttribute` method on the `ProviderSession` object. This method maps a `java.lang.Object` to a `java.lang.String` and stores that mapping inside the session object. The `String` can then be used to retrieve the `Object` during a subsequent request, provided the session is still valid.

A provider session may become invalid for the following reasons:

- session times out
- `invalidate` method on `ProviderSession` is called
- JVM process running the servlet container is terminated.

All portlets contained by the same `ProviderInstance` share the same session for a particular `ProviderUser`. Therefore, data unique to a particular portlet instance must be mapped to a unique `String` in the session. This is accomplished using the `portletParameter` method in the `PortletRendererUtil` class. This method makes a supplied `String` parameter or attribute name unique to a `PortletInstance`, by prefixing it with a generated identifier for that instance. The returned instance-specific name can be used to write portlet instance data into the session. For more detailed information on the PDK Framework classes, refer to the JavaDoc

To implement session storage, you need to perform the following tasks:

- Import `ProviderSession`, `PortletRendererUtil`, and `HttpPortletRendererUtil`.
- Retrieve the provider session.
- Read and write the session by accessing it from within your Java portlet.
- Set the session to true in `provider.xml`.
- Register the provider for session storage and set the Login Frequency.

The steps that follow describe how to add a session count to your portlet that displays how many times the portlet has been rendered for the current session.

1. After using the wizard to create a portlet, you can edit the JSP for the Show page in Oracle JDeveloper. You need to import the following classes:

```
<%@page contentType="text/html; charset=windows-1252"
import="oracle.portal.provider.v2.render.PortletRenderRequest"
import="oracle.portal.provider.v2.http.HttpCommonConstants"
import="oracle.portal.provider.v2.ProviderSession"
import="oracle.portal.provider.v2.render.PortletRendererUtil"
import="oracle.portal.provider.v2.render.http.HttpPortletRendererUtil"
%>
```

2. Insert code that checks for a valid session first and then increments the count and displays it. If the session is valid and a previously stored value exists, you display the value, increment the count, and store the new value. If the session is valid but no previously stored value exists, you initialize a new count starting with 1, and display and store the value. You also want to obtain the unique string key for this

portlet and then use an it in an array to count the session. If no session information was received, you want to provide information to the user indicating they may need to log back in.

```
<%
PortletRenderRequest pReq = (PortletRenderRequest)
request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
ProviderSession pSession = pReq.getSession();
  if (pSession != null)
  {
    String key = PortletRendererUtil.portletParameter(pReq, "count");
    Integer i = (Integer)pSession.getAttribute(key);
    if (i == null)
    {
      i = new Integer(0);
    }
    i = new Integer(i.intValue()+1);
    pSession.setAttribute(key, i);
%>

<p>Render count in this session: <%=i%> </p>

<%
  }
  else
  {
%>

<p>The session has become invalid</p>
<br>
Please log out and log in again.
<%
  }
%>
```

3. By default, the wizard does not set session to true in `provider.xml`. You need to update this flag in order for the provider to receive session information from the portal. You should only set this tag to true if you are using session information in your provider or portlets. By setting this flag to true, extra load is added to the provider calls.

```
<provider class="oracle.portal.provider.v2.DefaultProviderDefinition">
<session>true</session>
```

4. Register the provider for session support and set its Login Frequency to Once Per Session.

5.4.6.3 Viewing the Portlet

If you have not already added your Java portlet to a page, do so now. Ensure that you perform the following tasks:

- Set your provider to **Once per User Session** for the login frequency value.
- Refresh the provider to accept the new changes.
- Re-login in case your session is no longer valid.

5.4.7 Implementing Portlet Security

In the previous sections, you learned how to use the PDK Framework to render portlet content for various Show modes and how to implement features such as customization, parameters and events, and session storage. This section describes the available security services for your Java portlet. For more detailed information about the PDK classes referred to in this section, please refer to the **JavaDoc**.

5.4.7.1 Assumptions

1. You have followed through and understood Section 5.4.2, "Building PDK-Java Portlets".
2. You built a portlet using the wizard and successfully added it to a page.

5.4.7.2 Portlet Security Features

This section describes the various security features that are available to secure your portlet providers.

5.4.7.2.1 Authentication When a user first logs in to an OracleAS Portal instance, they must enter their password to verify their identity and obtain access. This authentication is performed by OracleAS Single Sign-On server. Refer to Section 5.4.7.3, "Single Sign-On" for more information.

5.4.7.2.2 Authorization Authorization determines if a particular user may view or interact with a portlet. OracleAS Portal provides two types of authorization checking:

- **Portal Access Control Lists (ACLs):** After you are authenticated by OracleAS Single Sign-On, OracleAS Portal uses ACLs to determine what users privileges you have to perform actions on portal objects, such as folders and portlets. The actions available to a user can range from simply viewing an object to performing administrative functions on it. If you do not belong to a group that has been granted a specific privilege, OracleAS Portal prevents you from performing the actions associated with that privilege. Refer to Section 5.4.7.4, "OracleAS Portal Access Control Lists (ACLs)" for more information.
- **Programmatic Portlet Security:** You can also implement your own security manager programmatically. Refer to Section 5.4.7.5, "Portlet Security Managers" for more information.

5.4.7.2.3 Communication Security To this point, we have covered user authentication and authorization, which do not check the authenticity of messages received by a provider. To completely secure your providers, secure the communication between OracleAS Portal and a Web provider. (These methods do not apply to database providers, which execute within the OracleAS Portal database.) If the communication is not secured, it is possible for someone to imitate an OracleAS Portal instance and fool the Web provider into returning sensitive information. There are three types of communication security:

- **OracleAS Portal Server Authentication** restricts access to a provider to a small number of recognized machines. This method compares the IP address or the host name of an incoming HTTP message with a list of trusted hosts. If the IP address or host name is in the list, the message is passed to the provider. If not, it is rejected before reaching the provider. Refer to Section 5.4.7.6, "OracleAS Portal Server Security" for more information.
- **Message Authentication** appends a checksum based on a shared key to provider messages. When a message is received by the provider, the authenticity of the

message is confirmed by calculating the expected value of the checksum and comparing it with the actual value received. If the values are the same, the message is accepted. If they are different, the message is rejected without further processing. The checksum includes a time stamp to reduce the chance of a message being illegally recorded in transit and resent later. Refer to Section 5.4.7.7, "Message Authentication" for more information.

- **Message Encryption** relies on the use of the HTTPS protocol for OracleAS Portal to provider communication. Messages are strongly encrypted to protect the data therein. Encryption provides a high level of security, but it incurs a performance penalty due to the additional processing required for each message. Refer to Section 5.4.7.8, "HTTPS Communication" for more information.

For more information about communication security, refer to the *Oracle Application Server Portal Configuration Guide*.

5.4.7.3 Single Sign-On

Portlets act as windows into an application. They display summary information and provide a way to access the full functionality of the application. Portlets expose application functionality directly in the portal or provide deep links that take you to the application itself to perform a task.

For more information about Single Sign-On, refer to the *Oracle Application Server Portal Configuration Guide*.

An application may need to authenticate the user accessing the application through the portlet. These are the possible application authentication methods:

- Partner Application. In this case, the application user is the same authenticated user used by OracleAS Portal.
- External Application. In this case, the OracleAS Portal user is different from the application user, but the application user name and password are managed by the OracleAS Portal user.
- No Application Authentication. In this case, the communication between provider and OracleAS Portal is not protected at all.

5.4.7.3.1 Partner Application A partner application is an application that shares the same OracleAS Single Sign-On as OracleAS Portal for its authentication. Thus, when a user is already logged in to OracleAS Portal, their identity can be asserted to the partner application without them having to log in again.

Partner applications are tightly integrated with OracleAS Single Sign-On. When a user attempts to access a partner application, the partner application delegates the authentication of the user to OracleAS Single Sign-On. Once a user is authenticated (that is, has provided a valid username and password) for one partner application, the user does not need to provide a username or password when accessing other partner applications that share the same OracleAS Single Sign-On instance. The OracleAS Single Sign-On determines that the user was successfully authenticated and indicates successful authentication to the new partner application.

The advantages of a partner application implementation are as follows:

- Provides the tightest integration with OracleAS Portal and OracleAS Single Sign-On Server.
- Provides the best single sign-on experience to users.
- Provides the most secure form of integration because user names and passwords are not transmitted between OracleAS Portal and the provider.

The disadvantages of a partner application implementation are as follows:

- The application must share the same user repository as OracleAS Portal even though the application's user community may be a subset of the OracleAS Portal user community. While worth some consideration, this issue is a minor one because the portal pages that expose the application can be easily restricted to the application's user community.
- The application can only be tightly integrated to one or more OracleAS Single Sign-On instances if they share the same user repository.
- The application must be written such that it delegates authentication to OracleAS Single Sign-On.
- You must have access to the application source code.

5.4.7.3.2 External Application An external application uses a different authentication server than OracleAS Portal. The application may use a different instance of Single Sign-On Server used by OracleAS Portal or some other authentication method. However the Single Sign-On Server does store the username and password of the external application for that user. This means that when a user is already logged into Oracle Portal, they will be logged into the external application without having to type in their username or password.

Applications that manage the authentication of users can be loosely integrated with OracleAS Single Sign-On if the administrator registers them as external applications. When a user who was previously authenticated by OracleAS Single Sign-On accesses an external application for the first time, OracleAS Single Sign-On attempts to authenticate the user with the external application. The authentication process submits an HTTP request that combines the registration information and the user's user name and password for the application. If the user has not yet registered their user name and password for the external application, OracleAS Single Sign-On prompts the user for the required information before making the authentication request. When a user supplies a user name and password for an external application, OracleAS Single Sign-On maps the new user name and password to the user's OracleAS Portal user name and stores them. They will be used the next time the user needs authentication with the external application.

The **advantages** of an external application implementation are as follows:

- Allows integration with many portals. If, however, one of the portals is preferred over the others, the application could be integrated as a partner application of that preferred portal and an external application of the others.
- Provides a single sign-on experience for users. However, users still must maintain different user names and passwords. In addition, the external application user name mapping must be maintained.
- Allows integration with multiple portals independent of their user repositories and OracleAS Single Sign-On.
- Avoids the requirement of having access to the application source code.

The **disadvantages** of an external application implementation are as follows:

- Does not share the same user repository as the portal, which requires additional maintenance of user information by the end user.
- Transmits the user name and password to the provider in plain text, unless you implement SSL.

5.4.7.3.3 No Application Authentication The provider trusts the OracleAS Portal instance sending the request completely. The provider can determine if the user is logged in and the portal user name, but the application has not authenticated the user.

The **advantages** of no application authentication are as follows:

- Provides the easiest form of integration and the fastest to implement.

The **disadvantages** of no application authentication are as follows:

- Provides the least security.
- Provides the weakest integration with OracleAS Portal.

5.4.7.4 OracleAS Portal Access Control Lists (ACLs)

When you log on to an OracleAS Portal instance, you are authenticated by an OracleAS Single Sign-On instance. Having verified your identity, OracleAS Portal uses ACLs to determine whether you are authorized to access particular portlets and add them to your pages from the Portlet Repository.

OracleAS Portal ACLs operate according to the following security characteristics:

- **Privileges** define the actions that can be performed on the object to which they are granted. Privileges include actions such as Manage and Execute.
- **OracleAS Portal users and their privileges** are granted from the Administer tab of the Builder.
- **OracleAS Portal user groups** are administered from the Administer tab of OracleAS Portal Builder. Membership in the groups and privileges granted to the groups are all defined and maintained here. A privilege granted to a user group is inherited by all the users of that group.
- **Provider privileges** apply to the provider and all of its portlets. Provider ACLs are administered on the Provider tab of the OracleAS Portal Navigator.
- **Portlet privileges** can override the privileges set for the provider of the portlet. Portlet ACLs are administered from the Provider tab of the OracleAS Portal Navigator. Clicking Open for a provider takes you to a page that manages the portlets of the provider.

For more information on the available privileges for each object, users, and user groups in OracleAS Portal, refer to the *Oracle Application Server Portal Configuration Guide*.

The **advantages** of ACLs are as follows:

- ACLs offer a simple, yet powerful, mechanism to secure OracleAS Portal objects.
- Central management of user group membership simplifies the management of ACLs because it negates the necessity of modifying the ACLs associated with each object.

The **disadvantages** of ACLs are as follows:

- ACLs are applied at the provider or portlet level. You cannot vary the security rules for a portlet depending on the page where you place it.

5.4.7.5 Portlet Security Managers

Portlet security managers are implemented within a provider to verify that a given users may view an instance of the portlet. When a user views a page with a portlet instance on it, security managers determine whether the user has the appropriate privileges to see the portlet. Implementing access control methods in the provider restricts the retrieval of content from a portlet (that is, hides the portlet) from users without the appropriate privileges. Only if the specified characteristics, such as user details and preferences, pass the authorization logic will the content be retrieved for the user. If no portlet security methods are implemented in the provider, then any user name may be passed in, even fictitious, unauthenticated ones.

A provider can implement two portlet security methods:

- Get a list of portlets.
- Verify the accessibility of the portlet.

Portlets have access to the OracleAS Portal user privileges and groups of which the user is a member. The following information can be used by the security methods:

- The default group of the user
- The privileges of a user or group
- The highest available privilege of a user across all groups
- The objects the user can access (only in database providers)

The `AuthLevelSecurityManager` has access to the following information about authorization level:

- Strongly authenticated.

  The user has been authenticated by OracleAS Single Sign-On in the current OracleAS Portal session (that is, the user logged in with a valid user name and password) and requested the portlet in the context of that session.

- Weakly authenticated.

  A user who was previously strongly authenticated returns to view a page without an active OracleAS Portal session. A persistent cookie (maintained by the user's browser) indicates that in some previous session the user logged on with a valid user name and password.

- Public or not authenticated.

  The user has not logged in within the context of the current OracleAS Portal session, and does not have a persistent cookie to indicate that such a state previously existed.

To incorporate these security services into your Java portlet, you simply need to update `provider.xml` and set the security level to strong, weak, or public. Place the following XML right above the `</portlet>` tag in `provider.xml`:

```
<securityManager class="oracle.portal.provider.v2.security.AuthLevelSecurityManager">
   <securityLevel>strong</securityLevel>
</securityManager>
```

After you make this change to `provider.xml`, refresh the provider.

The **advantages** of security methods are as follows:

- Enable a portlet to produce different output depending on the level of authorization.

The **disadvantages** of security methods are as follows:

- Most security manager implementations will use the authorization level or some other user specific element in an incoming message. A check of this type could be bypassed by an entity imitating an OracleAS Portal instance.

5.4.7.5.1 Viewing the Portlet

To demonstrate the behavior of the security manager added to your Java portlet, follow these steps:

1. Ensure you are logged in to an OracleAS Portal instance with privileges to create pages and add portlets to a page.
2. Create a new portal page, ensuring it is visible to PUBLIC.
3. Add your Java portlet to the page.
4. Make a note of the direct URL to your new Portal page.
5. Now log out of the Portal instance by clicking the **Logout** link.
6. Directly access the Portal page by entering the URL noted in Step 4 into your browser's address bar.

You will see the page created in Step 2 but not the portlet added in Step 3. When you added the portlet to the page, you were logged in and hence strongly authenticated. The PDK runtime detected this and allowed you to add the portlet. When you logged out and viewed the page, you were no longer strongly authenticated and hence the PDK Framework did not allow rendering of the portlet's contents.

If you log in again and view the page, you will see that the portlet is still there.

5.4.7.5.2 Implementing Your Own Security Manager If your portlet requires special security arrangements which are not provided by the implementations shipped with the PDK, you will need to supply your own custom `PortletSecurityManager` controller class. To do this, simply extend the `oracle.portal.provider.v2.security.PortletSecurityManager` class and supply implementations for the two methods specified by the interface. Then simply replace the class attribute of the `securityManager` controller element in the XML provider definition with you new class name and configure child elements appropriately.

5.4.7.6 OracleAS Portal Server Security

One way to prevent unauthorized access to providers is to restrict access to the provider to known client machines at the server level. This method goes some way toward defending against denial of service attacks.

In Oracle Application Server, you achieve this goal by using the allow and deny directives in the `httpd.conf` file to control access to client machines based on their host names or IP addresses. If host names are used as discriminators, the server needs to look them up on its Domain Name Server (DNS), which adds extra overhead to the processing of each request. Using the IP address circumvents this problem, but the IP address of a remote client may change without warning.

The advantages of server security are as follows:

- Limits access to the provider to trusted hosts only.
- Simplifies configuration.

The disadvantages of server security are as follows:

- OracleAS Web Cache does not have IP address checking capability. If OracleAS Web Cache sits in front of a provider, you have no protection from a client on any host sending show requests to OracleAS Web Cache.
- Restricting access to certain IP addresses and host names may be circumvented by sending messages to a provider containing fake IP addresses and host names. This

trick is difficult to perform effectively since return messages go to the machine whose IP address was copied, but it can still cause problems.

For more information on this topic, refer to the *Oracle Application Server Portal Configuration Guide*.

5.4.7.7 Message Authentication

PDK-Java supports message authentication so that access may be limited to a specified provider instance or group of provider instances. A provider is registered with a secret shared key known only to OracleAS Portal and provider administrators.

OracleAS Portal sends a digital signature, calculated using a Hashed Message Authentication Code (HMAC) algorithm, with each message to a provider. A provider may authenticate the message by checking the signature using its own copy of the shared key. This technique may be used in Secure Socket Layer (SSL) communication with a provider instead of client certificates.

OracleAS Portal calculates a signature based on user information, a shared key and a time stamp. The signature and time stamp are then sent as part of the SOAP message. The time stamp is based on UTC (coordinated universal time, the scientific name for Greenwich Mean Time) so that timestamps can be used in messages between computers in different time zones.

When the provider receives this message it generates its own copy of the signature. If the signatures agree, it will then compare the message time stamp with the current time. If the difference between the two is within an acceptable value the message is considered authentic and is processed accordingly.

A single provider instance cannot support more than one shared key because it could cause security and administration problems. For instance, if one copy of the shared key is compromised in some way, the provider administrator has to create a new key and distribute it to all of the OracleAS Portal clients, who then must update their provider definitions. The way around this problem is to deploy different provider services, specifying a unique shared key for each service. Each provider service has its own deployment properties file so that each service is configured independently of the others. The overhead of deploying multiple provider services within the same provider adapter is relatively small.

In a provider without OracleAS Web Cache in front of it, this use of the same signature cookie over the lifetime of a provider session implies a tradeoff between performance and the security provided by authenticating the requests. The signature cookie value is only calculated once after the initial SOAP request establishes the session with the provider. The shorter the provider session timeout, the more often a signature will be calculated providing greater security against a show request being resent illegally. However, the SOAP request required to establish a session incurs a time penalty.

In a provider using OracleAS Web Cache to cache show request responses, you have a similar tradeoff. Cached content is secured in the sense that incoming requests must include the signature cookie to retrieve it, but caching content for an extended period of time leaves the provider open to illegal show requests.

While the signature element provides protection against interception and resending of messages, it does nothing to prevent interception and reading of message contents. Messages are still transmitted in plain text. If you are concerned about the content of messages being read by unauthorized people, you should use message authentication in conjunction with SSL.

The advantages of message authentication are as follows:

- Ensures that the message received by a provider comes from a legitimate OracleAS Portal instance.

The disadvantages of message authentication are as follows:

- Causes administration problems if a provider serves more than one portal.
- Entails performance implications if made very secure by having a short session timeout.

For more information on this topic, refer to the *Oracle Application Server Portal Configuration Guide*.

5.4.7.8 HTTPS Communication

Normal communication between OracleAS Portal and a provider uses HTTP, a network protocol that transmits data as plain text using TCP as the transport layer. HTTPS uses an extra secured layer (SSL) on top of TCP to secure communication between a client and a server, making it difficult to intercept and read messages.

Each entity (for example, an OracleAS Web Cache instance) receiving a communication using SSL has a freely available public key and a private key known only to the entity itself. Any messages sent to an entity are encrypted with its public key. A message encrypted by the public key may only be decrypted by the private key so that, even if a message is intercepted by a felonious third party, it cannot be decrypted.

Certificates used to sign communications ensure that the public key does in fact belong to the correct entity. These are issued by trusted third parties, known as Certification Authorities (CA). They contain an entity's name, public key, and other security credentials and are installed on the server end of an SSL communication to verify the identity of the server. Client certificates may also be installed on the client to verify the identity of a client.

Oracle Wallet Manager manages public key security credentials. It generates public and private key pairs, creates a certificate request to a CA, and installs the certificate on a server.

For more information on this topic, refer to the *Oracle Application Server Portal Configuration Guide*.

5.4.7.8.1 Configuration of SSL When a provider is registered from an OracleAS Portal instance, only one URL is entered, which means either HTTP or HTTPS may be used but not both.

Each port on each server that may be used to receive SSL messages must have a server side certificate installed (that is, the OracleAS Web Cache instance (if any)) in front of the Web provider and the server which hosts the provider. The certificate installed on a server port ensures that communication between two points is encrypted but does not authenticate the source of a message. Message authentication should be used as well to fully secure communication between a trusted OracleAS Portal instance and a provider.

For more information about SSL configuration for OracleAS Portal, refer to the *Oracle Application Server Portal Configuration Guide*.

5.4.7.9 LDAP (Oracle Internet Directory) Security

PDK-Java uses Portlet Security Managers for LDAP (Oracle Internet Directory) security. PDK-Java uses Oracle Internet Directory as a repository of users, groups, and permissions. It retrieves information about the logged-in user and determines whether the user has the required permissions to view the portlet and data within the portlet. By enabling Oracle Internet Directory security, your providers can:

- Secure portlets based on groups.
- Restrict access to the administrative functions of your portlets (using your own security manager).
- Retrieve all of the user property information stored in the Oracle Internet Directory including first name, last name, title, email, telephone number, groups, and photo.
- Create users and groups for OracleAS Portal.

By default, Oracle Internet Directory security is disabled. You must make a change in the deployment properties file for a specific provider to enable this feature. Enabling and using Oracle Internet Directory to secure your portlets can be done quickly and easily:

1. Enable the Oracle Internet Directory manager in the deployment properties files (`provider_name.properties`).

   ```
   oidManager=true
   oidAdminClass=class_that_extends_oracle.portal.provider.v2.oid.OidInfo
   ```

2. Provide the connection information for Oracle Internet Directory by extending the simple class called `OidInfo`.
3. Provide a list of groups that can view your portlet in the provider definition file.

   ```
   <group>cn=group1,cn=groups,dc=us,dc=oracle,dc=com</group>
   ```

   Your provider connects to Oracle Internet Directory using the information provided to the OidInfo class by you. The portlet accesses Oracle Internet Directory using the credentials provided (for example, user name and password) and performs the specified tasks. We recommend that you create an Oracle Internet Directory user specifically for your provider connection with the minimum set of privileges needed to complete the tasks requested by your portlets. For example, if your portlet only checks group information, do not connect to the Oracle Internet Directory as an administrator.

5.4.7.9.1 Implementing Oracle Internet Directory Security

PDK-Java provides a set of default classes specifically for Oracle Internet Directory integration. These classes handle the connection from your portlets to Oracle Internet Directory, enable your portlets to be secured based on OracleAS Portal groups, and provide access to user property information from within Oracle Internet Directory. The classes used by your Web provider for Oracle Internet Directory integration:

- oracle.portal.provider.v2.oid.OidInfo receives the Oracle Internet Directory connection information provided by the developer and connects to Oracle Internet Directory. When building your own portlets, you should extend this class to send secure connection details from the provider to Oracle Internet Directory.
- `oracle.portal.sample.v2.devguide.oid.UnsafeOidInfo` is an extension of `OidInfo` and provides an easy way to test portlet security. This class is used by the Oracle Internet Directory samples in PDK-Java and parses the deployment properties file for the Oracle Internet Directory connection information (seen below). This class should only be used for testing and development, it is not safe to use in a production scenario.
- `oidManager` is set to false by default. It must be set to true in *`provider_name`*`.properties` to enable Oracle Internet Directory. (If you have only one

provider in your Web application, ensure that `provider_name.properties` is identical to `_default.properties`.) For example:

```
serviceClass=oracle.webdb.provider.v2.adapter.soapV1.ProviderAdapter
loaderClass=oracle.portal.provider.v2.http.DefaultProviderLoader
showTestPage=true
definition=providers/lab_provider/provider.xml
autoReload=true
oidManager=true
oidAdminClass=oracle.portal.sample.v2.devguide.oid.UnsafeOidInfo
oidHost=myhost.mydomain.com
oidPort=oidPort
oidUser=oidUser
oidPasswd=oidPassword
```

- `oidAdminClass` is set to the class that extends `OidInfo`. PDK-Java provides `UnsafeOidInfo` by default, but as the name suggests, this class should not be used in production scenarios.
  - `oidHost` is the machine where Oracle Internet Directory is hosted.
  - `oidPort` is the port used by the Oracle Internet Directory.
  - `oidUser` is the Oracle Internet Directory account.
  - `oidPasswd` is the Oracle Internet Directory password.

  For example:

```
serviceClass=oracle.webdb.provider.v2.adapter.soapV1.ProviderAdapter
loaderClass=oracle.portal.provider.v2.http.DefaultProviderLoader
showTestPage=true
definition=providers/lab_provider/provider.xml
autoReload=true
oidManager=true
oidAdminClass=oracle.portal.sample.v2.devguide.oid.UnsafeOidInfo
oidHost=myhost.mydomain.com
oidPort=oidPort
oidUser=oidUser
oidPasswd=oidPassword
```

- `oracle.portal.provider.v2.security.GroupSecurityManager` manages which groups have access to your provider and its portlets. It retrieves this information from the provider definition file and is portlet specific. Each portlet in a provider may have different group settings. There is no limit on the number of groups that can be set using this tag, but, since the Web provider parses and validates each group in turn, listing many groups may degrade performance.
- <group> is the tag in `provider.xml` that handles group management. It lists the groups allowed to access the portlet. The group information here follows the same case sensitivity as the Oracle Internet Directory.

> **Note:** The following example refers to your *`portal_instance_id`*, which is specific to your installation. To find your instance identifier, refer to your *Oracle Internet Directory Administrator's Guide*.

```
<securityManager class="oracle.portal.provider.v2.security.
    GroupSecurityManager">
      <group>cn=DBA,cn=portal_instance_id,cn=groups,
             dc=us,dc=oracle,dc=com</group>
```

```
</securityManager>
```

The **advantages** of Oracle Internet Directory security are as follows:

- Offers a simple, powerful way to secure your portlets.
- Secures data within your portlets based on the user's group membership.
- Creates users and groups directly from your portlets exposed as Web providers.

The **disadvantages** of Oracle Internet Directory security are as follows:

- Slightly degrades performance when authorizing your portlet through Oracle Internet Directory. There is a cost associated with obtaining group information from any LDAP server, but this cost only happens the first time a user accesses a portlet in a session.
- Requires provider access to Oracle Internet Directory.
- Assumes all OracleAS Portal instances served by the provider use the same Oracle Internet Directory instance.

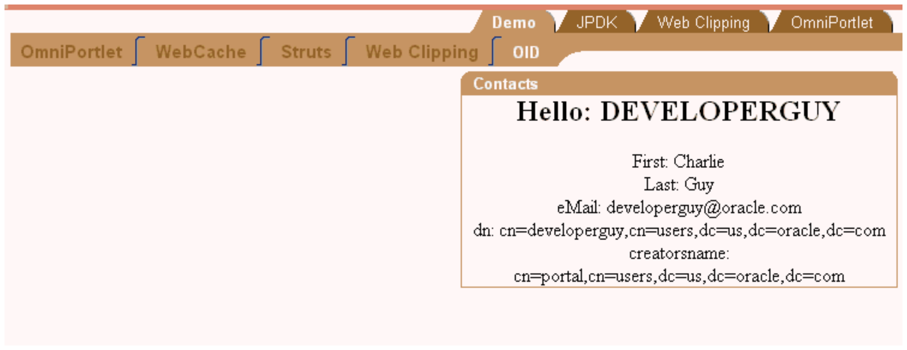
For more information on securing your providers using Oracle Internet Directory or to set up the sample portlets secured using Oracle Internet Directory, review the technical note, *Installing the OID Portlets*.

5.4.7.9.2 Viewing Your Portlets To demonstrate the behavior of the security manager added to your Java portlet, follow these steps:

1. Ensure you are logged in to an OracleAS Portal instance as a user who is a member of the group specified in the `<group>` tag in `provider.xml`.
2. Use an existing page or create a new one, ensuring it is visible to PUBLIC.
3. Add your Java portlet to the page.
4. Make a note of the direct URL to your new page.
5. Click **Logout**.
6. Directly access the page by entering the URL noted in Step 4 in your browser's address bar or login to OracleAS Portal using a user that is not part of the group listed in `provider.xml`.

You will see the page created in Step 2 but not the portlet added in Step 3. When you added the portlet to the page, you were logged in as a user authorized to view the portlet. The PDK runtime detected this and allowed you to add the portlet. When you logged out and viewed the page, you were no longer part of the group allowed to view the portlet and hence the PDK Framework did not allow rendering of the portlet's contents.

Figure 5–44 Page and Portlets for Developer

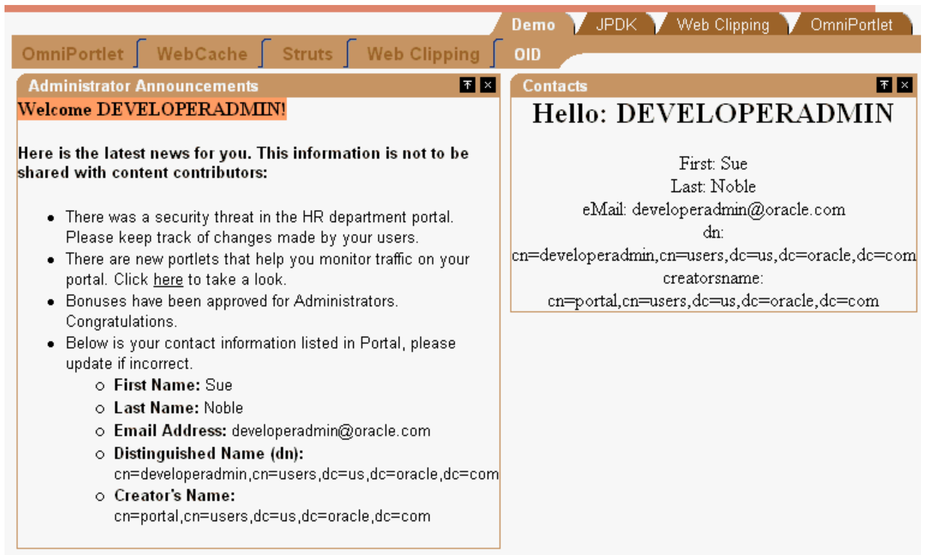


Hello: DEVELOPERGUY

First: Charlie
Last: Guy
eMail: developerguy@oracle.com
dn: cn=developerguy,cn=users,dc=us,dc=oracle,dc=com
creatorsname:
cn=portal,cn=users,dc=us,dc=oracle,dc=com

If you log in again and view the page, you will see that the portlet is still there.

Figure 5–45 Page and Portlets for Developer/Administrator



Welcome DEVELOPERADMIN!

Here is the latest news for you. This information is not to be shared with content contributors:

- There was a security threat in the HR department portal. Please keep track of changes made by your users.
- There are new portlets that help you monitor traffic on your portal. Click here to take a look.
- Bonuses have been approved for Administrators. Congratulations.
- Below is your contact information listed in Portal, please update if incorrect.
  - **First Name:** Sue
  - **Last Name:** Noble
  - **Email Address:** developeradmin@oracle.com
  - **Distinguished Name (dn):** cn=developeradmin,cn=users,dc=us,dc=oracle,dc=com
  - **Creator's Name:** cn=portal,cn=users,dc=us,dc=oracle,dc=com

Hello: DEVELOPERADMIN

First: Sue
Last: Noble
eMail: developeradmin@oracle.com
dn:
cn=developeradmin,cn=users,dc=us,dc=oracle,dc=com
creatorsname:
cn=portal,cn=users,dc=us,dc=oracle,dc=com

5.4.8 Controlling the Export/Import of Portlet Customizations

The export/import facility of OracleAS Portal is a multi-purpose tool for moving your portal objects, such as portlets, between instances of OracleAS Portal. For example, you might use export/import to move objects from a development environment to a stage environment and then, finally, to a production environment. You might also use export/import to move pages and page groups between OracleAS Portal instances, or to move Web providers from one machine to another. For more information about export/import in general, please refer to the *Oracle Application Server Portal Configuration Guide*.

Because portlet default settings can be set by the administrator and then changed by the user, they require some special consideration when you import and export them. To simplify the transport process, OracleAS Portal provides default functionality that handles administrator customization data (that is, data created through Edit Defaults mode) for you. When a portlet is exported, the default customization data stored using PDK-Java's `PreferenceStore` mechanism is exported with the portlet by default. Hence, when the portlet is imported into a target instance of OracleAS Portal, this data is imported along with it. As a result, the portlet instance's default settings are maintained when the portlet is moved from one portal instance to another.[3]

The aforementioned behavior is provided to you as a convenience and it requires no action on your part to leverage. You might, however, want to exercise more granular control over the exportation of customization data than that provided by the default functionality. To implement your own requirements for export/import, you can make use of the programming interface to augment or override the default handling of customizations.

The export/import functionality for customizations requires that your OracleAS Portal instance and provider are on Release 10.1.2. Export/import of customizations behaves the same regardless of the location of your provider:

- in the default Oracle Application Server Containers for J2EE of the Oracle Application Server, where the OracleAS Portal instance is different.
- in a separate Oracle Application Server Containers for J2EE, where the OracleAS Portal instance may be different, and the provider is the same but is not registered on the target OracleAS Portal instance.

5.4.8.1 Import/Export Programming Interface

The PDK-Java's preference store mechanism allows data to be persisted by any number of application entities. The following three entities are the ones that persist data for the purposes of export/import:

1. The **portlet instance** is the portlet on a page with the default customizations made to it by the administrator. The API for the portlet instance is:
   - oracle.portal.provider.v2.PortletInstance
     - exportData

       ```
       public byte[] exportData
           (
               boolean exportUsers,
               String[] userNames,
               TransportLogger logger
           )
           throws PortletException
       ```

     - importData

       ```
       public void importData
           (
               byte[] data,
               TransportLogger logger
           )
           throws PortletException
       ```

2. The **portlet definition** is the base portlet without any customizations applied to it. You might think of the portlet definition as the version of the portlet that exists in the Portlet Repository before it is placed on a particular page for use. The API for the portlet definition is:
   - oracle.portal.provider.v2.PortletDefinition
     - exportData

       ```
       public byte[] exportData
           (
               ProviderInstance pi,
               boolean exportUsers,
       ```

[3] User customization data for OracleAS Portal objects is never exported. This restriction applies to portlets as well as other objects, such as pages.

```
            String[] userNames,
            TransportLogger logger
        )
        throws PortletException
```

- importData

```
public void importData
    (
        ProviderInstance pi,
        byte[] data,
        TransportLogger logger
    )
    throws PortletException
```

3. The **provider instance** is the entity that contains and communicates with a set of portlets. The API for the provider instance is:

   - oracle.portal.provider.v2.ProviderInstance

     – exportData

```
public byte[] exportData
    (
        boolean exportUsers,
        String[] userNames,
        TransportLogger logger
    )
    throws ProviderException
```

     – importData

```
public void importData
    (
        byte[] data,
        TransportLogger logger
    )
    throws ProviderException
```

By default, each of the above entities employs an instance of `oracle.portal.provider.v2.transport.PrefStoreTransporter` to transform the data from an `oracle.portal.provider.v2.preference.PreferenceStore` to a byte array for transport. For the default export/import behavior, though, only the portlet instance entity's customization data is exported and imported. If you have persisted data at the portlet definition or provider instance level, you may want to export that data as well. For example, a billing handle that you persisted at the `ProviderInstance` level may need to be exported.

To change the behavior of PrefStoreTransporter, you can override its default implementation. The example in Section 5.4.8.3.7, "Exporting by Reference Example" illustrates how you can override PrefStoreTransporter.

5.4.8.1.1 Logging Interface To simplify troubleshooting of your export/import transactions, you can send messages to both the calling OracleAS Portal instance and the Web provider log. PDK-Java provides a transport logging class that enables you to add events to the log during export and import operations. In this way, you can better keep track of events that occur during the transport of portlet customizations. The log can be a valuable troubleshooting tool if you encounter unexpected behavior in your portlets during or after transport. For example, you can log events when incompatibilities between PDK-Java versions are found.

You log events using the logger object, an instance of the `oracle.portal.provider.v2.transport.TransportLogger` class provided for each of the methods mentioned above. You log events with the calling portal through the instance provided for each method. You record events in the Web provider log with the normal logging mechanism, `oracle.portal.log.LogManager`. The log levels for export/import are as follows:

- `TransportLogger.SEVERITY_INFO`
- `TransportLogger.SEVERITY_WARNING`
- `TransportLogger.SEVERITY_ERROR`

5.4.8.2 Exporting Customizations Example

This example illustrates the most basic case where you build a portlet and accept the default behavior for the export of customizations. In the examples in Section 5.4.8.3.6, "Encrypting Customization Data Example" and Section 5.4.8.3.7, "Exporting by Reference Example", you will see how to enhance the security of your customizations during export and import. To implement the more basic form of exporting customizations, do the following:

1. Create a stock portlet and implement the Show mode with the following `MyStockPortletShowRenderer.java` class. Note that this class does not incorporate any special code to enable export/import.

```
package oracle.portal.sample.v2.devguide.tx;
import java.util.StringTokenizer;
import oracle.portal.provider.v2.PortletException;
import oracle.portal.provider.v2.personalize.NameValuePersonalizationObject;
import oracle.portal.provider.v2.render.PortletRenderRequest;
import oracle.portal.provider.v2.render.PortletRendererUtil;
import oracle.portal.provider.v2.render.http.BaseManagedRenderer;
import java.io.PrintWriter;
import oracle.portal.sample.v2.devguide.webservices.
    NetXmethodsServicesStockquoteStockQuoteServiceStub;
public class MyStockPortletShowRenderer extends BaseManagedRenderer
{
    private String pid = null;
    private String userdata;
    private String stockList;
    private String stockCode;
  public void renderBody(PortletRenderRequest request) throws PortletException
  {
    // Use the PrintWriter from the PortletRenderRequest
    PrintWriter out = null;
    NetXmethodsServicesStockquoteStockQuoteServiceStub ns = new
      NetXmethodsServicesStockquoteStockQuoteServiceStub();
    try
    {
      out = request.getWriter();
      NameValuePersonalizationObject data = null;
      data = (NameValuePersonalizationObject)PortletRendererUtil.
         getEditDefaultData(request);
      stockList= data.getString("stock");
      if(stockList!=null) {
        StringTokenizer st = new  StringTokenizer(stockList,",");
        out.println("<table border='0'>");
        out.println("<thead>");
        out.println("<tr>");
        out.println("<th width='20%'>");
```

```
          out.println("<p align='left'> Stock Code</p></th><th width='20%'>");
          out.println("<p align='left'> Quote</p>");
          out.println("</th>");
          out.println("</tr>");
          out.println("<thead>");
              while(st.hasMoreElements()) {
                stockCode= st.nextElement().toString();
                out.println("<tr>");
                out.println("<td width='20%'>");
                out.println("<p align='left'>"+  stockCode +
                   "</p></td><td width='20%'>");
                out.println(ns.getQuote(stockCode));
                out.println("</td>");
                out.println("</tr>");
              }
          out.println("</table>");
        }
        else
        {
         out.println("<br> Click <b>Edit Defaults</b> to define stock codes.");
        }
      }
      catch(Exception ioe)
      {
        throw new PortletException(ioe);
      }
    }
  }
```

2. Implement the Edit Defaults mode for your stock portlet with the following class, `MyStockPortletEditDefaultsRenderer.java`. This class enables the administrator to make and store default customizations, which are then exported according to the default behavior.

```
package oracle.portal.sample.v2.devguide.tx;
import oracle.portal.provider.v2.PortletException;
import oracle.portal.provider.v2.http.HttpCommonConstants;
import oracle.portal.provider.v2.render.PortletRenderRequest;
import oracle.portal.provider.v2.render.http.BaseManagedRenderer;
import oracle.portal.provider.v2.render.PortletRendererUtil;
import oracle.portal.provider.v2.personalize.NameValuePersonalizationObject;
import java.io.PrintWriter;
import java.io.IOException;
import oracle.portal.provider.v2.render.http.HttpPortletRendererUtil;
public class MyStockPortletEditDefaultsRenderer extends BaseManagedRenderer
{
  public void renderBody(PortletRenderRequest request) throws PortletException
  {
    PrintWriter out = null;
    try
    {
      out = request.getWriter();
    }
    catch(IOException ioe)
    {
      throw new PortletException(ioe);
    }

    // Customize the portlet title and stock
    String actionParam = PortletRendererUtil.getEditFormParameter(request);
```

```
      PortletRenderRequest prr = (PortletRenderRequest)
          request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
      String action = request.getParameter(actionParam);
      String title = prr.getQualifiedParameter("myportlet_title");
      String stock = prr.getQualifiedParameter("myportlet_stock");
      NameValuePersonalizationObject data = null;
      try
      {
        data = (NameValuePersonalizationObject)
                PortletRendererUtil.getEditDefaultData(request);
      }
      catch(IOException io)
      {
        throw new PortletException(io);
      }
      // Cancel automatically redirects to the page, so
      // will only recieve OK or APPLY
      if (action != null)
      {
        data.setPortletTitle(title);
        data.putString("stock",stock);
        try
        {
          PortletRendererUtil.submitEditData(request, data);
        }
        catch(IOException ioe)
        {
          throw new PortletException(ioe);
        }
        return;
      }
      // Otherwise just render the form
      title = data.getPortletTitle();
      stock = data.getString("stock");
      out.print("<table border='0'> <tr> ");
      out.println("<td width='20%'> <p align='right'>Title:</p></td>
                   <td width='80%'>");
      out.print("<input type='TEXT' name='" +
                 HttpPortletRendererUtil.portletParameter(prr, "myportlet_title")
                 + "' value='" + title + "'>");
      out.println("</td> </tr>");
      out.print("<tr> <td width='20%'> <p align='right'>Stock Codes:</p></td>
                 <td width='80%'>");
      out.print("<input type='TEXT' name='" +
                 HttpPortletRendererUtil.portletParameter(prr, "myportlet_stock")
                 + "' value='" + stock + "'>");
      out.println("<br> For example use US Stock Codes separated by comma:
                   <i> SUNW,IBM,ORCL</i>");
      out.print("</td> </tr>");
      out.println("</table>");
    }
  }
```

3. Create the following class, `NetXmethodsServicesStockquoteStockQuoteServiceStub.java`, for your stock portlet:

```
package oracle.portal.sample.v2.devguide.webservices;
import oracle.soap.transport.http.OracleSOAPHTTPConnection;
import org.apache.soap.encoding.SOAPMappingRegistry;
import java.net.URL;
import org.apache.soap.rpc.Call;
import org.apache.soap.Constants;
import java.util.Vector;
import org.apache.soap.rpc.Parameter;
import org.apache.soap.rpc.Response;
import org.apache.soap.Fault;
import org.apache.soap.SOAPException;
import java.util.Properties;
public class NetXmethodsServicesStockquoteStockQuoteServiceStub
{
  public NetXmethodsServicesStockquoteStockQuoteServiceStub()
  {
    m_httpConnection = new OracleSOAPHTTPConnection();
    m_smr = new SOAPMappingRegistry();
  }
  private String _endpoint = "http://64.124.140.30:9090/soap";
  public String getEndpoint()
  {
    return _endpoint;
  }
  public void setEndpoint(String endpoint)
  {
    _endpoint = endpoint;
  }
  private OracleSOAPHTTPConnection m_httpConnection = null;
  private SOAPMappingRegistry m_smr = null;
  public Float getQuote(String symbol) throws Exception
  {
    Float returnVal = null;
    URL endpointURL = new URL(_endpoint);
    Call call = new Call();
    call.setSOAPTransport(m_httpConnection);
    call.setTargetObjectURI("urn:xmethods-delayed-quotes");
    call.setMethodName("getQuote");
    call.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);
    Vector params = new Vector();
    params.addElement(new Parameter("symbol", String.class, symbol, null));
    call.setParams(params);
    call.setSOAPMappingRegistry(m_smr);
    Response response = call.invoke(endpointURL,
      "urn:xmethods-delayed-quotes#getQuote");
    if (!response.generatedFault())
    {
      Parameter result = response.getReturnValue();
      returnVal = (Float)result.getValue();
    }
    else
    {
      Fault fault = response.getFault();
      throw new SOAPException(fault.getFaultCode(), fault.getFaultString());
    }
    return returnVal;
  }
  public void setMaintainSession(boolean maintainSession)
  {
    m_httpConnection.setMaintainSession(maintainSession);
  }
  public boolean getMaintainSession()
  {
    return m_httpConnection.getMaintainSession();
  }
  public void setTransportProperties(Properties props)
  {
    m_httpConnection.setProperties(props);
  }
  public Properties getTransportProperties()
  {
    return m_httpConnection.getProperties();
  }
}
```

4. Create a Web provider through `provider.xml` for this portlet. Notice the use of the `<preferenceStore>` element to allow for the storing of customizations:

```
<provider class="oracle.portal.provider.v2.DefaultProviderDefinition">
   <session>false</session>
   <passAllUrlParams>false</passAllUrlParams>
   <preferenceStore class="oracle.portal.provider.
                           v2.preference.FilePreferenceStore">
      <name>prefStore1</name>
      <useHashing>true</useHashing>
   </preferenceStore>
   <portlet class="oracle.portal.provider.v2.DefaultPortletDefinition">
      <id>1</id>
      <name>MyStockPortlet</name>
      <title>My Stock Portlet</title>
      <description>Simple Stock Portlet to show Export and Import
                   feature of web providers</description>
      <timeout>80</timeout>
      <showEditToPublic>false</showEditToPublic>
      <hasAbout>false</hasAbout>
      <showEdit>false</showEdit>
      <hasHelp>false</hasHelp>
      <showEditDefault>true</showEditDefault>
      <showDetails>false</showDetails>
      <renderer class="oracle.portal.provider.v2.render.RenderManager">
         <renderContainer>true</renderContainer>
         <renderCustomize>true</renderCustomize>
         <autoRedirect>true</autoRedirect>
         <contentType>text/html</contentType>
         <showPage class="oracle.portal.sample.v2.
                          devguide.tx.MyStockPortletShowRenderer"/>
         <editDefaultsPage class="oracle.portal.sample.v2.devguide.tx.
                                 MyStockPortletEditDefaultsRenderer"/>
      </renderer>
      <personalizationManager class="oracle.portal.provider.v2.personalize.
                                     PrefStorePersonalizationManager">
         <dataClass>oracle.portal.provider.v2.personalize.
                    NameValuePersonalizationObject
         </dataClass>
      </personalizationManager>
   </portlet>
</provider>
```

5. Register this export-enabled provider with the source OracleAS Portal instance. For more information about registering Web providers, refer to Section 5.4.2.5, "Registering and Viewing Your Portlet".

Note: If the Web provider is running in a secured environment, remember to provide the proxy host and port while starting up Oracle Application Server Containers for J2EE. For example:

```
java -Dhttp.proxyHost=www-proxy.us.oracle.com -Dhttp.proxyPort=80
     -jar oc4j.jar
```

6. Create two regions on a sample page and add **My Stock Portlet** to the first region. For information on creating regions and pages, refer to the *Oracle Application Server Portal User's Guide*.
7. Edit the page and choose **Edit Defaults** for **My Stock Portlet**. Choose the stock codes `SUNW, IBM, ORCL`. For more information on how to edit defaults for a portlet on a page, refer to the *Oracle Application Server Portal User's Guide*.
8. Add **My Stock Portlet** to a second region and again edit the defaults. for it Use a different stock code this time, `MSFT`.
9. Export the page group containing this page. For instructions on how to export a page group, refer to Chapter 10, "Exporting and Importing Content," in the *Oracle Application Server Portal Configuration Guide*.
10. Import the page group into a target OracleAS Portal instance. For instructions on how to import a page group, refer to Chapter 10, "Exporting and Importing Content," in the *Oracle Application Server Portal Configuration Guide*.
11. View the page with **My Stock Portlet** in the target OracleAS Portal instance and ensure that the customizations were maintained.

5.4.8.3 Implementing Security for Export/Import

Transporting customizations can present a security concern if your portlet stores sensitive data and is not operating in a secured environment. At the provider and portlet level, OracleAS Portal provides several ways for you to secure the export and subsequent import of portlet customizations. To better secure portlets and providers for exportation and importation, you can take the following actions:

- Securing Provider Communications. Using OracleAS Portal configuration options, you can secure the communications between providers and OracleAS Portal. This step in turn makes the export/import of portlets more secure.
- Disabling Export/Import of Customizations. You can disable the export of all portlet customization data on a per Web application basis. This method provides the greatest security but only at a significant cost in functionality because it prevents administrators from retaining their default customizations when the portlet is moved.
- Obfuscating Data for Transport (Automatic). By default, OracleAS Portal obfuscates but does not encrypt customization data before transporting it.
- Encrypting Customization Data for Transport. You may want to encrypt customization data for transport if any of the following are true:
  - Your Web provider connection is not secured through HTTPS.
  - You want to ensure the data is secured during transit.
  - You want the data to remain secure while stored in the OracleAS Portal instance.

- Exporting by Reference. Instead of including portlet customization data directly in the transport set, you can include it by reference in the transport set. Because the data itself is not present in the transport set, export by reference is the most secure way of transporting customizations.

5.4.8.3.1 Securing Provider Communications If the security of exporting/importing portlets is of concern to you, you should configure OracleAS Portal to secure communications with your portlet providers. The chief mechanisms for securing provider communications in OracleAS Portal are:

- Message authentication through a Hashed Message Authentication Code (HMAC) algorithm. For more information on message authentication for providers, refer to Section 6.1.7.8, "Message Authentication", in the *Oracle Application Server Portal Configuration Guide*.
- HTTPS between providers and OracleAS Portal. For more information on HTTPS for provider communications, refer to Section 6.1.7.9, "HTTPS Communication", in the *Oracle Application Server Portal Configuration Guide*.

Note: You cannot use certificates for the HTTPS communication with providers.

5.4.8.3.2 Disabling Export/Import of Customizations The JNDI variable, `oracle/portal/provider/global/transportEnabled`, controls whether to allow the exportation and importation of customizations. If you set the variable to true, customizations are transported as part of export/import. If you set it to false, they are not transported. You can set JNDI variables for PDK-Java through Oracle Enterprise Manager 10*g*. If for some reason Oracle Enterprise Manager 10*g* is not available for the instance, you can set the values manually in `orion-web.xml`. For a full Oracle Application Server installation, this file is located in:

```
ORACLE_HOME/j2ee/OC4J_instance/application-deployments/jpdk\jpdk
```

For a standalone OC4J installation, this file is located in:

```
ORACLE_HOME/j2ee/home/application-deployments/jpdk/jpdk
```

5.4.8.3.3 Obfuscating Data for Transport (Automatic) By default, customization data is encoded (Base64). This encoding ensures that data is obfuscated during transport. You do not need to take any actions to leverage Base64 encoding as it is provided by default. However, if you want greater security, you can encrypt the data. Refer to Section 5.4.8.3.4, "Encrypting Customization Data for Transport".

5.4.8.3.4 Encrypting Customization Data for Transport By implementing the `oracle.portal.provider.v2.security.CipherManager` class for your provider, you can encrypt the customization data prior to exporting it. Upon import, the cipher manager is invoked again to decrypt the data. Refer to Section 5.4.8.3.6, "Encrypting Customization Data Example".

Note: If you choose to encrypt your Web providers for export through the cipher manager, you must also devise your own key management strategy for the encryption algorithm.

5.4.8.3.5 Exporting by Reference As mentioned previously, the default behavior for exportation of portlets is to include the actual customization data in the transport set.

For a more secure transport, you can code your portlet such that the customizations are exported through pointer rather than by including the actual preference data. When the transport set is imported, the target OracleAS Portal instance sends the pointer back to the Web provider, which then has the opportunity to reassociate the actual data with the new portlet instance. Refer to Section 5.4.8.3.7, "Exporting by Reference Example".

Note: When exporting across security zones, exporting by reference may not work effectively. In general, you should only employ export by reference when transporting within the same general security environment.

5.4.8.3.6 Encrypting Customization Data Example To encrypt customization data in your Web provider, you need to create your own cipher manager and associate it with your portlet provider. This example provides a simple, insecure cipher manager for illustrative purposes only. To implement a secure implementation of the cipher manager for your production system, you would need to significantly extend this sample. Some of the issues you would need to consider for a production implementation are as follows:

- Do not hold the key object in memory. Read it from a persistent store as necessary.
- Use the provider's `PreferenceStore` API supported by a `DBPreferenceStore` to work in the clustered case.
- On import, if the cipher manager instance obtained from `provider.xml` matches the class name returned in the SOAP message, that `CipherManager` instance is used to perform the decryption. Hence, the instance maintained in the portlet/provider definition may be configured using any applicable means (for example, tags in `provider.xml` or JNDI variable) and that configuration is reused on import.

To encrypt customization data in your Web provider, do the following:

Note: The sample provided below is for illustrative purposes only. You would need to significantly enhance it for use in a production environment.

---

1. Create a cipher manager class, `InsecureCipherManager`. This class would be used for encryption and decryption of customization data exported from or imported to a Web provider. A base64 encoded, hard coded secret key is used with the DES algorithm supplied by the default `javax.crypto` provider of the underlying Java Runtime Environment. As a result, this particular sample is insecure because the encoded key can be recovered by a malicious party simply by decompiling the byte code.

Note: This sample makes use of the javax.crypto package, which is optional in Java 1.3 and must be installed manually. In Java 1.4, though, this package is present by default.

```
package oracle.portal.sample.v2.devguide.tx;
import java.io.IOException;
import java.security.GeneralSecurityException;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.DESKeySpec;
import oracle.portal.provider.v2.ProviderException;
import oracle.portal.provider.v2.security.CipherManager;
import sun.misc.BASE64Decoder;
public final class InsecureCipherManager implements CipherManager
{
    /**
     * Base64 encoded external form of a javax.crypto.SecretKey which was
     * generated for the DES algorithm. This is completely insecure! Anyone
     * can decompile the bytecode and recostitue the key. A more secure
     * implementation would implement a key management policy in a
     * java.security.KeyStore.
     */
    private static final String sEncodedKey = "UTJds807Arw=";
    /**
     * Generated from the (insecure) encoded form in sEncodedKey.
     */
    private SecretKey mKey;
    /**
     * Transforms the input data to a more secure form, in a single operation,
     * using the DES cryptographic algorithm along with a statically defined
     * secret key.
     * @param toEncode the input data.
     * @return an encoded form of the input data.
     * @throws ProviderException if an error occurs during transform.
     */
    public final byte[] encode(byte[] toEncode) throws ProviderException
    {
        try
        {
            Cipher c = Cipher.getInstance("DES");
            c.init(Cipher.ENCRYPT_MODE, getSecretKey());
            return c.doFinal(toEncode);
        }
        catch (GeneralSecurityException gse)
        {
            throw new ProviderException(gse);
        }
        catch (IOException ioe)
        {
            throw new ProviderException(ioe);
        }
    }
    /**
     * Transforms the input data to its original form, in a single operation,
     * using the DES cryptographic algorithm along with a statically defined
     * secret key.
     * @param toDecode the input data.
     * @return a decoded form of the input data.
     * @throws ProviderException if an error occurs during transform.
     */
    public final byte[] decode(byte[] toDecode) throws ProviderException
    {
        try
        {
            Cipher c = Cipher.getInstance("DES");
            c.init(Cipher.DECRYPT_MODE, getSecretKey());
            return c.doFinal(toDecode);
        }
        catch (GeneralSecurityException gse)
        {
            throw new ProviderException(gse);
        }
        catch (IOException ioe)
        {
            throw new ProviderException(ioe);
        }
    }
    /**
     * Returns a <code>javax.crypto.SecretKey</code> deserialized from the
     * obuscated form in sEncodedKey. Note, this is highly insecure!!
     */
    private SecretKey getSecretKey()
        throws GeneralSecurityException, IOException
    {
        if (mKey == null)
        {
            DESKeySpec ks = new DESKeySpec((new BASE64Decoder()).decodeBuffer(
                sEncodedKey));
            SecretKeyFactory skf = SecretKeyFactory.getInstance("DES");
            mKey = skf.generateSecret(ks);
        }
        return mKey;
    }
}
```

2. Modify your `provider.xml` to reference the cipher manager:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<?providerDefinition version="3.1"?>
<provider class="oracle.portal.provider.v2.DefaultProviderDefinition">
<providerInstanceClass>net.mzyg.tx.TxProviderInstance</providerInstanceClass>
   <session>false</session>
   <passAllUrlParams>false</passAllUrlParams>
   <preferenceStore class="oracle.portal.provider.v2.
      preference.DBPreferenceStore">
      <name>prefStore1</name>
      <connection>java:comp/env/jdbc/PortletPrefs</connection>
   </preferenceStore>
<cipherManager class="oracle.portal.sample.v2.devguide.tx.
   InsecureCipherManager"/>
```

5.4.8.3.7 Exporting by Reference Example To export by reference rather than exporting the actual customization, do the following:

1. Override the `DefaultPortletInstance` with the following `ExportByRefDefaultPortletInstance`:

```
package oracle.portal.sample.v2.devguide.tx;
import oracle.portal.provider.v2.DefaultPortletInstance;
import oracle.portal.provider.v2.preference.PreferenceStore;
import oracle.portal.provider.v2.transport.PrefStoreTransporter;
public class ExportByRefDefaultPortletInstance extends DefaultPortletInstance
{
  /**
   * Returns a {@link oracle.portal.provider.v2.transport.PrefStoreTransporter}
   * capable of carrying out transport operations such as export/import on
   * data applicable to {@link oracle.portal.provider.v2.PortletInstance}
   * persisted in {@link oracle.portal.provider.v2.preference.PreferenceStore}.
```

```
      * This implementation returns an {@link ExportByRefPrefStoreTransporter}.
      * @param ps the {@link oracle.portal.provider.v2.preference.PreferenceStore}
      * containing the data to be transported.
      * @return a {@link oracle.portal.provider.v2.transport.PrefStoreTransporter}
      */
    protected PrefStoreTransporter getPrefStoreTransporter(PreferenceStore ps)
    {
        return new ExportByRefPrefStoreTransporter(ps);
    }
}
```

2. Create the `ExportByRefPrefStoreTransporter` class referenced in `ExportByRefDefaultPortletInstance`. This class implements an alternative preference store transporter that does not send preference data during the export operation. Instead, it writes the context path of the source preference to the stream. During the export, it reads the context path and uses that path to look up the preference data and copy it to the new instance. This method of exporting by reference assumes that the source and target providers have access to the same preference store. In fact, the best case for this example would be the situation where the source and target providers are the same.

```
package oracle.portal.sample.v2.devguide.tx;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import oracle.portal.provider.v2.transport.PrefStoreTransporter;
import oracle.portal.provider.v2.transport.TransportLogger;
import oracle.portal.provider.v2.preference.Preference;
import oracle.portal.provider.v2.preference.PreferenceStore;
import oracle.portal.provider.v2.preference.PreferenceStoreException;
public class ExportByRefPrefStoreTransporter extends PrefStoreTransporter
{
    public ExportByRefPrefStoreTransporter(PreferenceStore prefStore)
    {
        super(prefStore);
    }
   /**
     * Exports the context path of the supplied {@link
     * oracle.portal.provider.v2.preference.Preference} from the {@link
     * oracle.portal.provider.v2.preference.PreferenceStore}.
     * @param pref the source {@link
     * oracle.portal.provider.v2.preference.Preference}
     * @param out the <code>java.io.OutputStream</out> to which data will be
     * written.
     * @param logger
     */
   protected void exportPreference(Preference pref, OutputStream out,
        TransportLogger logger) throws PreferenceStoreException, IOException
    {
        // Get the context path of the preference we are exporting.
        String contextPath = pref.getContextPath();
        DataOutputStream dos = new DataOutputStream(out);
        // Write the context path in the export data. The import process
        // will use this context path to lookup this preference in the
        // preference store and copy it to the new context
        dos.writeUTF(contextPath);
    }
    /**
```

```
         * Reads a context path from the stream and copies preference data
         * from that location into the {@link
         * oracle.portal.provider.v2.preference.PreferenceStore}.
         * @param pref the target {@link
         * oracle.portal.provider.v2.preference.Preference}
         * @param in the <code>java.io.InputStream</code> from which to read data.
         * @param logger
         */
        protected void importPreference(Preference pref, InputStream in,
            TransportLogger logger) throws PreferenceStoreException, IOException
        {
            // Read the context path to copy the value for this
            // preference from.
            DataInputStream dis = new DataInputStream(in);
            String contextPath = dis.readUTF();
            // Create preference object to copy from (identical to the
            // target preference but with a different context path)
            Preference sourcePref = new Preference(contextPath,
               pref.getName(), pref.getType(), (String)null);
            // Copy across the preference
            getPrefStore().copy(sourcePref, pref, true);
        }
    }
```

3. Update `provider.xml` to include the following element for your portlet:

```
<portlet class="oracle.portal.provider.v2.DefaultPortletDefinition">
...
<portletInstanceClass>oracle.portal.sample.v2.devguide.tx.
     ExportByRefDefaultPortletInstance</portletInstanceClass>
</portlet>
```

5.4.9 Enhancing Portlet Performance with Caching

In the previous sections of this chapter, you learned how to write fully-functional Java portlets using the PDK Framework. Once you complete the basic functionality of your portlet, you may want to turn your attention to portlet performance.

Caching is a common technique for enhancing the performance of Web sites that include a great deal of dynamic content. The overhead involved in retrieving data and generating the output for dynamic content can be significantly reduced by proxying requests through a local agent backed by a large, low-latency data store known as a cache. The cache agent responds to a request in one of two ways:

- If a valid version of the requested content exists in the cache, the agent simply returns the existing cached copy, thus skipping the costly process of content retrieval and generation. This condition is called a cache hit.
- If a valid version of the requested content does not exist in the cache, the agent forwards the request to its destination and awaits the return of the content. The agent returns the content to the requestor and stores a local copy in its cache for reuse if a subsequent request for the same content arises. This condition is called a cache miss.

Web providers generate dynamic content (that is, portlets) and they often reside remotely from the OracleAS Portal instance on which they are deployed. As such, caching might improve their performance. The architecture of OracleAS Portal lends itself well to caching, since all rendering requests originate from a single page assembling agent, known as the Parallel Page Engine (PPE), which resides on the middle tier. You can make the PPE cache the portlets rendered by your Web provider and reuse the cached copies to handle subsequent requests, minimizing the overhead your Web provider imposes on page assembly.

The Web provider can use any one of three different caching methods, depending upon which one is best suited to the application. The methods differ chiefly in how they determine whether content is still valid:

1. **Expiry-based Caching**: When a provider receives a render request, it stamps its response with an expiry time. The rendered response remains in the cache and fills all subsequent requests for the same content until its expiry time passes. This caching scheme is perhaps the simplest and most performant because the test for cache validity requires very little overhead and does not involve any network round trips. Expiry-based caching suits applications where the content has a well-defined life span. For content with a less certain life span, however, expiry-based caching is less effective. Refer to Section 5.4.9.2, "Activating Caching" and Section 5.4.9.3, "Adding Expiry-Based Caching" for more information.

2. **Invalidation-based Caching:** Invalidation-based caching works essentially the same way as expiry-based caching, except that the contents of the cache can expire or become invalid at any time. Invalidation of cache content is usually triggered by an event.

   For example, if you have a calendar portlet that shows your appointments for the day, the content for the portlet could be generated once, the first time you show the calendar for that day. The content remains cached until something happens to change your schedule for that day, such as the addition of an appointment, the deletion of an existing appointment, or a change of time for an appointment. Each of these change events can trigger an action in the calendar application. When such an event takes place, your calendar application can generate an invalidation request for any cached portlet content affected by the change. The next time you view a page containing your calendar portlet, the cache will not contain an entry. Your Web provider will be contacted to regenerate the new content with the modified schedule.

   This method is a very efficient way to cache content because the originator of the content (that is, your Web provider) is contacted only when new content needs to be generated, but you are not bound to a fixed regeneration schedule. Refer to Section 5.4.9.2, "Activating Caching" and Section 5.4.9.4, "Adding Invalidation Based Caching" for more information.

3. **Validation-based Caching**: When a provider receives a render request, it stamps its response with a version identifier (or E Tag). The response goes into the cache, but, before the PPE can reuse the cached response, it must determine whether the cached version is still valid. It sends the provider a render request that includes the version identifier of the cached content. The provider determines whether the version identifier remains valid. If the version identifier is still valid, the provider immediately sends a lightweight response to the PPE without any content, which indicates the cached version can be used. Otherwise, the provider generates new content with a new version identifier, which replaces the previously cached version. In this form of caching, the PPE must always confirm with the provider whether the content is up to date. The validity of the cached copy is determined by some logic in the provider. The advantage of this approach is that the provider controls the use of the cached content rather than relying on a fixed period of time. Refer to Section 5.4.9.2, "Activating Caching" and Section 5.4.9.5, "Adding Validation-Based Caching" for more information.

5.4.9.1 Assumptions

1. You have followed through and understood Section 5.4.2, "Building PDK-Java Portlets".
2. You built a portlet using the wizard and successfully added it to a page.

5.4.9.2 Activating Caching

To use the caching features of OracleAS Portal in your Web providers, you must first activate the middle tier cache. This cache is known as the PL/SQL Cache because it is the same cache used by mod_plsql, the Oracle HTTP Server plug-in that calls database procedures, and hence database providers, over HTTP.

Usually, your OracleAS Portal administrator takes care of the configuration details for caching.

For invalidation-based caching, you need to configure OracleAS Web Cache in front of the Web provider. Bear in mind that remote Web providers often do not have OracleAS Web Cache installed. For more information about OracleAS Web Cache, refer to the *Oracle Application Server Web Cache Administrator's Guide*.

Once you have installed and configured OracleAS Web Cache, ensure the following in the OracleAS Web Cache Manager:

- Points to the host name and port of the Web provider.
- Caching rules do not cause the caching of provider content. Content should be cached according to the surrogate control headers generated by the provider in its response.

5.4.9.3 Adding Expiry-Based Caching

Expiry-based caching is one of the simplest caching schemes to implement, and can be activated declaratively in your XML provider definition. You can set an expiry time for the output of any `ManagedRenderer` you utilize by setting its `pageExpires` property to the number of minutes you want the output to be cached for. For example, suppose we want portlet output to be cached for one minute.

1. After you have used the Portlet Wizard to build a portlet as described in Section 5.4.2, "Building PDK-Java Portlets", edit the `provider.xml` file and set the `pageExpires` property tag of showPage to 1. This will set an expires entry of 1 minute for the portlet.

   By default the wizard generates a standard and compressed tag for `showPage`. You need to expand the tag to include a subtag of `pageExpires`:

   ```
   <showPage class="oracle.portal.provider.v2.render.http.ResourceRenderer">
      <resourcePath>/htdocs/mycacheportlet/MyCachePortletShowPage.jsp
         </resourcePath>
      <PageExpires>1</PageExpires>
   </showPage>
   ```

2. Test that the portlet is cached for 1 minute by adding some JSP code to your show page. You can simply add the current time to your JSP.

   ```
   <%@page contentType="text/html; charset=windows-1252"
      import="oracle.portal.provider.v2.render.PortletRenderRequest"
      import="oracle.portal.provider.v2.http.HttpCommonConstants"
      import="java.Util.Date"
      import="java.text.DateFormat"
   %>
   ```

```
<%
 PortletRenderRequest pReq = (PortletRenderRequest)
   request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
 DateFormat df = DateFormat.getDateTimeInstance(DateFormat.LONG,
   DateFormat.LONG,pReq.getLocale());
 String time = df.format(new Date());
%>

<P>Hello <%=pReq.getUser().getName() %>.</P>
<P>This is the <b><i>Edit</i></b> render mode!</P>
<P>This information is correct as of <%=time%>.</P>
```

When viewing the portlet, you see that the time (including seconds) is constant for 1 minute. After the time has expired, the portlet displays the most current time and a new cache is set.

5.4.9.4 Adding Invalidation Based Caching

When using OracleAS Web Cache, requests for content are sent through HTTP and content is either returned from the cache or the HTTP request is forwarded to the originator of the content. When content is returned to OracleAS Web Cache, it is added to the cache before being returned to the requestor. Cache content is invalidated by sending invalidation requests directly to OracleAS Web Cache. PDK-Java uses the Java API for Web Cache (JAWC) to generate invalidation requests. This section describes how to configure OracleAS Web Cache and use the invalidation-based caching sample that comes with PDK-Java.

Not all requests sent to OracleAS Web Cache are cached. In order for the content to be cached, the content must include directives that tell OracleAS Web Cache to cache the content. Usually OracleAS Web Cache uses the URL associated with the request as the cache key, but you can specify additional keys by setting special HTTP headers, known as surrogate control headers, on the response.

To configure a Java portlet to use invalidation-based caching, you do the following:

- Configure OracleAS Web Cache. Refer to *Oracle Application Server Web Cache Administrator's Guide* for more information.
- Switch invalidation-based caching on at the provider servlet level.
- Define the Invalidation Port.
- Activate invalidation-based caching in the `provider.xml` file.
- To have the portlet invalidate the cache, update the JSP code in your portlet to trigger an invalidation request.

5.4.9.4.1 Configuring the Provider Servlet To enable invalidation-based caching, you must switch it on at the provider servlet level. The flag is set in an initialization parameter inside the PDK-Java Web application deployment descriptor, web.xml. For example:

```
<servlet-class>oracle.webdb.provider.v2.adapter.SOAPServlet</servlet-class>
    <init-param>
      <param-name>invalidation_caching</param-name>
      <param-value>true</param-value>
    </init-param>
</servlet>
```

If the flag is not defined here, then invalidation-based caching is switched off. The status of this flag can easily be checked by displaying the provider's test page. For example:

```
http://<provider_hostname>:<port>/jpdk/providers/webcache/MyWebPRovider
```

5.4.9.4.2 Defining the OracleAS Web Cache Invalidation Port If you are using an Oracle Application Server installation type where PDK-Java was automatically pre-installed (for example, an OracleAS Portal and Wireless type installation), you should find that OracleAS Web Cache invalidation settings have already been preconfigured in MID_TIER_ORACLE_HOME/portal/conf/cache.xml. In this case, you can safely ignore the instructions in this section and proceed to Section 5.4.9.4.3, "Configuring the XML Provider Definition". Otherwise, you will need to configure the invalidation portlet for OracleAS Web Cache.

First, you need the username and password for the invalidation port(s) of the OracleAS Web Cache instance(s) associated with your application server. After you obtain those, use the provided `oracle.webdb.provider.v2.cache.Obfuscate` Java utility to convert the username and password into an obfuscated string of the format required by the JAWC API. In a default Oracle Application Server installation, the username for the invalidation port is usually invalidator and the password is usually the same as the application server administrator's password. For example, suppose you installed Oracle Application Server in `D:\as904`, with an invalidation port username of invalidator and a password of welcome. You would run the following command:

Note: The command that follows assumes that `pdkjava.jar` is present in `ORACLE_HOME/portal/jlib` and `jawc.jar` is present in `ORACLE_HOME/webcache/jlib`, as required by the PDK-Java installation guide.

If you are using a standalone Oracle Application Server Containers for J2EE installation, you need to substitute in the path to the java executable an external Java 2 SDK installation.

```
D:\as904\jdk\bin\java -classpath D:\as904\portal\jlib\pdkjava.jar
  oracle.webdb.provider.v2.cache.Obfuscate invalidator:welcome
```

If successful, the command should print a hexadecimal string like the following:

```
0510198d5df8efd5779406342be2528aa0cccb179ea6b77baf49f019f5075a3a11
```

Now, use this information to create a JAWC configuration file, cache.xml, which specifies one or more OracleAS Web Cache instances and their invalidation ports. For example:

```
<?xml version="1.0">
<webcache>
<invalidation
 host=cache.mycompany.com
 port=4001
authorization="0510198d5df8efd5779406342be2528aa0cccb179ea6b77baf49f019f5075a3a11"/>
</webcache>
```

You may have more than one OracleAS Web Cache defined, in which case the same invalidation request goes to each OracleAS Web Cache in the configuration file, for example:

Note: The tag names `connectionPool`, `invalidationConnection1`, and `invalidationConnection2` are user defined. A template file is supplied in the directory:

```
MID_TIER_ORACLE_HOME/j2ee/OC4Jinstance/applications/jpdk/jpdk/WEB-INF/
  providers/webcache
```

```
<?xml version="1.0">
<webcache>
<connectionPool>
<invalidationConnection1
  host=cache.mycompany.com
  port=4001
authorization="0510198d5df8efd5779406342be2528aa0cccb179ea6b77baf49f019f5075a3a11"/>
<invalidationConnection2
  host=cache.mycompany.com
  port=4002
authorization="0510198d5df8efd5779406342be2528aa0cccb179ea6b77baf49f019f5075a3a11"/>
</connectionPool>
</webcache>
```

JAWC finds this configuration file using the system property `oracle.http.configfile`. To set this system property for an Oracle Application Server Containers for J2EE instance within an Oracle Application Server installation, simply add an appropriate line to the oc4j.properties file for the particular instance in which PDK-Java is installed (for example, `MID_TIER_ORACLE_HOME/j2ee/OC4Jinstance/config/oc4j.properties`) and then restart that instance. For example:

```
oracle.http.configfile=fully_qualified_filename
```

If you are running Oracle Application Server Containers for J2EE standalone, you can specify the option in the command line you use to start it.

```
java -Doracle.http.configfile=<fully_qualified_filename> -jar oc4j.jar
```

Note: Since the location of the cache configuration file is defined as a system property, only one cache may be defined per server instance. It is not possible to have two different Web Provider instances behind separate OracleAS Web Cache configurations.

5.4.9.4.3 Configuring the XML Provider Definition Using a combination of tags in `provider.xml`, you can activate automatic invalidation-based caching for an entire portlet or some of its pages. To turn on automatic invalidation-based caching, you need to declare it as follows either at the level of individual pages or the renderer, to indicate that invalidation-based caching should be activated for all pages:

```
<useInvalidationCaching>true</useInvalidationCaching>
```

If your portlet supports customization (through the Edit or Edit Defaults modes), you may also want to declare `<autoInvalidate>true</autoInvalidate>` for your renderer. This declaration causes invalidation of all the cached content for the portlet when you click OK or Apply in Edit mode, thus causing new markup to be generated based on the customized settings.

The maximum time for holding the page in the cache is the minimum of the following:

- Maximum expiry time from OracleAS Portal defined in the Cache tab of the Global Settings page.
- Web Provider default (24 hours) if no maximum expiry time is specified by OracleAS Portal.
- The time in minutes of the `<pageExpires>` tag, if present.

The following excerpt from `provider.xml` specifies that this portlet shall be cached for up to 5 minutes and shall be automatically invalidated upon customization:

```
<renderer class="oracle.portal.provider.v2.render.RenderManager">
 <contentType>text/html</contentType>
 <renderContainer>true</renderContainer>
 <autoRedirect>true</autoRedirect>
 <autoInvalidate>true</autoInvalidate>
 <showPage class="oracle.portal.provider.v2.render.http.ResourceRenderer">
  <resourcePath>/htdocs/invalidation/invalidation1.jsp</resourcePath>
  <useInvalidationCaching>true</useInvalidationCaching>
  <pageExpires>5</pageExpires>
 </showPage>
 <editPage class="oracle.portal.sample.v2.devguide.invalidation.
   InvalidationEditRenderer"/>
</renderer>
```

Note: The `pageExpires` tag is also used for normal expiry based caching. These two forms of caching are mutually exclusive. Invalidation-based caching takes place in an OracleAS Web Cache instance located in the same place as the Web provider. Pages stored using expiry based caching are cached in the middle tier of OracleAS Portal.

5.4.9.4.4 Manually Invalidating the Cache

You may want the cached version of the portlet invalidated when a request is processed or information somewhere has been updated. In these cases, you may want to manually invalidate the cache.

The invalidation-based caching portlet sample included with PDK-Java contains a single portlet that displays the time the content was cached and a link to trigger an invalidation request. The first time a page request is made to the Web provider through the cache, the response is cached. Subsequent requests for the portlet content are fulfilled by returning content from OracleAS Web Cache. When you click the link at the bottom of the portlet an invalidation request is generated by the provider that removes the portlet from the cache. The next request for the portlet is forwarded to the provider and the provider generates a new portlet with the current time.

To perform invalidation calls to OracleAS Web Cache, first you need to have a handle to a `ServletInvalidationContext` object. You can get this handle by calling the `static getServletInvalidationContext()` method of the `ServletInvalidationContextFactory` class.

Once you have the handle, you need to specify the cache key. In the cache key, you need to specify whether you want to invalidate a particular portlet instance, all the instances of a portlet, or all the portlets managed by a provider. To perform this task, you use the methods of the `ServletInvalidationContext` class or the methods of its super class, `InvalidationContext`. This is where you can specify whether the portlet should be cached for this particular user (USER level). If there is no user specified in the cache key, then the cached content is returned to all users, which means you are using SYSTEM level caching.

In the sample code below, we are invalidating a portlet instance and calling the `setFullProviderPath()` and `setPortletReference()` methods. When the caching key is set, you call the `invalidate()` method on the `InvalidationContext` object that sends the invalidation message to OracleAS Web Cache.

```
ServletInvalidationContext inv =
  ServletInvalidationContextFactory.getServletInvalidationContext();
inv.setFullProviderPath
  ((ServletPortletRenderRequest)pReq);
inv.setPortletReference
  (pReq.getPortletReference());
int num = inv.invalidate();
```

Review the Web cache sample provider for more information.

5.4.9.5 Adding Validation-Based Caching

Adding validation-based caching requires slightly more effort, but gives you explicit control over exactly which requests to your provider are cache hits. As an example, you may want to update the cache only when data within the portlet has changed. To implement this algorithm, you need to override the `prepareResponse` method. The signature of the `BaseManagedRenderer.prepareResponse` method is:

```
public boolean prepareResponse(PortletRenderRequest pr)
                        throws PortletException,
                               PortletNotFoundException
```

In your version of `prepareResponse()`, you need to do the following:

- Retrieve the cached version identifier set by the PPE in the render request by calling the `HttpPortletRendererUtil.getCachedVersion()` method:

  ```
  public static java.lang.String getCachedVersion
      (PortletRenderRequest request)
  ```

- If the portlet finds the previously cached version valid, the appropriate header has to be set by calling the `HttpPortletRendererUtil.useCachedVersion()` method. It also instructs the `RenderManager` that it won't be necessary to call `renderBody()` to render the portlet body.

  ```
  public static void useCachedVersion(PortletRenderRequest request)
  ```

  Otherwise, use `HttpPortletRendererUtil.setCachedVersion()` to generate a new version of the portlet, which will be cached. It also indicates to the PPE that the `renderBody()` method has to be called to regenerate the portlet content.

  ```
  public static void setCachedVersion(PortletRenderRequest request,
                                      java.lang.String version,
                                      int level)
                               throws java.lang.IllegalArgumentException
  ```

For validation-based caching, you need not update `provider.xml`. You can view the portlet by refreshing the page or adding the portlet to a page and updating the content. If the content has changed, the portlet shows the new content. If the content has not changed, a cached version of the portlet is displayed.

5.4.10 Writing Multi-Lingual Portlets

This section shows you how to build a Java portlet that can be rendered in different languages. The language used in your portlet will depend upon on the language setting that has been chosen in the portal that is displaying it.

Once you have completed this section you will be able to write portlets that support as many or as few languages as you wish. You will also be able to convert your existing portlets to support multiple languages. Once a portlet is written to support multiple languages, it is easy to plug in new languages. The basic model for multi-lingual Java portlets is similar to the standard Java Internationalization model. If you already know about Java Internationalization, you should find this process very familiar.

5.4.10.1 Assumptions

1. You have followed through and understood Section 5.4.2, "Building PDK-Java Portlets".
2. You built a portlet using the wizard and successfully added it to a page.

5.4.10.2 Internationalizing Your Portlet

- Providing Translations for Portlet Content
- Providing Translation for Portlet Attributes

5.4.10.2.1 Providing Translations for Portlet Content In Building PDK-Java Portlets, you created a portlet using the Java Portlet Wizard. The basic message created by the wizard is only available in one language and the text to be displayed is hard-coded in to the portlet's renderer class. To make your portlets available in multiple languages, you have to store such language dependent elements in their own *resource bundles*.

Creating Resource Bundles

For each language you want your portlet to be available in, you will need a resource bundle. You will also need to create a 'default' resource bundle that will be used when there is no resource bundle corresponding to the language setting chosen in the portal.

- **Create a Default Resource Bundle**

  1. In Oracle JDeveloper, create a Java class called MyProviderBundle that extends ListResourceBundle from the java.util.package. The class should contain a multi-dimensional array of objects that holds key-value pairs representing each of the language dependent elements from your JSP show page. This implementation is demonstrated in the following code:

```
package mypackage2;
import java.util.ListResourceBundle;
public class MyProviderBundle extends ListResourceBundle
{
public static String HELLO_MSG = "MyPortletHelloMessage";
public static String INFO_MSG = "MyPortletInfoMessage";
public Object[][] getContents()
{
return contents;
}
static private final Object[][] contents =
{
{HELLO_MSG, "Hello"},
{INFO_MSG, "This is the show page of the portlet and it is being generated
in the default language!"}
};
}
```

2. Save `MyProviderBundle`.

- **Creating Resource Bundles for Other Supported Languages**

  Now you must create a resource bundle class for each language you want your portlet to support. Each of these classes must be named the same as your default resource bundle class, but with a language code appended to the end. For example, if you want to support the French language, create a Java class named `MyProviderBundle_fr`. The language code `fr` is the same as the code that will be used by the *locale* object in the portal if the language setting is set to French (for more information on Locales, see the JavaDoc for `java.util.Locale`). When you change the language setting in OracleAS Portal, you change the value of the current locale object and therefore the locale object's language code. These language codes adhere to the ISO:639 codes for representation for names of languages.

  1. To create a French resource bundle, create a Java class named `MyProviderBundle_fr`, as described above.
  2. Using your default resource bundle as a template, replace the English language strings with their French equivalents. An example is given below:

```
package mypackage2;

import java.util.ListResourceBundle;
public class MyProviderBundle_fr extends
ListResourceBundle
{
public Object[][] getContents()
{
return contents;
}
static private final Object[][] contents =
{
{MyProviderBundle.HELLO_MSG, "Bonjour"},
{MyProviderBundle.INFO_MSG, "Cette page est le 'show mode' de la portlet
    et est generee dans la langue par defaut."}
};
}
```

  3. Save `MyProviderBundle_fr`.
  4. Repeat steps 1 through 3 for every language that you wish to create a resource bundle for, updating the class name with the appropriate language code and the message strings with their equivalent in the appropriate language.

Updating Your Renderer

To make use of the resource bundles you just created, you need to edit the JSP show page and replace the hard-coded messages with references that will pickup the messages at run time from the resource bundle that corresponds most closely with the locale object of the portal.

1. Open the JSP that represents your show page and change the following:

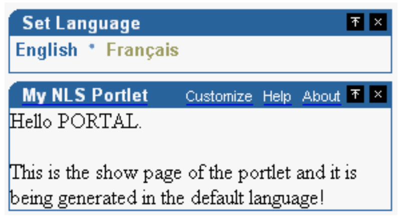
```
<%@page contentType="text/html; charset=windows-1252"
   import="oracle.portal.provider.v2.render.PortletRenderRequest"
   import="oracle.portal.provider.v2.http.HttpCommonConstants"
   import="java.util.ResourceBundle"
%>
```

```
<%
 PortletRenderRequest pReq = (PortletRenderRequest)
   request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);

<-- Get a resource bundle object for the current language. -->
ResourceBundle b =
ResourceBundle.getBundle("mypackage2.MyProviderBundle",pReq.getLocale());
%>

<--  Pull the message from the appropriate resource bundle. -->
<P> <%= b.getString(mypackage2.MyProviderBundle.HELLO_MSG) %>
   <%= pReq.getUser().getName() %>.</P>
<P> <%= b.getString(mypackage2.MyProviderBundle.INFO_MSG) %></P>
```

2. Save your JSP page.

Now you can refresh your portlet and view the changes.

Figure 5–46 Portlet in English

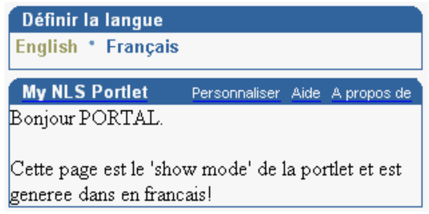


Hello PORTAL.

This is the show page of the portlet and it is being generated in the default language!

To view the French greeting, you set the language in the Set Language portlet to French instead of English.

Figure 5–47 Portlet in French



Bonjour PORTAL.

Cette page est le 'show mode' de la portlet et est generee dans en francais!

Notice that the text inside the portlet has changed, but the portlet title remains in the default language, English. You can also have the portlet set the appropriate portlet attributes (such as portlet name, portlet title, and portlet description) by pointing to a resource bundle from `provider.xml`.

5.4.10.2.2 Providing Translation for Portlet Attributes In your provider's definition file, `provider.xml`, a number of attributes describing your portlet are defined such as the portlets name and description, these are used in places, for example in your portlet's title bar in Show mode and so should be translated, too. There are two different ways of providing these translations, which one you choose is up to you. Both of these methods are outlined below:

- Method 1: Using Resource Bundles at the Provider Level
- Method 2: Creating Resource Bundles at Portlet Level

Method 1: Using Resource Bundles at the Provider Level

You can provide translations for your portlet attributes in your resource bundle(s), then specify that you want to use these resource bundles in `provider.xml`, specifying the keys you have used in your resource bundles. Using this method you can use the keys you want to, and as long as you use different keys for each corresponding attribute in your provider's various portlets you can have just one set of resource bundles that all of your provider's portlets can use.

- **Updating Your Resource Bundles**

1. Open your default resource bundle, `MyProviderBundle.java`.
2. Add additional strings to your resource bundle that represent your portlet attributes and then add text for those strings:

```
package mypackage2;

import java.util.ListResourceBundle;
public class MyProviderBundle extends ListResourceBundle
{
public static String HELLO_MSG = "MyPortletHelloMessage";
public static String INFO_MSG = "MyPortletInfoMessage";
public static String PORTLET_NAME = "FirstPortletName";
public static String PORTLET_TITLE = "FirstPortletTitle";
public static String PORTLET_SHORT_TITLE = "FirstPortletShortTitle";
public static String PORTLET_DESCRIPTION = "FirstPortletDescription";
public static String TIMEOUT_MESSAGE = "FirstPortletTimeoutMessage";

public Object[][] getContents()
{
return contents;
}
static private final Object[][] contents =
{
{HELLO_MSG, "Hi"},
{INFO_MSG, "This is the show page of the portlet and it is being generated
   in the default language!"},
{PORTLET_NAME, "MyGlobalPortlet"},
{PORTLET_TITLE, "My Global Portlet"},
{PORTLET_SHORT_TITLE, "MyGlobalPortlet"},
{PORTLET_DESCRIPTION, "My first ever Global portlet, using my
   MyPortletShowPage.jsp"},
{TIMEOUT_MESSAGE, "Timed out waiting for MyGlobalPortlet"}
};
}
```

3. Save `MyProviderBundle.java`.
4. Open `MyProviderBundle_fr.java`. Change it so that it contains the French strings that match the strings declared in `MyProviderBundle`.

```
package mypackage2;

import java.util.ListResourceBundle;
public class MyProviderBundle_fr extends ListResourceBundle
{
public Object[][] getContents()
{
return contents;
}
static private final Object[][] contents =
{
```

```
{MyProviderBundle.HELLO_MSG, "Bonjour"},
{MyProviderBundle.INFO_MSG, "Cette page est le 'show mode' de la portlet
   et est generee en francais!"},
{MyProviderBundle.PORTLET_NAME, "MaPremierePortlet"},
{MyProviderBundle.PORTLET_TITLE, "Ma Portlet Multi-Langue"},
{MyProviderBundle.PORTLET_SHORT_TITLE, "Ma Global Portlet"},
{MyProviderBundle.PORTLET_DESCRIPTION, "Ma premiere portlet
   multi-langue, utilisant mon renderer"},
{MyProviderBundle.TIMEOUT_MESSAGE, "Temps d'acces a la portlet
   demandee expire"}
};
}
```

5. Save `MyProviderBundle_fr.java`.

- **Updating provider.xml**

1. Open the XML provider definition file and update it to point to the resource bundle instead of using the hard-coded portlet attribute values.

```
<portlet class="oracle.portal.provider.v2.DefaultPortletDefinition">
 <id>3</id>
 <resource>MyProviderBundle</resource>
 <nameKey>FirstPortletName</nameKey>
 <titleKey>FirstPortletTitle</titleKey>
 <ShortTitleKey>FirstPortletShortTitle</ShortTitleKey>
 <descriptionKey>FirstPortletDescription</descriptionKey>
 <timeout>10</timeout>
 <timeoutMessageKey>FirstPortletTimeoutMessage</timeoutMessageKey>
 <showEditToPublic>false</showEditToPublic>
 <hasAbout>true</hasAbout>
```

Method 2: Creating Resource Bundles at Portlet Level

PDK-Java defines a set of resource bundle keys that you can use for providing translations for your portlet attributes. Making use of these keys means that you don't have to specify the resource bundle keys in your `provider.xml` file, as we did in Method 1: Using Resource Bundles at the Provider Level. However, you do have to provide a separate set of resource bundles for each portlet in your provider as the keys you use for each portlet need to be the same, but their values will differ.

- **Updating Your Resource Bundles**

1. Open your default resource bundle, `MyProviderBundle.java`.
2. Remove any changes you made from the previous section, and import `oracle.portal.provider.v2.PortletConstants`. You can then reference the following constants instead of the strings. You do not have to declare static strings when using `PortletConstants`:

```
{PortletConstants.NAME, "MyGlobalPortlet"},
{PortletConstants.TITLE, "My Global portlet"},
{PortletConstants.SHORTTITLE, "MyGlobalPortlet"},
{PortletConstants.DESCRIPTION, "My first ever Global portlet"},
{PortletConstants.TIMEOUTMSG, "Timed out waiting for MyGlobalPortlet"}
```

3. Save `MyProviderBundle.java`.
4. Open `MyProviderBundle_fr.java`. Remove the portlet attributes added in the previous section, import `oracle.portal.provider.v2.PortletConstants`, and reference the constants instead of the strings.

```
{PortletConstants.NAME, "MaPremierePortlet"},
{PortletConstants.TITLE, "Ma Portlet Multi-Langue"},
{PortletConstants.SHORTTITLE, "Ma Global Portlet"},
{PortletConstants.DESCRIPTION, "Ma premiere portlet multi-langue,
   utilisant mon renderer"},
{PortletConstants.TIMEOUTMSG, "Temps d'acces a la portlet demandee
   expire"}
```

5. Save `MyProviderBundle_fr.java`.

- **Updating provider.xml**

  1. In `provider.xml`, you need to use only one tag instead of one tag for each string as you did in Method 1: Using Resource Bundles at the Provider Level. Add the tag between the portlet id and the timeout number value.

  ```
  <resource>mypackage2.MyProviderBundle</resource>
  ```

  For more information on Java Internationalization see the **Internationalization trail** of the Java Tutorial.

5.4.10.3 Viewing the Portlet

Once you have updated your provider and deployed it to Oracle Application Server Containers for J2EE, refresh the provider and portal page containing your portlet. To see your resource bundles working, add the "Set Language" portlet to your page and try changing the language setting to French. Remember that the default resource bundle is English, and that selecting any other language that doesn't have a corresponding resource bundle will result in the portlet being displayed in English.

5.5 Building Struts Portlets with Oracle JDeveloper

This section describes the framework for building Struts portlets with Oracle JDeveloper for use in OracleAS Portal. You will learn how to build a Struts portlet from an existing application by adding the Struts Tag Library from the Oracle Application Server Portal Developer Kit (version 9.0.4.0.2 or higher) to Oracle JDeveloper, then use the Oracle PDK Java Portlet wizard to create the Java portlet itself.

- OracleAS Portal and the Apache Struts Framework
- Creating a Struts Portlet

5.5.1 OracleAS Portal and the Apache Struts Framework

This section discusses the use of the Apache Struts with OracleAS Portal. Struts is an implementation of the Model-View-Controller (MVC) design pattern.

- Model View Controller Overview
- Apache Struts Overview
- OracleAS Portal Integration with Struts
- Summary

5.5.1.1 Model View Controller Overview

Database applications undertake several distinct tasks:

- Data access

- Business logic implementation
- User interface display
- User interaction
- Application (page) Flow

The MVC (Model View Controller) architecture provides a way of compartmentalizing these tasks, based on the premise that activities, such as data presentation, should be separate from data access. This architecture enables you to easily plug a data source to the application without having to rewrite the user interface. MVC allows the logical separation of an application into three distinct layers: the Model, the View, and the Controller.

The Model

The Model is the repository for the application data and business logic. One facet of the Model's purpose is to retrieve data from and persist data to the database. It is also responsible for exposing the data in such a way that the View can access it, and for implementing a business logic layer to validate and consume the data entered through the View. At the application level, the Model acts as the validation and abstraction layer between the user interface and the business data that is displayed. The database server itself is simply a persistence layer for the Model.

The View

The View is responsible for rendering the Model data using JSPs. The View code does not include a hardcoded application or navigation logic, but may contain some logic to carry out tasks like displaying conditional data based on a user role. When an end user executes an action within the HTML page that the View renders, an event is submitted to the Controller. The Controller then determines the next step.

The Controller

The Controller is the linchpin of the MVC pattern. Every user action carried out in the View is submitted through the Controller. The Controller then performs the next action, based on the content of the request from the browser.

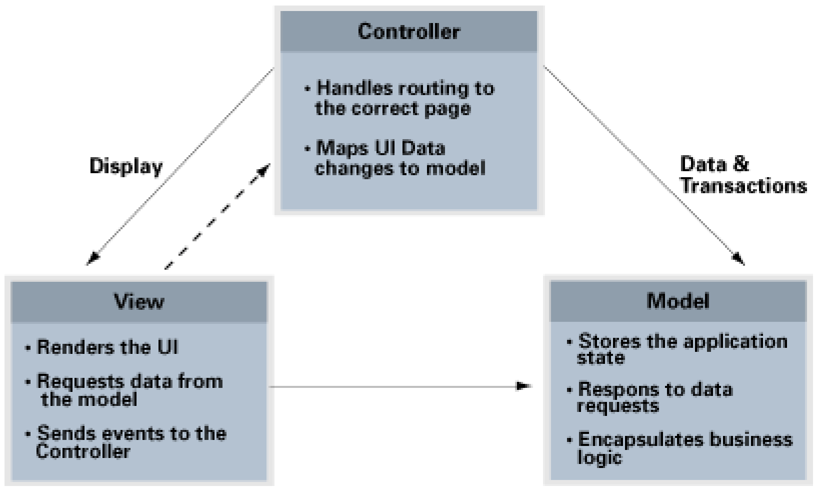
The Controller can be driven in several different ways. For example, you can use URL arguments to route the requests to the correct code,. The MVC pattern itself determines the function of the Controller, not how it should work.

Benefits

The MVC architecture provides a clear and modular view of the application and its design. By separating the different components and roles of the application logic, it allows developers to design applications as a series of simple and different components: the Model, the View, and the Controller. This pattern should help to create applications that are easier to maintain and evolve. For example, once you create one view, you can easily create another view using the same business logic. Because of the ease and reusability, the MVC pattern is the most widely used pattern in the context of Web-based application development.

The following diagram shows how the MVC pattern applies to a conventional thin-client Web application:

Figure 5–48 The MVC Pattern


Controller
• Handles routing to the correct page
• Maps UI Data changes to model
Display
Data & Transactions
View
• Renders the UI
• Requests data from the model
• Sends events to the Controller
Model
• Stores the application state
• Respons to data requests
• Encapsulates business logic

5.5.1.2 Apache Struts Overview

The Apache Struts framework (`http://struts.apache.org`) is one of the most popular frameworks for building Web applications, and provides an architecture based on the JSP Model 2 approach of the MVC design paradigm. In the Model 2 approach, the end user requests are managed by a servlet that controls the flow, and uses components such as JavaBeans, EJBs, or ADF Business Components to access and manipulate the data. It then uses JSPs to render the application content in a Web browser. This model differs from JSP Model 1, where the JSPs managed the browser request and data access.

The Struts framework provides its own HTTP Servlet as a controller component. The Struts framework is driven by an XML configuration file that contains the page flow of the application. Struts does not provide the Model, but allows the developers to integrate it to any data access mechanism, for example EJBs, TopLink, ADF Business Components, or JDBC. The Struts View by default uses JavaServer Pages (JSP), including the Java Standard Tag Library (JSTL) and JavaServer Faces (JSF). Struts provides a set of JavaBeans and JSP tags to help you use the different components of the MVC.

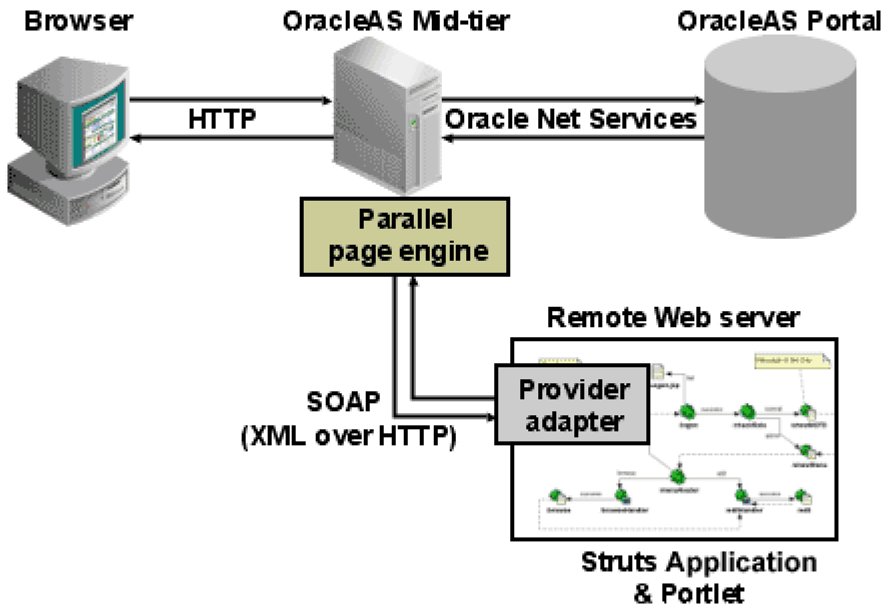
Note: For more information about JSTL and JSF, see the FAQ on the Apache Software Foundation Web site (`http://struts.apache.org/faqs/kickstart.html`).

5.5.1.3 OracleAS Portal Integration with Struts

The Oracle Application Server Portal Developer Kit contains numerous examples and documents regarding the usage of the OracleAS Portal APIs, such as personalization and caching. The integration of the application flow and business logic is not part of the portlet APIs. By using the Struts framework, however, you can leverage the MVC architecture to create and publish applications within your enterprise portal.

5.5.1.3.1 Oracle Struts Portlet To create a portlet using the Struts framework, or to generate a portlet from an existing Struts application, you must deploy all the components in the J2EE container. In the context of OracleAS Portal, the Struts application is called by the PPE, and not by the browser as compared to a standalone Struts application. When a request is made, Oracle Parallel Page Engine (PPE), calls the Struts portlet that then forwards the request to the Apache Struts Controller servlet.

Figure 5–49 Integrating Struts Applications with OracleAS Portal



The following code shows a portion of the provider definition file (`provider.xml`):

```
...
<renderContainer>true</renderContainer>
    <renderCustomize>true</renderCustomize>
    <autoRedirect>true</autoRedirect>
    <contentType>text/html</contentType>
    <showPage class="oracle.portal.provider.v2.render.http.StrutsRenderer">
        <defaultAction>showCustomer.do</defaultAction>
    </showPage>
</renderer>
...
```

The `showPage` tag defines the business logic that will be executed in the Show mode of the portlet. The `showPage` of the Struts portlet contains two important components:

1. The renderer class (`oracle.portal.provider.v2.render.http.StrutsRenderer`), which receives the portlet request from the PPE and acts as a proxy to forward the request to the Struts Action Servlet.
2. The `defaultAction` tag, which defines the Struts action that will be used by default when the portlet is called for the first time.

The Oracle Application Server Portal Developer Kit enables you to easily develop a view (Portal View) of your Struts application. This view enforces a consistent look and feel of your Struts portlet using Portal styles, and allows the end user to use the application within the portal.

To create a Struts portlet, you must use the OracleAS Portal JSP tags, which are extensions of the default Struts JSP tags. This development process is similar to that of creating a standalone Struts application. To learn how to build a Struts portlet, refer to

Section 5.5.2.1, "Creating a Struts Portlet". Also, since the portlet and struts application must also be in the same Servlet Context, you must create a single Web application that contains both elements. To learn how to easily create this Web application in Oracle JDeveloper, refer to the next section, Section 5.5.2.1, "Creating a Struts Portlet".

5.5.1.4 Summary

Apache Struts has become the de facto standard for developing MVC-based J2EE applications, because it offers a clean and simple implementation of the MVC design paradigm. This framework enables you, as the portlet developer, to separate the different components of an application, and to leverage the Struts controller to easily publish an existing Struts application to OracleAS Portal without completely changing the existing business logic.

The Oracle Application Development Framework (ADF) can use the Struts action servlet as a controller, or use its own controller (the ADF Controller or ADFc). This framework enables you to use the same approach to build Struts portlets as well as ADF portlets.

Note: For more information on the Oracle Application Server Portal Developer Kit, see Portal Center (`http://www.oracle.com/technology/products/ias/portal/index.html`).

5.5.2 Creating a Struts Portlet

OracleAS PDK (version 9.0.4.0.2) introduced new extensions to integrate Apache Struts applications. This section explains how to build a portlet from an existing Struts application. You can also follow these steps to create a portlet that uses the Model View Controller paradigm. To learn more about the Apache Struts framework, refer to Section 5.5.1, "OracleAS Portal and the Apache Struts Framework".

The PDK-Java extensions described in this article are compliant to the 1.1 production version of the Apache Struts project. Oracle JDeveloper version 9.0.3.x contains by default the Apache Struts version 1.1b2 (1.1 Beta 2). If you want to use Oracle JDeveloper 9.0.3.x to build Struts applications and portlets, you must upgrade the Oracle JDeveloper Struts libraries to the production version of Struts. Apache Struts 1.1 production is embedded with Oracle JDeveloper 10*g* (9.0.4) and higher.

This section contains the following steps:

- Creating a Struts Portlet
- Registering the Provider
- Summary

5.5.2.1 Creating a Struts Portlet

To publish a part of an existing Struts application as portlet, you must first create a new view that will serve as the Portal View of your application. This view will use existing objects (`Actions`, `ActionForm`, and so on) with a new mapping and new JavaServer Pages.

In this example, you will create a portlet that enables you to add a new entry to a Web Logger (Blog).

Figure 5–50 Submitting a Blog

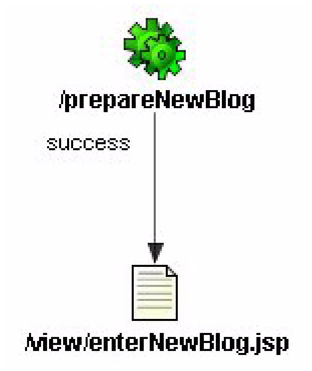
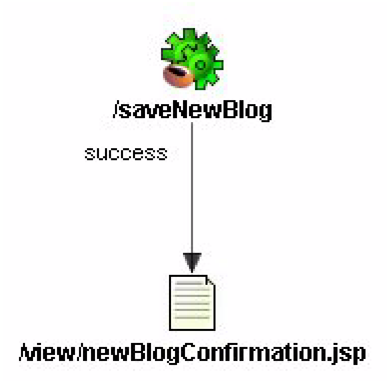


Figure 5–51 Saving a Blog Entry



`prepareNewBlog` is a simple empty action that redirects the request to the `enterNewBlog.jsp` page. This page shows a form for submitting a new blog.

The corresponding entry in the `struts-config.xml` is:

```
<action path="/prepareNewBlog" scope="request"
type="view.PrepareNewBlogAction" >
    <forward name="success" path="/view/enterNewBlog.jsp"/>
</action>
<action path="/saveNewBlog" name="blogForm" scope="request"
    type="view.SaveNewBlogAction" input"/view/enterNewBlog.jsp"  >
    <forward name="success" path="/view/newBlogConfirmation.jsp"/>
</action>
```

5.5.2.1.1 Create a new flow and view to host the Portlet actions To create a new view, first create a new set of `ActionMappings` (page flow) that will redirect the various actions and requests to Portal-specific JSPs.

```
<action path="/portal/prepareNewBlog" scope="request"
    type="view.PrepareNewBlogAction" >
    <forward name="success" path="/view/portal/enterNewBlog.jsp"/>
</action>
<action path="/portal/saveNewBlog" name="blogForm" scope="request"
type="view.SaveNewBlogAction" input="/view/enterNewBlog.jsp"  >
    <forward name="success" path="/view/portal/newBlogConfirmation.jsp"/>
</action>
```

As you can see, only the path attributes are modified. The `FormBean` Action responsible for the application business logic remains unchanged.

5.5.2.1.2 Creating the new JSPs As specified in the previous step, the actions forward the request to new JSPs, which are responsible for rendering the portlet content. Your new portlet view JSPs can share the HTML with the standalone view, but be sure that the portlet:

- Uses Portal styles that enforce a consistent look and feel with the rest of the portal page.
- Contains HTML code that is allowed in HTML table cells (that is, no `<html>`, `<body>`, and `<frame>` tags).
- Renders portal-aware links and forms. This is necessary to ensure that your Struts portlet renders its content inline, thus keeping your users within the context of the portal page by rendering the requested content within the same portlet container.

To achieve inline rendering in your Struts portlet, you must use OracleAS PDK tags:

```
<pdk-struts-html:form action="/portal/saveNewBlog.do">
...
...
</pdk-struts-html:form>
```

During the rendering of the portlet, one of the JSP tags (for example, the `pdk-struts-html:form` tag), submits the form to the Parallel Page Engine (PPE), which then sends the parameters to the Struts portlet. The Struts controller executes the logic behind these actions and returns the next JSP to the portlet within the portal page.

The PDK contains all the Struts tags, and extends all the tags that are related to URLs. The following is a list of the PDK extended tags:

- `form`: creates an HTML form and embeds the portal page context in the form to ensure inline rendering
- `link` and `rewrite`: create a link to the portal page, and are required for inline rendering
- `img`: creates an absolute link that points to the Web provider. If you want to use this tag in the context of Internet Web sites that have firewalls, you must make sure the provider is directly accessible from the Internet. If it is not possible, you can deploy the images to the OracleAS Portal middle tier and use the Apache Struts image link to generate a relative link (relative to the portal, not to the application).

Note: You can register the OracleAS PDK with Oracle JDeveloper so that you can drop the tags from the Oracle JDeveloper Components Palette. For more information, see the "Registering a Custom Tag Library in JDeveloper" section in the Oracle JDeveloper online help.

5.5.2.1.3 Creating a Portlet You can create your Struts portlet either manually or by using the Java Portlet wizard. Although the wizard does not explicitly offer Struts support, you can utilize the wizard to build your Struts portlet.

To create a portlet:

1. In Oracle JDeveloper, open the OracleAS PDK Java Portlet Wizard.

Note: For more information on opening the wizard, see Section 5.4.2.1, "Creating a Portlet and Provider".

---

2. For the implementation style of the show page, select **Java Class**.
3. For the Package name, enter `oracle.portal.provider.v2.render.http`
4. For the class name, enter `StrutsRenderer`.
5. The Java Portlet Wizard generates the skeleton of the portlet renderer class, StrutsRenderer, for you. Since the StrutsRenderer is part of the PDK, you do not need this generated file. So, when you finish the wizard, you must delete the file generated by the wizard. To do so, click the file in the System Navigator window, then select the **Erase from Disk** option in the File menu of Oracle JDeveloper.
6. Edit the `provider.xml` and change the following properties:

   At the provider level:

   - Enable session handling, because Struts applications use session management:

     ```
     <session>true</session>
     ```

   - Enable the passing of URL parameters, so that all parameters from any form submission can be passed to the Struts controller:

     ```
     <passAllUrlParams>true</passAllUrlParams>
     ```

   At the portlet level:

   - Specify the first action to raise when the portlet is called. Use the following code:

     ```
     <showPage class="oracle.portal.provider.v2.render.http.StrutsRenderer">
     <defaultAction>/portal/prepareNewBlog.do</defaultAction>
     </showPage>
     ```

5.5.2.1.4 Extending the portlet to add Portal Business Logic In your application, you should add code specific to your portal, such as the user's information, personalization, and localization. To do so, you can create a new `Action` class that is only called in the Portal context, and handles all Portal-specific business logic.

5.5.2.2 Registering the Provider

Now that your portlet is ready to be used by OracleAS Portal, you must make it accessible to OracleAS Portal by registering it. When your register your provider, make sure the session information is passed to your Struts portlet by specifying the provider login frequency as "Once per user session."

5.5.2.3 Summary

Oracle Application Server enables you to easily create Struts portlets using Oracle JDeveloper and publish existing Struts applications to OracleAS Portal. For more information on using the Oracle JDeveloper Java Portlet wizards, refer to the beginning of this chapter. For more information on using OracleAS Portal, refer to the *Oracle Application Server Portal User's Guide* and the *OracleAS Portal Online Help*.

6 Building PL/SQL Portlets

Note: In general, Oracle recommends that you build your portlets using Java rather than PL/SQL. For more information on choosing a technology for building your portlets, refer to Chapter 2, "Portlet Technologies Matrix". For more information on building your portlets using Java, refer to Chapter 5, "Building Java Portlets".

The OracleAS Portal PL/SQL APIs are implemented as a set of PL/SQL packages and objects. Database providers and portlets are deployed to a database schema as PL/SQL packages. This chapter explains how to create PL/SQL portlets based on the Oracle Application Server Portal Developer Kit-PL/SQL (PDK-PL/SQL). To make effective use of this chapter, you should already know PL/SQL and have some familiarity with the PL/SQL Web Toolkit.

This chapter contains the following sections:

- Guidelines for Creating PL/SQL Portlets
- Building PL/SQL Portlets with the PL/SQL Generator
- Building PL/SQL Portlets Manually
- Implementing Information Storage
- Using Parameters
- Accessing Context Information
- Implementing Portlet Security
- Improving Portlet Performance with Caching
- Implementing Error Handling
- Implementing Event Logging
- Writing Multi-Lingual Portlets

6.1 Guidelines for Creating PL/SQL Portlets

When you write your portlets in PL/SQL, you should follow the best practices described in this section:

- Portlet Show Modes
- Recommended Portlet Procedures and Functions

6.1.1 Portlet Show Modes

Just like a Java portlet, a PL/SQL portlet has a variety of Show modes available to it. A Show mode is an area of functionality provided by a portlet. The available Show modes are described more fully in Chapter 5, "Building Java Portlets":

- Shared Screen mode is described in Section 5.1.1, "Shared Screen Mode (View Mode for JPS)"
- Edit mode is described in Section 5.1.2, "Edit Mode (JPS and OracleAS Portal)".
- Edit Defaults mode is described in Section 5.1.3, "Edit Defaults Mode (JPS and OracleAS Portal)".
- Preview mode is describe in Section 5.1.4, "Preview Mode (JPS and OracleAS Portal)".
- Full Screen mode is described in Section 5.1.5, "Full Screen Mode (OracleAS Portal)".
- Help mode is described in Section 5.1.6, "Help Mode (JPS and OracleAS Portal)".
- About mode is described in Section 5.1.7, "About Mode (JPS and OracleAS Portal)".
- Link mode is described in Section 5.1.8, "Link Mode (OracleAS Portal)".

To check the selected Show mode, you can use the constants in the `wwpro_api_provider` package. These constants are listed with their corresponding Show mode in Table 6–1.

Table 6–1 Show Mode Constants in wwpro_api_provider

Show mode	Constant
Shared Screen	`MODE_SHOW`
Edit	`MODE_SHOW_EDIT`
Edit Defaults	`MODE_SHOW_EDIT_DEFAULTS`
Preview	`MODE_SHOW_PREVIEW`
Full Screen	`MODE_SHOW_DETAILS`
Help	`MODE_SHOW_HELP`
About	`MODE_SHOW_ABOUT`
Link	`MODE_SHOW_LINK`

6.1.2 Recommended Portlet Procedures and Functions

The primary goal of the portlet's code is to generate the HTML output that displays on a page for all of the Show modes required by OracleAS Portal. Although it is possible to implement the portlet as a set of separate PL/SQL stored program units, organizing the portlet's code into a PL/SQL package is the best way of encapsulating related portlet code and data as a single unit in the database. You also achieve better database performance and ease of portlet maintenance.

As you may recall from Section 2.4, "Deployment Type", requests from OracleAS Portal for a particular portlet go through the portlet's provider. To communicate with its portlets, the provider contains a set of required methods that make calls to the portlet code.

When implementing a portlet as a PL/SQL package, it is a good idea to organize the portlet code in parallel with the provider code. For example, when the provider needs to retrieve information about one of its portlets, it uses its `get_portlet` function. Hence, it makes sense for the portlet to contain a `get_portlet_info` function that returns the requested information when called by the provider's `get_portlet` function. Similarly, it is logical for the provider's `show_portlet` procedure to call the portlet's `show` procedure, which produces the HTML output for a requested Show mode and returns it to the provider.

The following procedures and functions are recommended for use in PL/SQL portlets to communicate efficiently with the database provider:

- `get_portlet_info` returns the portlet record to the provider.
- `show` produces HTML output for a requested Show mode and returns it to the provider.
- `register` initializes the portlet at the instance level.
- `deregister` enables cleanups at the instance level.
- `is_runnable` determines whether the portlet can be run. Security checks can be performed in this function.
- `copy` copies the customized and default values of portlet preferences from one portlet instance to a new portlet instance when OracleAS Portal makes a copy of the page.
- `describe_parameters` returns a list of public portlet parameters.

6.2 Building PL/SQL Portlets with the PL/SQL Generator

To facilitate the development of database providers and PL/SQL portlets, you can use the PL/SQL Generator, a utility that creates installable PL/SQL code for a database provider and its portlets. The PL/SQL Generator is a standalone Web application that receives the provider and portlet definitions in the form of an XML file (similar in format to the `provider.xml` file). The XML tags used for the provider and portlet definition are a subset of the XML tags used for defining Web providers with PDK-Java. The output of the PL/SQL Generator is a SQL script that can be run from SQL*Plus. The script contains SQL commands for installing the provider and portlet packages in the correct order.

You can download the PL/SQL Generator along with its installation instructions from:

```
http://www.oracle.com/technology/products/ias/portal/files/plsqlgenerator.zip
```

The general model for working with the PL/SQL Generator is as follows:

1. Create an XML file that defines the provider and portlets that you want to build, as described in Section 6.2.1, "Creating the Input XML File".
2. Run the PL/SQL Generator using the XML file as input, as described in Section 6.2.2, "Running the PL/SQL Generator".
3. Publish the generated PL/SQL portlet, which includes the following steps:
   - Install the provider generated by the PL/SQL Generator into the database, as described in Section 6.2.3.1, "Installing the Packages in the Database".
   - Register the database provider with the Oracle Application Server, as described in Section 6.2.3.2, "Registering the Database Provider".

- Add the generated portlet to a page, as described in Section 6.2.3.3, "Adding Your Portlet to a Page".

6.2.1 Creating the Input XML File

The source XML file starts and ends with the `<provider>` and `</provider>` tags, and can include one or many portlet definitions. Each portlet definition is bracketed by the `<portlet>` and `</portlet>` tags. A portlet definition includes the XML tags that specify values for the portlet record attributes and enable the links in the portlet header. For example, the `<name>` tag specifies the portlet name in the provider domain and the `<title>` tag specifies the portlet display name or title. When set to true, the `<showEdit>` tag enables the Edit mode for the portlet and the corresponding link in the portlet header. Table 6–2 lists the available XML tags for PL/SQL Generator input.

Table 6–2 XML Tags for PL/SQL Generator Input

XML Tag	Definition	Value Type
`provider`	Encloses provider definition tags.	Not applicable
`portlet`	Encloses portlet definition tags.	Not applicable
`id`	Specifies the portlet ID in the provider. This value must be unique within the provider.	string
`name`	Specifies the portlet name. The name should not contain any spaces. The generator uses the information provided in the `name` tag for the portlet package name.	string
`title`	Specifies the portlet display name.	string
`shortTitle`	Specifies the portlet short display name. This tag is useful for mobile portlets.	string
`description`	Specifies the portlet description.	string
`defaultLocale`	Specifies the language the portlet renders by default. The value is the two letter ISO language and country codes expressed as l*`anguage.country`*.	string
`timeout`	Specifies the portlet's timeout interval in seconds.	number
`timeoutMsg`	Specifies the message to display when the portlet times out.	string
`showEdit`	Indicates whether the portlet supports Edit mode, which enables the user to customize the portlet's properties.	Boolean
`showEditDefault`	Indicates whether the portlet supports the Edit Defaults mode, which enables page administrators to customize the default values of the portlet's properties.	Boolean
`showDetails`	Indicates whether the portlet can be viewed in Full Screen mode. In this mode, the entire browser window is dedicated to the portlet. Full screen mode enables the portlet to show more details than when it shares the page with other portlets.	Boolean
`showPreview`	Indicates whether the portlet supports the Preview mode.	Boolean
`hasHelp`	Indicates whether the portlet supports the Help mode.	Boolean

Table 6–2 (Cont.) XML Tags for PL/SQL Generator Input

XML Tag	Definition	Value Type
`hasAbout`	Indicates whether the portlet supports the About mode.	Boolean
`language`	Defines the portlet's default language (for example, `en`).	string

`contentType`	Indicates the default content type supported by the portlet. The tag can take one of the following values:
	`wwpro_api_provider.CONTENT_TYPE_HTML`
	`wwpro_api_provider.CONTENT_TYPE_XML`
	`wwpro_api_provider.CONTENT_TYPE_MOBILE`
		string

`apiVersion`	Specifies the version of the OracleAS Portal PL/SQL API to which the portlet conforms. For OracleAS Portal 9.0.4 and earlier releases, the tag value should be `wwpro_api_provider.API_VERSION_1`.	string
`callIsRunnable`	Indicates whether OracleAS Portal must check for the user's credentials before displaying the portlet. The default value is `true`.	Boolean
`callGetPortlet`	Indicates whether the portal can use the portlet record data stored in the Portlet Metadata Repository (PMR) instead of contacting the provider for the portlet record. If the portlet record (specified by provider `id`, portlet `id`, and `language`) returned by a provider does not change, then the provider should set the value for `call_get_portlet` to false. This tells the portal to use the PMR instead of making calls to the provider's `get_portlet` and `get_portlet_list` functions. An example of when a provider would not want the portal to use portlet metadata from the PMR is when the value of the portlet records is different for logged on users. The default value is `true`.	Boolean
`acceptContentType`	Specifies a comma delimited list of content types that the portlet can produce. For example, if a portlet can produce content of both HTML and MOBILEXML type, then the tag value is: `text/html,text/vnd.oracle.mobilexml`	string
`hasShowLinkMode`	Indicates whether the portlet implements the Link mode. If the value is `false`, the portlet uses its short or full title to display a link label that references the portlet content in a mobile device. Otherwise, a customized link can be generated in the portlet code. The default value is `false`.	Boolean
`mobileOnly`	Indicates whether the portlet is available only to mobile devices. The default value is `false`.	Boolean
`preferenceStorePath`	Specifies the base preference store path where the provider has stored the portlet customization information. This path is used when exporting portlets.	string
`createdOn`	Defines the portlet creation date. The default value is `sysdate`.	date
`createdBy`	Identifies the user who created the portlet record.	string
`lastUpdatedOn`	Defines the most recent date on which the portlet record was changed. The default value is `sysdate`.	date

Table 6–2 (Cont.) XML Tags for PL/SQL Generator Input

XML Tag	Definition	Value Type
`lastUpdatedBy`	Identifies the user who most recently changed the portlet record.	string
`passAllUrlParams`	Indicates parameter passing behavior in the portlet. If the tag value is `true`, then OracleAS Portal passes all parameters in the URL to the portlet. If the tag value is `false`, then the portlet receives only those parameters that are intended for the portlet. The default value is `true`.	Boolean

`cacheLevel`	Indicates a portlet's cache level. It can take one of the following values:
	`wwpro_api_provider.CACHE_LEVEL_SYSTEM`
	`wwpro_api_provider.CACHE_LEVEL_USER`
	`wwpro_api_provider.CACHE_LEVEL_PTL_SESSION`
		string

`rewriteUrls`	Indicates whether or not URL rewriting will be performed on the output from a portlet render request. The default value is `false`.	Boolean

Following is a sample of the input XML for the PL/SQL Generator. Mandatory information is shown in bold.

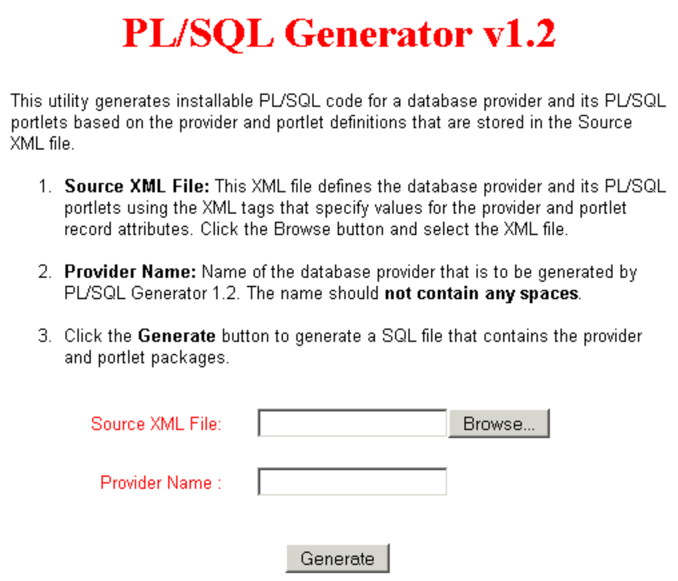
```
<!-- This is a sample provider.xml file for the PLSQL Generator 1.2 -->
<provider>
    <portlet>
        <id>1</id>
        <name>Test_Portlet</name>
        <title>Test Portlet Title</title>
        <shortTitle>Short portlet title</shortTitle>
        <description>This is a Test portlet</description>
        <timeout>30</timeout>
        <timeoutMsg>Test Portlet Timed Out</timeoutMsg>
        <showEdit>true</showEdit>
        <showEditDefault>true</showEditDefault>
        <showDetails>true</showDetails>
        <showPreview>true</showPreview>
        <hasHelp>true</hasHelp>
        <hasAbout>true</hasAbout>
        <language>en</language>
        <contentType>wwpro_api_provider.CONTENT_TYPE_HTML</contentType>
        <apiVersion>wwpro_api_provider.API_VERSION_1</apiVersion>
        <callIsRunnable>true</callIsRunnable>
        <callGetPortlet>true</callGetPortlet>
        <acceptContentType>'text/html'</acceptContentType>
        <hasShowLinkMode>false</hasShowLinkMode>
        <mobileOnly>false</mobileOnly>
        <passAllUrlParams>true</passAllUrlParams>
        <cacheLevel>wwpro_api_provider.CACHE_LEVEL_USER</cacheLevel>
        <rewriteUrls>true</rewriteUrls>
    </portlet>
</provider>
```

6.2.2 Running the PL/SQL Generator

Once you have created a valid XML input file, you can run the PL/SQL Generator to generate the provider and portlet packages in the form of a SQL file:

1. If you have not already done so, install the PL/SQL Generator according to the instructions that came with the download.
2. From your browser, go to the URL for the PL/SQL Generator. It should look something like the page shown in Figure 6–1.

Figure 6–1 PL/SQL Generator Page



PL/SQL Generator v1.2

This utility generates installable PL/SQL code for a database provider and its PL/SQL portlets based on the provider and portlet definitions that are stored in the Source XML file.

1. **Source XML File:** This XML file defines the database provider and its PL/SQL portlets using the XML tags that specify values for the provider and portlet record attributes. Click the Browse button and select the XML file.
2. **Provider Name:** Name of the database provider that is to be generated by PL/SQL Generator 1.2. The name should **not contain any spaces**.
3. Click the **Generate** button to generate a SQL file that contains the provider and portlet packages.

3. Click **Browse** and select the source XML file for the **Source XML File** field. Refer to Section 6.2.1, "Creating the Input XML File" for more information on creating the XML file.
4. In the **Provider Name** field, enter the name of the provider. The provider name must not contain any spaces in it. The generator uses the value entered in the Provider Name field for the provider package name.
5. Click **Generate** to generate the SQL file that contains the installable PL/SQL code for the provider and portlet packages. When the browser prompts you to save or open the file, choose **Save**.
6. In the Save dialog box, change the file extension to `.sql` and revise the file name as you wish.
7. Save the file.

6.2.3 Publishing the Generated PL/SQL Portlet

After you have run the PL/SQL Generator and obtained a SQL file, you still need to perform the following tasks to make the provider and portlets available to OracleAS Portal:

- Installing the Packages in the Database
- Registering the Database Provider
- Adding Your Portlet to a Page

6.2.3.1 Installing the Packages in the Database

To install the generated provider and portlet packages into the database where you installed OracleAS Portal, perform the following steps:

1. Start a SQL*Plus session and log in to the PORTAL schema.
2. Create a new database schema, the provider schema, to store the generated provider and portlet packages by entering the following commands in SQL*Plus:

   ```
   create user provider_schema identified by provider_schema_password;
   grant resource, connect to provider_schema;
   ```

3. Grant the EXECUTE privilege for the OracleAS Portal APIs to the provider schema by running the `provsyns.sql` script that is located in the `MID_TIER_ORACLE_HOME/portal/admin/plsql/wwc` directory as follows:

   ```
   @provsyns.sql provider_schema
   ```

4. Log in to the provider schema and run the generated SQL file. It will create the provider and portlet packages in the database.

6.2.3.2 Registering the Database Provider

After creating the provider and portlet packages in the database, you must register the provider with OracleAS Portal before adding the PL/SQL portlet to a portal page:

1. Log in to OracleAS Portal as an administrator.
2. From the Portal Builder, click **Administer > Portlets**.
3. In the **Remote Providers** portlet, click **Register a Provider**.
4. Fill in the **Name**, **Display Name**, **Timeout**, and **Timeout Message** as desired.
5. Choose an **Implementation Style** of **Database**.
6. Click **Next** and complete the remainder of the wizard.
7. When you complete the wizard, click **Finish**.
8. From the Portlet Repository portlet, click **Display Portlet Repository**.
9. Browse the repository and find the provider that you just registered. Typically, new providers appear in the Portlet Staging Area of the repository.
10. Once you find the provider, confirm that it contains all of the portlets you created in the provider. If the provider or its portlets do not appear, then retrace the steps in this section and the preceding sections (Section 6.2.3.1, "Installing the Packages in the Database", Section 6.2.1, "Creating the Input XML File", and Section 6.2.2, "Running the PL/SQL Generator") to ensure that you correctly created and registered your provider and portlet.

6.2.3.3 Adding Your Portlet to a Page

Once your provider and its portlets appear in the repository, you can add it to a page. To add your portlet to a page, follow the instructions in Section 7.6.2, "Adding Portlets," of the *Oracle Application Server Portal User's Guide*.

6.3 Building PL/SQL Portlets Manually

This section describes how to build a basic PL/SQL portlet using the `hello world` sample contained in the `starter` provider sample. The `starter` provider sample, located in `..\pdkplsql\pdk\plsql\starter` in PDK-PL/SQL (`pdkplsql.zip`), consists of the following files:

- `starter_provider.pks` is the package specification of the `starter` provider.
- `starter_provider.pkb` is the package body of the `starter` provider.
- `helloworld_portlet.pks` is the package specification of the `hello world` portlet.
- `helloworld_portlet.pkb` is the package body of the `hello world` portlet.
- `snoop_portlet.pks` is the package specification of the `snoop` portlet.
- `snoop_portlet.pkb` is the package body of the `snoop` portlet.
- `insintpr.sql` is the installation script for the `starter` provider.

The general model for building PL/SQL portlets manually is as follows:

1. Modify the `hello world` portlet package specification and body to create your own portlet package, as described in Section 6.3.1, "Implementing the Portlet Package"
2. Modify the `starter` provider package specification and body to add your new portlet to a provider, as described in Section 6.3.2, "Implementing the Provider Package"
3. Add your portlet to a page, as described in Section 6.3.3, "Adding Your Portlet to a Page"

6.3.1 Implementing the Portlet Package

To modify `helloworld_portlet.pks` and `helloworld_portlet.pkb` to create your own portlet package, perform the following steps:

1. Make copies of the package specification, `helloworld_portlet.pks`, and body, `helloworld_portlet.pkb`.
2. Rename the copies to `my_first_portlet.pks` and `my_first_portlet.pkb`, respectively.
3. Open `my_first_portlet.pks` in an editor and change the name of the package to `my_first_portlet`:

   ```
   CREATE OR REPLACE
   package my_first_portlet
   is
   ...

   end my_first_portlet;
   ```

4. Open `my_first_portlet.pkb` in an editor and repeat the change that you made in the previous step; that is, change the name of the package to `my_first_portlet`.
5. In `my_first_portlet.pkb`, find the function named `get_portlet_info` and modify it as follows:

   ```
   function get_portlet_info
   (
        p_provider_id in integer
       ,p_language in varchar2
   )
   return wwpro_api_provider.portlet_record
   ```

```
is
    l_portlet      wwpro_api_provider.portlet_record;
begin
    l_portlet.id := starter_provider.PORTLET_FIRST;
    l_portlet.provider_id := p_provider_id;
    l_portlet.title := 'My First Portlet';
    l_portlet.name := 'My_First_Portlet';
    ...
```

6. In `my_first_portlet.pkb`, find the procedure named `show` and modify it as follows:

```
procedure show
(
   p_portlet_record       wwpro_api_provider.portlet_runtime_record
)
is
    l_portlet      wwpro_api_provider.portlet_record;
    l_text_name in varchar2(100);
    l_text in varchar2(200);
begin
...
        /*
        Display the content of the portlet in the show mode.
        Use the wwui_api_portlet.portlet_text() API when
        generating the content of the portlet so that the
        output uses the portlet CSS.
        */
        htp.p(wwui_api_portlet.portlet_text(
             p_string   =>  'Hello World - Mode Show'
            ,p_level    =>  1
            ));
        /*
        Add the functionality you want here. In this case we are adding
        a welcome message addressed to the current user.
        */
        l_text_name := 'Welcome to my first portlet ' || wwctx_api.get_user;
        l_text := wwui_api_portlet.portlet_text(
            p_string   =>  l_text_name,
            p_level    =>  1       );
        htp.p(l_text);       htp.para;
        if (p_portlet_record.has_border) then
            wwui_api_portlet.close_portlet;
        end if;
...
```

7. Save `my_first_portlet.pkb`.

6.3.2 Implementing the Provider Package

After you implement the portlet package, you must add your portlet to a provider. To modify `starter_provider.pks` and `starter_provider.pkb` to add your new portlet to a provider, perform the following steps:

1. Make copies of the package specification, `starter_provider.pks`, and body, `starter_provider.pkb`.

2. Rename the copies to `starter_provider2.pks` and `starter_provider2.pkb`, respectively.

Note: If you want to create a new, empty provider, remove all references to the `hello world` and `snoop` portlets from `starter_provider2.pks` and `starter_provider2.pkb` before performing the steps that follow.

3. Open `starter_provider2.pks` in an editor.
4. Add a constant called `PORTLET_FIRST`. This constant is used as the identifier for the portlet within the provider. Hence, the constant's value must be unique within the provider.

```
CREATE OR REPLACE
package STARTER_PROVIDER
is
  /**
    * This package is used as an example to show how providers can be created
    * in the portal system.
    *
    * This provider contains the following portlets:
    *
    * Hello World (PORTLET_HELLOWORLD)
    * Snoop (PORTLET_SNOOP)
    *
    */
    PORTLET_HELLOWORLD constant integer := 1;
    PORTLET_SNOOP      constant integer := 2;
    PORTLET_FIRST      constant integer := 3;
```

5. Save `starter_provider2.pks`.
6. Open `starter_provider2.pkb` in an editor.
7. In `starter_provider2.pkb`, add a call for the new portlet's `get_portlet_info` function in the `get_portlet` function of the provider package. This step entails adding the call `my_first_portlet.get_portlet_info` in the `get_portlet` function. The `get_portlet` function allows the portal to retrieve information for the portlet when necessary.

```
function get_portlet
     p_provider_id in integer
    ,p_portlet_id in integer
    ,p_language in varchar2
)
return wwpro_api_provider.portlet_record
is
begin
    if (p_portlet_id = PORTLET_HELLOWORLD) then
        return helloworld_portlet.get_portlet_info(
             p_provider_id  => p_provider_id
            ,p_language     => p_language
            );
    elsif (p_portlet_id = PORTLET_SNOOP) then
        return snoop_portlet.get_portlet_info(
             p_provider_id  => p_provider_id
            ,p_language     => p_language
            );
    elsif (p_portlet_id = PORTLET_FIRST) then
        return my_first_portlet.get_portlet_info(
             p_provider_id  => p_provider_id
```

```
                ,p_language     => p_language
                );
        else
            raise wwpro_api_provider.PORTLET_NOT_FOUND_EXCEPTION;
        end if;
    end get_portlet;
```

8. In `starter_provider2.pkb`, add the new portlet to the list of portlets returned by the provider. This step entails adding the new portlet to the `get_portlet_list` function of the provider. The `get_portlet_list` function tells the portal which portlets the provider implements.

```
function get_portlet_list
...
begin
    l_cnt := 0;
    if (p_security_level = false ) then
            l_cnt := l_cnt + 1;
            l_portlet_list(l_cnt) := get_portlet(
                 p_provider_id  => p_provider_id
                ,p_portlet_id   => PORTLET_HELLOWORLD
                ,p_language     => p_language
                );
            l_cnt := l_cnt + 1;
            l_portlet_list(l_cnt) := get_portlet(
                 p_provider_id  => p_provider_id
                ,p_portlet_id   => PORTLET_SNOOP
                ,p_language     => p_language
                );
            l_cnt := l_cnt + 1;
            l_portlet_list(l_cnt) := get_portlet(
                 p_provider_id  => p_provider_id
                ,p_portlet_id   => PORTLET_FIRST
                ,p_language     => p_language
                );
    else
        if (helloworld_portlet.is_runnable(
             p_provider_id     =>  p_provider_id
            ,p_reference_path  =>  null)
        ) then
            l_cnt := l_cnt + 1;
            l_portlet_list(l_cnt) := get_portlet(
                 p_provider_id  => p_provider_id
                ,p_portlet_id   => PORTLET_HELLOWORLD
                ,p_language     => p_language
                );
        end if;
        if (snoop_portlet.is_runnable
             p_provider_id     =>  p_provider_id
            ,p_reference_path  =>  null)
        ) then
            l_cnt := l_cnt + 1;
            l_portlet_list(l_cnt) := get_portlet(
                 p_provider_id  => p_provider_id
                    ,p_portlet_id   => PORTLET_SNOOP
                    ,p_language     => p_language
                    );
        end if;
        if (my_first_portlet.is_runnable(
             p_provider_id     =>  p_provider_id
```

```
                ,p_reference_path  =>  null)
            ) then
                l_cnt := l_cnt + 1;
                l_portlet_list(l_cnt) := get_portlet(
                     p_provider_id  => p_provider_id
                    ,p_portlet_id   => PORTLET_FIRST
                    ,p_language     => p_language
                    );
            end if;
        end if;
        return l_portlet_list;
    end get_portlet_list;
```

9. In `starter_provider2.pkb`, modify the `is_portlet_runnable` function to add a call to the `is_runnable` function of the new portlet.

```
function is_portlet_runnable
(
    p_portlet_instance in wwpro_api_provider.portlet_instance_record
)
return boolean
is
begin
    if (p_portlet_instance.portlet_id = PORTLET_HELLOWORLD) then
        return helloworld_portlet.is_runnable(
             p_provider_id     =>  p_portlet_instance.provider_id
            ,p_reference_path  =>  p_portlet_instance.reference_path
            );
    elsif (p_portlet_instance.portlet_id = PORTLET_SNOOP) then
        return snoop_portlet.is_runnable(
             p_provider_id     =>  p_portlet_instance.provider_id
            ,p_reference_path  =>  p_portlet_instance.reference_path
            );
    elsif (p_portlet_instance.portlet_id = PORTLET_FIRST) then
        return my_first_portlet.is_runnable(
             p_provider_id     =>  p_portlet_instance.provider_id
            ,p_reference_path  =>  p_portlet_instance.reference_path
            );
    else
        raise wwpro_api_provider.PORTLET_NOT_FOUND_EXCEPTION;
    end if;
end is_portlet_runnable;
```

10. Repeat step 9 according to the information in Table 6–3.

Table 6–3 Changes to starter_provider2.pkb

Procedure/Function	Addition
`procedure register_portlet`	`elsif (p_portlet_instance.portlet_id = PORTLET_FIRST) then my_first_portlet.register(p_portlet_instance)`
`procedure deregister_portlet`	`elsif (p_portlet_instance.portlet_id = PORTLET_FIRST) then my_first_portlet.deregister (p_portlet_instance)`
`function describe_portlet_ parameters`	`elsif (p_portlet_id = PORTLET_FIRST) then return my_first_portlet.describe_parameters (p_provider_id, p_language);`

Table 6–3 (Cont.) Changes to starter_provider2.pkb

Procedure/Function	Addition
`procedure show_portlet`	`elsif (p_portlet_record.portlet_id = PORTLET_FIRST) then my_first_portlet.show(p_portlet_record)`
`procedure copy_portlet`	`elsif (p_copy_portlet_info.portlet_id = PORTLET_FIRST) then my_first_portlet.copy(p_portlet_record)`

11. Save and close `starter_provider2.pkb`.
12. Log in to OracleAS Portal as you normally would.
13. From the Portal Builder, click **Administer > Portlets**.
14. From the Portlet Repository portlet, click **Display Portlet Repository**.
15. Browse the repository and find the starter provider (typically it will appear in the Portlet Staging Area of the repository). It should contain its two original portlets: `hello world` and `snoop`.
16. From a command line prompt, start SQL*Plus and connect as the owner of the `starter` provider schema.
17. Compile the new and modified PL/SQL packages in the following order:
    - `starter_provider2.pks`
    - `my_first_portlet.pks`
    - `starter_provider2.pkb`
    - `my_first_portlet.pkb`
18. If any compilation errors occur, fix and recompile them until all of the packages compile successfully.
19. From the Portlet Repository portlet, click **Display Portlet Repository**.
20. Browse the repository and find the `starter` provider again. It should now contain your new portlet, `my_first_portlet`, in addition to its original portlets.

Note: If you make changes to an existing provider or the portlet record, you need to refresh your provider before seeing the changes reflected in your OracleAS Portal instance.

6.3.3 Adding Your Portlet to a Page

Once your portlet appears in the repository, you can add it to a page. To add your portlet to a page, follow the instructions in Section 7.6.2, "Adding Portlets," of the Oracle Application Server Portal User's Guide.

6.4 Implementing Information Storage

OracleAS Portal provides APIs for storing and retrieving individual portlet preferences, and storing and manipulating temporary data for the current session:

- Implementing a Preference Store
- Implementing a Session Store

6.4.1 Implementing a Preference Store

OracleAS Portal provides a set of APIs for storing and retrieving individual preferences for each unique portlet instance in a persistent manner. It provides a unique identifier for each individual, a preference store automatically mapped by user, and access mechanisms for storing and retrieving personalization information in your PL/SQL portlets.

By default, when you enable end-user personalization, **Customize** appears on the title bar of your portlet. This link displays a form where users can choose settings for that portlet.

End-user personalization options are available through the `wwpre_api_name` and `wwpre_api_value` packages.

6.4.1.1 Using a Preference Store

In general, you can set up preference storage as follows:

1. Create the preference path through `wwpre_api_name.create_path`.
2. Create the preference with `wwpre_api_name.create_name`.
3. Set the preference values by providing the preference name and scoping level for which you want to set the value. Use `wwpre_api_value.set_value_as_varchar2`, `set_value_as_number`, or `set_value_as_date` for this purpose.
4. Get preference values by providing the preference name and path whenever you want to retrieve the preference value. Use `wwpre_api_value.get_value_as_varchar2`, `get_value_as_number`, or `get_value_as_date` for this purpose.

6.4.1.2 Creating and Accessing a Preference Store

The services example, located in `..\pdkplsql\pdk\plsql\svcex` in PDK-PL/SQL (`pdkplsql.zip`), illustrates how you can implement preference storage. The objective is to achieve the following functionality:

- When a user clicks **Customize**, they can enter text in two fields.
- The first field prompts for personalized text. The second prompts for a personalized portlet title.
- The values the user enters for these two fields is stored in the preference store.
- The personalized text and portlet titles are retrieved whenever that user invokes the portlet instance.

You can browse through this example as follows to see how to create the preference store, store values in it, and retrieve values from it:

1. Open the `services_portlet.pkb` file in an editor.
2. The portlet path and preference names are provided with aliases in the constants part of your portlet definition.

```
DOMAIN           constant varchar2(30) := 'provider';
SUBDOMAIN        constant varchar2(32) := 'services';
PORTLET_PATH     constant varchar2(256):= 'oracle.portal.pdk.servicesportlet';
PREFNAME_STRING  constant varchar2(30) := 'services_string';
PREFNAME_TITLE   constant varchar2(30) := 'services_title';
```

3. Find the `register` procedure. Your portlet needs to create a path for storing your preferences. To do so, it calls `wwpre_api_name.create_path` for creating the preference path. It then calls `wwpre_api_name.create_name` for creating the

preference name, taking the portlet path, name, and description as input parameters. Another input parameter is the `p_type_name` that indicates special value types. The `NLSID` type indicates that the value stored is a Globalization Support ID. The functions for setting and retrieving this type treat it as a number value. Apart from that, when a preference store value of this type is exported or copied, so are its associated strings. The last input parameter, the language, is obtained from a context API.

```
procedure register
(
    p_portlet_instance in wwpro_api_provider.portlet_instance_record
)
is
begin
    --
    -- Create a path for the portlet instance.  This is used to create
    -- the preferences for the portlet instance in the preference store.
    --
    wwpre_api_name.create_path(
        p_path => PORTLET_PATH || p_portlet_instance.reference_path
        );
    --
    -- Create the names to store the portlet preferences.
    --
    wwpre_api_name.create_name(
        p_path        => PORTLET_PATH
            || p_portlet_instance.reference_path,
        p_name        => PREFNAME_STRING,
        p_description => 'Single custom row in '
            || 'Introductory Example portlet.',
        p_type_name   => 'NLSID',
        p_language    => wwctx_api.get_nls_language);
    wwpre_api_name.create_name(
        p_path        => PORTLET_PATH
            || p_portlet_instance.reference_path,
        p_name        => PREFNAME_TITLE,
        p_description => 'Single custom row in '
            || 'Introductory Example portlet.',
        p_type_name   => 'NLSID',
        p_language    => wwctx_api.get_nls_language);
exception
    when others then
        raise;
end register;
```

4. The deregister procedure must eliminate the preference store with a call to `wwpre_api_name.delete_name`.

```
procedure deregister
(
    p_portlet_instance in wwpro_api_provider.portlet_instance_record
)
is
begin
     --
     -- Delete the path used by the portlet instance.  This will delete
     -- all the names and all the values associated with the path.
     --
     wwpre_api_name.delete_path(
        p_path => PORTLET_PATH || p_portlet_instance.reference_path
        );
exception
    when others then
        raise;
end deregister;
```

5. The portlet must also get and set the values in the preference store using `wwpre_api_value.set_value` and `wwpre_api_value.get_value`. Find the `get_default_preference` function. Notice how this function loads the system level default values from the preference store. The default preferences are associated with an instance. The language strings are set in the database.

```
function get_default_preference
...
begin
    --
    -- Try to find a previously entered portlet instance string preference,
    -- if any.
    -- A portlet instance string preference is stored in the preference
    -- store and has a level of SYSTEM_LEVEL_TYPE.
    --
        p_path       => PORTLET_PATH || p_reference_path,
    l_prefs.string_id := to_char(wwpre_api_value.get_value_as_number(
        p_name       => PREFNAME_STRING,
        p_level_type => wwpre_api_value.SYSTEM_LEVEL_TYPE
        ));
    --
    -- If the value returned above is null it is an indication that there
    -- is no default string yet.  Initialize the string id to 0 to indicate
    -- this and load the default string value.
    --
    if (l_prefs.string_id is null or to_number(l_prefs.string_id) = 0) then
        wwpre_api_value.set_value_as_number(
            p_path       => PORTLET_PATH || p_reference_path,
            p_name       => PREFNAME_STRING,
            p_level_type => wwpre_api_value.SYSTEM_LEVEL_TYPE,
            p_level_name => null,
            p_value      => 0
            );
...
end get_default_preference;
```

6. Find the `show` procedure. Notice the behavior when the portlet is in Edit Defaults or Edit mode. Note also how `p_action` is populated when **APPLY**, **CANCEL**, or **OK** is clicked. Once the form is submitted, the `show` procedure of the portlet is called again and, if the `p_action` parameter is not null, then the `save_prefs` procedure is called to save the customizations and redirect to the relevant page.

```
procedure show
(
    p_portlet_record        wwpro_api_provider.portlet_runtime_record
)
is
    l_str varchar2(32000);
    l_pref_record preference_record;
    l_action   varchar2(10);
    l_names    owa.vc_arr;
    l_values   owa.vc_arr;
begin
...
```

```
        elsif (p_portlet_record.exec_mode =
                        wwpro_api_provider.MODE_SHOW_EDIT)
            or (p_portlet_record.exec_mode =
                        wwpro_api_provider.MODE_SHOW_EDIT_DEFAULTS)
        then
            wwpro_api_parameters.retrieve(l_names, l_values);
            for i in 1..l_names.count loop
                if (upper(l_names(i)) = upper('p_string')) then
                    l_pref_record.string := l_values(i);
                elsif l_names(i) = 'p_title' then
                    l_pref_record.title_string := l_values(i);
                elsif l_names(i) = 'p_action' then
                    l_action := l_values(i);
                end if;
            end loop;
            if (l_action in (ACTION_OK,ACTION_APPLY,ACTION_CANCEL)) then
                if (p_portlet_record.exec_mode =
                                    wwpro_api_provider.MODE_SHOW_EDIT) then
                    save_prefs(p_string => l_pref_record.string,
                               p_title  => l_pref_record.title_string,
                               p_action => l_action,
                               p_level  => wwpre_api_value.USER_LEVEL_TYPE,
                               p_portlet_record  => p_portlet_record);
                else
                    save_prefs(p_string => l_pref_record.string,
                               p_title  => l_pref_record.title_string,
                               p_action => l_action,
                               p_level  => wwpre_api_value.SYSTEM_LEVEL_TYPE,
                               p_portlet_record  => p_portlet_record);
                end if;
            else
                show_edit(p_portlet_record  => p_portlet_record);
            end if;
    ...
    end show;
```

7. The `show_edit` procedure renders the page for Edit or Edit Defaults mode. It renders two text fields that allow the user to change the customizable values in a form with three buttons (**Apply**, **OK**, and **Cancel**). Note that this function uses the `wwpro_api_adapter.open_form` to create the HTML form with the correct action attribute for the `<FORM>` tag and with the correct hidden fields. It is important to use this procedure to create the `<FORM>` tag if you want to use the portlet with the Federated Portal Adapter from remote OracleAS Portal instances.

```
procedure show_edit
(
    p_portlet_record in wwpro_api_provider.portlet_runtime_record
)
is
    l_prefs    preference_record;
    l_text_prompt_string  varchar2(30);
    l_title_prompt_string varchar2(30);
begin
...
    htp.centeropen;
    htp.tableOpen(cattributes => 'BORDER="1" WIDTH=90%');
    htp.tableRowOpen;
    htp.p('<TD>');
    --
    --  This procedure call creates the <FORM> tags with a set of
    --  standard parameters.  Using this procedure makes the
    --  customisation page work through the pl/sql http adapter.
    --
    wwpro_api_adapter.open_form(p_formattr => 'NAME="services"',
                                p_prr      => p_portlet_record);
    htp.p('</TD>');
    htp.tableRowClose;
    htp.tableClose;
    htp.centerclose;
    htp.formclose;
end show_edit;
```

8. Review the following procedures and functions, which are related to the preference storage implementation in this example:
   - `get_user_preference` retrieves the user customized string and title for the portlet.
   - `save_prefs` is invoked to save the preferences to the preference store when the user clicks **OK** or **Apply** after making customization changes.
   - `entered_text_is_valid` checks to see if the text entered in the customizable text fields is valid.
9. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 6.3.2, "Implementing the Provider Package" for information on how to add it.
10. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in Section 7.6.2, "Adding Portlets," of the *Oracle Application Server Portal User's Guide*.

6.4.2 Implementing a Session Store

The services example, located in `..\pdkplsql\pdk\plsql\svcex` in PDK-PL/SQL (`pdkplsql.zip`), illustrates how you can implement a session store. The objective is to achieve the following functionality:

- When a user invokes this portlet, it displays text that reads: "This portlet has rendered *x* times in this session." *x* is the number of times the portlet has been rendered.
- Every time the user invokes the portlet, the counter increases by 1.
- Clicking **Details** in the portlet enables the user to reset the counter through **Clear**. After clearing the counter, the counter starts again from zero.

6.4.2.1 Creating and Accessing a Session Store

You can browse through this example as follows to see how to create the session store, store values in it, and retrieve values from it:

1. Open the `services_portlet.pkb` file in an editor.
2. The domain and subdomain definitions for your session object are provided with aliases in the constants part of your portlet definition.

```
DOMAIN          constant varchar2(30) := 'provider';
SUBDOMAIN       constant varchar2(32) := 'services';
PORTLET_PATH    constant varchar2(256):= 'oracle.portal.pdk.servicesportlet';
PREFNAME_STRING constant varchar2(30) := 'services_string';
PREFNAME_TITLE  constant varchar2(30) := 'services_title';
```

3. Find the `clear_count` procedure. `clear_count` is called from the `show` procedure when the user clicks **Clear** to reset the counter. First, `clear_count` calls `wwsto_api_session.load_session` to load the session object. Second, it calls `wwsto_api_session.set_attribute` to set the counter to zero. Third, it saves the session object by calling `save_session`.

```
procedure clear_count
(
    p_action          in varchar2,
    p_back_url        in varchar2,
    p_reference_path  in varchar2
)
is
    ex_counter integer;
    session_parms &&1..wwsto_api_session;
begin
    --
    -- Clear the display counter.
    --
    if (p_action = ACTION_CLEAR) then
        --
        -- Load the session object that contains the display counter
        --
        session_parms :=
            &&1..wwsto_api_session.load_session (DOMAIN,SUBDOMAIN);
        ex_counter :=
            session_parms.get_attribute_as_number(
                'ex_counter' || p_reference_path);
        --
        -- Reset the display counter.
        --
        ex_counter := 0;
        session_parms.set_attribute(
        'ex_counter' || p_reference_path, ex_counter);
        --
        -- Save the changes to the database immediately to avoid any
        -- data consistency problems with the data stored in the
        -- session object.
        --
        session_parms.save_session;
    end if;
    owa_util.redirect_url(curl=>p_back_url);
end clear_count;
```

4. Find the `show_contents` procedure. `show_contents` is called from the `show` procedure to retrieve the counter, increment it by one, and save the value in the session store. Notice how it retrieves the session object to display the number of times the user has rendered the portlet. It also retrieves the counter value with `get_attribute_as_number` and increments the counter for every invocation of this procedure.

```
procedure show_contents
(
    p_portlet_record  wwpro_api_provider.portlet_runtime_record
)
is
    l_prefs preference_record;
    session_parms &&1..wwsto_api_session;
    ex_counter integer;
    l_portlet wwpro_api_provider.portlet_record;
    l_str varchar2(32000);
begin
    --
    -- In this mode a session counter is used to indicate
    -- the number of invocations of this portlet during the
    -- current session.  The counter is stored in the session
    -- store.
    --
    session_parms :=
        &&1..wwsto_api_session.load_session(DOMAIN,SUBDOMAIN);
    ex_counter    :=
        session_parms.get_attribute_as_number(
            'ex_counter' || p_portlet_record.reference_path);
    if (ex_counter is null) then -- first invocation
        session_parms.set_attribute(
            'ex_counter' || p_portlet_record.reference_path,1);
        ex_counter := session_parms.get_attribute_as_number(
            'ex_counter' || p_portlet_record.reference_path);
    else -- on every invocation increase by 1
        ex_counter := ex_counter + 1;
        session_parms.set_attribute(
            'ex_counter'
            || p_portlet_record.reference_path, ex_counter);
    end if;
    session_parms.save_session;
...
end show_contents;
```

5. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 6.3.2, "Implementing the Provider Package" for information on how to add it.
6. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in Section 7.6.2, "Adding Portlets," of the Oracle Application Server Portal User's Guide.

6.5 Using Parameters

The functionality of portlets can be extended with the help of parameters. The business logic implemented by portlets may produce different HTML output depending on the parameters passed to the page. By using the portlet parameters, you can navigate within the portlet in the Shared Screen mode without changing the current page. Portlets can also communicate with each other through parameters.

Portlet parameters are structured as name-value pairs. These pairs map directly to the URL parameter passing format by using the `GET` submission method or can use the HTTP message body by using the `POST` submission method. Portlets can also expose their parameters to OracleAS Portal. When added to a page, these portlets can accept values in the form of page parameters created by the page designer.

Portlets do not have direct access to the URL, the HTTP message body, or the page parameters. To retrieve the parameter values, portlets must call the OracleAS Portal PL/SQL parameter APIs provided in the `wwpro_api_parameters` package.

OracleAS Portal offers the following types of parameters:

WARNING: You cannot mix the usage of public and private parameters in a portlet. To enable public parameters for your portlet, you must take steps that preclude the usage of private parameters and vice versa.

- **Private portlet parameters** enable the implementation of internal navigation in your portlet.
- **Public portlet parameters** let you pass control over the data flow of your portlet to the page designer. The page designer can map the public portlet parameters to their page parameters, provide default values, and allow users to customize those values.
- **Page parameters** are defined in a simple user interface by page designers. These page parameters can be mapped to public portlet parameters in order for the page designer to pass parameter values from the page to the portlets on it.

For more information about parameters, refer to Section 2.12, "Public Portlet Parameters Support" and Section 2.13, "Private Portlet Parameter Support".

6.5.1 Passing Private Parameters

You can use either `GET` or `POST` HTML submission methods when passing private portlet parameters. The `GET` method uses the URL to pass the parameters, whereas the `POST` method places the parameters in an HTTP message body. For both methods, you must specify the portlet instance on the portal page, how the parameter is called, and the value of the parameter.

There are two types of private portlet parameters:

- **Qualified parameters** ensure that a private portlet parameter is not read by any other portlet on the page. The reference path, which is assigned when the portlet is added to a page, is the unique prefix of the parameter. For example, `http://page_url?277_MAP_368673.region=Europe`. The qualified parameter's reference path is `277_MAP_368673`, the name is region, and the value is `Europe`. For private parameters, we strongly recommend that you always use qualified parameters.
- **Unqualified parameters** have no information about the portlet instance and can be read by any portlet on the page. For example, `http://page_url?region=Europe`. The unqualified parameter's name is region and its value is `Europe`. For private parameters, we strongly recommend that you avoid unqualified parameters.

6.5.2 Passing Page Parameters and Mapping Public Portlet Parameters

Public portlet parameters enhance the flexibility of your portlets by enabling page designers to reuse your portlets on multiple pages. As a result, page designers do not have to ask you to make changes to the portlet code when adding the portlet to different pages. By using public portlet parameters, any portlet on a page can easily receive its value from the mapped page parameter, regardless of the portlet parameter name.

For example, suppose you have a page parameter named `dept_id`. Three portlets need this value, but one portlet calls it `dept`, another calls it `deptno`, and still another `department_id`. Mapping the page parameter enables all three portlets to receive the value from the `dept_id` parameter and place it in the appropriate portlet parameter.

Furthermore, the page designer may set a default value (for example, department 20) that can be customized by users (for example, department 30) and applied to all three portlets.

The general model for passing public and page parameters is as follows:

1. Enable public parameters in the portlet record by setting `pass_all_url_params` to `false`. This ensures that the portlet is only passed parameters intended for that portlet.
2. Declare the public parameters in the provider's `describe_portlet_parameters` function. For each of the portlets that belong to the provider, this procedure should return a list of the parameters that the portlet accepts in the form of a PL/SQL table of records of the type:

   ```
   type portlet_parameter_table is table of
   portlet_parameter_record index by binary_integer;
   ```

3. Provide descriptive information for the parameters in the portlet's `describe_parameters` function. For example:

   ```
   function describe_parameters
      (p_provider_id in integer, p_language in varchar2)
   return wwpro_api_provider.portlet_parameter_table
      is
   l_params wwpro_api_provider.portlet_parameter_table;
      begin
        l_params(1).name := 'dept_id';
        l_params(1).datatype := wwpro_api_provider.STRING_TYPE;
        l_params(1).description := 'Defines a department ID';
        l_params(1).display_name := 'Department ID';
        return l_params;
   end describe_parameters;
   ```

4. Assign values to the public parameters. Public parameters typically get their values through page parameters. Page parameters are usually assigned default values by the page designer and the user can then customize the value at runtime. Alternatively, page parameter values can be assigned in the calling URL. For more information about how page designers can use page parameters, refer to the *Oracle Application Server Portal User's Guide*.

6.5.3 Retrieving Parameter Values

Regardless of whether you are using private or public parameters, you use the same APIs to retrieve their values. Portlets obtain their parameters by calling the PL/SQL parameter APIs in the `wwpro_api_parameters` package:

- `wwpro_api_parameters_get_value` returns the parameter value that is specified by a given parameter name. Parameter names are not case sensitive, whereas parameter values are case sensitive. For example:

  ```
  l_region := wwpro_api_parameters.get_value
     (p_name => 'region',
      p_reference_path => p_portlet_record.reference_path);
  ```

- `wwpro_api_parameters_get_values` returns an array of parameter values. This function returns all the values that are associated with a single parameter name or an empty list if no matches are found. Some business logic may require multiple selections, when multiple values are passed to the portlet by using the same parameter name. Portlets can take one or more values of the same parameter. For example:

```
l_region_values owa.vc_arr;
...
l_region_values := wwpro_api_parameters.get_values
   (p_name = 'region',
    p_reference_path => p_portlet_record.reference_path);
```

- `wwpro_api_parameters_get_names` returns the names of the parameters that are passed on to a specified portlet that is identified by the reference path. The returned list is a PL/SQL table of the `owa.vc_ar` type that is defined as follows:

```
type vc_arr is table of varchar2(32000) index by binary_integer;
```

Note: Portlet parameter names should not start with an underscore (_) because those parameters are reserved for internal use by OracleAS Portal, and are not passed to the portlet.

For example:

```
l_names owa.vc_arr;
...
l_names := wwpro_api_parameters.get_names
   (p_reference_path => p_portlet_record.reference_path);
```

- `wwpro_api_parameters.retrieve` returns the names and values of all of the portlet's parameters. For example:

```
procedure show_portlet
   ( p_portlet_record in out
         wwpro_api_provider.portlet_runtime_record )
is
   l_names owa.vc_arr;
   l_values owa.vc_arr;
...
begin
...
   wwpro_api_parameters.retrieve (l_names, l_values);
   for i in 1..l_names.count loop
     htp.p('Parameter Name: '||l_names(i));
     htp.p('Parameter Value: '||l_values(i));
     htp.br;
   end loop;
...
end show_portlet;
```

6.6 Accessing Context Information

Whenever a user accesses a page in OracleAS Portal, a public session is established. When the user logs in to OracleAS Portal, the public session becomes an authenticated session. This session contains several pieces of context information about the user, such as username, current session ID, IP address, and language preference. It also includes supporting information such as the OracleAS Portal schema currently in use.

Session context services return information about a user's session and are available through the `wwctx_api` package.

6.6.1 Using Context Information

The general model for working with the session context is as follows:

1. Identify the piece of information you require for your functionality.
2. Use the appropriate method from `wwctx_api` to get and optionally set this value.

Table 6–4 lists the function calls used to obtain the various pieces of session information.

> **Note:** For more information on the context APIs, see the PL/SQL API Reference. The API Reference can be found on Portal Center (`http://www.oracle.com/technology/products/ias/portal/index.html`) or, if you downloaded PDK-PL/SQL (`pdkplsql.zip`), in `..\pdkplsql\pdk\plsql\doc`.

Table 6–4 Context Information Function Calls

Session Information	Function Call
Current user	`wwctx_api.get_user`
Login status of user	`wwctx_api.is_logged_on`
Login time	`wwctx_api.get_login_time`
Language	`wwctx_api.get_nls_language`
Current session id	`wwctx_api.get_sessionid`
IP address of user client	`wwctx_api.get_ip_address`
User schema	`wwctx_api.get_db_user`
OracleAS Portal schema	`wwctx_api.get_product_schema`
OracleAS Portal version	`wwctx_api.get_product_version`

6.6.2 Using wwctx_api to Obtain Context Information

The services example, located in `..\pdkplsql\pdk\plsql\svcex` in PDK-PL/SQL (`pdkplsql.zip`), illustrates how you can obtain session information using the `wwwctx_api` package. You can browse through this example as follows to see how the function calls are implemented in a portlet:

1. Open the `services_portlet.pkb` file in an editor.
2. Find the `get_portlet_info` function.
3. Notice the usage of `wwctx_api.get_user` to derive the user information and set that value in the portlet information record:

```
...
    l_portlet.timeout               := null;
    l_portlet.timeout_msg           := null;
    l_portlet.created_on            := to_date('10/19/2000', 'MM/DD/YYYY');
    l_portlet.created_by            := wwctx_api.get_user;
    l_portlet.last_updated_on       := to_date('10/19/2000', 'MM/DD/YYYY');
    l_portlet.last_updated_by       := wwctx_api.get_user;
    l_portlet.has_show_edit_defaults := true;
    l_portlet.has_show_preview      := true;
    l_portlet.preference_store_path := PORTLET_PATH;
...
```

4. `wwctx_api.get_user` is used similarly in various places throughout `services_portlet.pkb`. Search the code for other occurrences of `wwctx_api.get_user`.

5. Another example of getting context information occurs in the `is_runnable` function:

```
function is_runnable
(
     p_provider_id in integer
    ,p_reference_path in varchar2
)
return boolean
is
begin
    --
    -- Portlet security check.  It allows the portlet to be visible
    -- if the user is logged on (that is, the current session is not a
    -- public session).
    --
    return wwctx_api.is_logged_on;
end is_runnable;
```

6. In the `register` procedure, `wwctx_api.get_nls_language` is used to get the language:

```
    --
    -- Create the names to store the portlet preferences.
    --
    wwpre_api_name.create_name(
        p_path        => PORTLET_PATH
            || p_portlet_instance.reference_path,
        p_name        => PREFNAME_STRING,
        p_description => 'Single custom row in '
            || 'Introductory Example portlet.',
        p_type_name   => 'NLSID',
        p_language    => wwctx_api.get_nls_language);
    wwpre_api_name.create_name(
        p_path        => PORTLET_PATH
            || p_portlet_instance.reference_path,
        p_name        => PREFNAME_TITLE,
        p_description => 'Single custom row in '
            || 'Introductory Example portlet.',
        p_type_name   => 'NLSID',
        p_language    => wwctx_api.get_nls_language);
```

7. Close `services_portlet.pkb`. You can implement session context in similar fashion to what is illustrated here, but based upon your own functional requirements.

8. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 6.3.2, "Implementing the Provider Package" for information on how to add it.

9. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in Section 7.6.2, "Adding Portlets," of the *Oracle Application Server Portal User's Guide*.

6.7 Implementing Portlet Security

Portlet security refers to the techniques and methods used by portlets to control their access by end users. The portlets leave authentication to OracleAS Portal and trust that the portal will return the portlet to the correct, validated user upon request.

OracleAS Portal strictly controls access to information and applications by assigning specific privileges to users and groups. Portal security services allow you to specify access control programmatically and check for the appropriate privileges at runtime. The security mechanisms used by portlets ensure that only authorized users gain access to these portlets. These security services are available through the `wwsec_api` package.

Portlet security is invoked when a portlet is displayed on a portal page and when a portlet is returned in a portlet list by the `get_portlet_list` function for database providers. Security services in the Portal framework have the following key features:

- **Portlet Display:** Before a portlet is displayed on a page, the provider checks for the portlet’s access privileges. The provider needs to define the `is_portlet_runnable` function which calls the portlet's `is_runnable` function to check access privileges.
- **User Group:** You can find which default group a user belongs to by using the `wwsec_api.get_defaultgroup` function.
- **Check Privileges:** You can find whether a user/group has the required privileges to customize a portlet by using the `wwsec_api.has_privilege` function.
- **Highest Privilege:** You can find the highest available privilege of a user across all groups by using the `wwsec_api.get_privilege_level` function.
- **Accessible Objects:** You can find all the objects to which a user has access, given a privilege level, by using the `wwsec_api.accessible_objects` function. You can find other similar associated functions in the API documentation. The API Reference can be found on Portal Center (`http://www.oracle.com/technology/products/ias/portal/index.html`) or, if you downloaded PDK-PL/SQL (`pdkplsql.zip`), in `..\pdkplsql\pdk\plsql\doc`.

6.7.1 Using Security

To implement PL/SQL portlet security, the portal requires the function `is_portlet_runnable` be implemented by database providers. The actual implementation of this function is up to the application; that is, the security scheme that determines whether the current user has enough privileges to access the portlet is defined by the individual portlet implementation. The portal also requires the function `get_portlet_list` for database providers to return the set of portlets that are accessible by the current user.

6.7.1.1 Guidelines for Using the Security APIs

The portlet security mechanism may use the context and security subsystem APIs and infrastructure. The context APIs can be used to retrieve information about the current user. The security subsystem can be used to check the privileges of the current user.

Note: For more information on the context and security subsystem APIs, see the PL/SQL API Reference. The API Reference can be found on Portal Center (`http://www.oracle.com/technology/products/ias/portal/index.html`) or, if you downloaded PDK-PL/SQL (`pdkplsql.zip`), in `..\pdkplsql\pdk\plsql\doc`.

While using these APIs, keep in mind the following:

- Only authorized users should be able to see your portlet in the Add Portlet dialog. This objective can be accomplished by implementing the `is_portlet_runnable` function in the provider. You can also allow public access to your portlet.
- If a portlet does not want to render itself to a user, it should return no HTML or return an exception that the page engine will ignore. It should not return an error message. Doing so adds unnecessarily to the error stack, which has its limits. Refer to Section 6.9, "Implementing Error Handling" for more information.
- Portlet security allows the portlet to perform a runtime security check to ensure that the current user has the necessary authorization to access the portlet.
- When a portlet is rendered in Show mode, it may call the `is_runnable` method for database providers to determine whether the portlet should be displayed for the currently logged on user. The portal does not make the call to this function directly. It is not a requirement, however, for the portlet to make this call. The portlet should make this call in its Show mode only if it implements portlet security.
- The result of the call to `is_runnable` determines whether the portlet is actually displayed. If the result is true, the portlet displays, otherwise it does not display. The portlet is rendered in Show mode when it is displayed in a portal page.
- When a portlet is returned in a portlet list by a call to the provider function `get_portlet_list`, the value of the `p_security_level` parameter determines the purpose of the function call. When the call is made from the Portlet Repository refresh operation in order to retrieve the master list of portlets that the provider implements, the parameter `p_security_level` has a value of `false`. This setting indicates to the provider that no portlet security check should be made and a master list of all the portlets that the provider implements must be returned. The master list of portlets returned in this case is used to populate the Portlet Repository for that provider.
- If the value of `p_security_level` is `true`, then it is up to the provider implementation to decide whether portlet security should be performed. If portlet security is implemented, the provider may return a different list of portlets depending on the current user.
- When the Portlet Repository is displayed, OracleAS Portal calls the `is_portlet_runnable` function for database providers for each of the portlets that exist in the Portlet Repository. This step is done to display only the portlets that the currently logged on user is authorized to see. One example where the Portlet Repository is displayed is in the Add Portlets dialog.

6.7.2 Coding Security

The services example, located in `..\pdkplsql\pdk\plsql\svcex` in PDK-PL/SQL (`pdkplsql.zip`), illustrates how you can implement security. You can browse through this example as follows to see how the security functions are implemented in a portlet:

1. Open the `services_provider.pkb` file in an editor.
2. Find the `is_portlet_runnable` function. This function calls the security implementation through the portlet's `is_runnable` function to check portlet access privileges.

```
function is_portlet_runnable
(
    p_portlet_instance in wwpro_api_provider.portlet_instance_record
)
return boolean
is
begin
    if (p_portlet_instance.portlet_id = SERVICES_PORTLET_ID) then
        return services_portlet.is_runnable(
             p_provider_id     =>  p_portlet_instance.provider_id
            ,p_reference_path  =>  p_portlet_instance.reference_path
            );
    else
        raise wwpro_api_provider.PORTLET_NOT_FOUND_EXCEPTION;
    end if;
end is_portlet_runnable;
```

3. Find the `get_portlet_list` procedure. `get_portlet_list` allows the portlet to be included in the list of portlets implemented by this provider. `get_portlet_list` first checks the security flag (`p_security_level`) to find out whether security is enabled. If the flag is set to true, `get_portlet_list` uses `is_runnable` to check whether the portlet is accessible. The value of the `p_security_level` parameter indicates whether to perform security checks before returning a portlet in the list. When a portlet repository refresh operation retrieves the master list of portlets implemented by the provider, `p_security_level` has a value of `false`. A value of `false` means the provider need not perform a security check and that a master list of all of the portlets implemented by the provider must be returned. The master list of portlets returned is used to populate the portlet repository for that provider. If the value of `p_security_level` is true, then the provider implementation decides whether to perform portlet security checks. If portlet security is implemented, the provider may return a different list of portlets depending on the currently logged on user.

```
function get_portlet_list
...
    if (p_security_level = false) then
        l_cnt := l_cnt + 1;
        l_portlet_list(l_cnt) := get_portlet(
             p_provider_id  => p_provider_id
            ,p_portlet_id   => SERVICES_PORTLET_ID
            ,p_language     => p_language
            );
    else
        if (services_portlet.is_runnable(
             p_provider_id     =>  p_provider_id
            ,p_reference_path  =>  null)
        ) then
            l_cnt := l_cnt + 1;
            l_portlet_list(l_cnt) := get_portlet(
                 p_provider_id  => p_provider_id
                ,p_portlet_id   => SERVICES_PORTLET_ID
                ,p_language     => p_language
                );
        end if;
...
end get_portlet_list;
```

4. Open the `services_portlet.pkb` file in an editor.
5. Find the `show` procedure. Before displaying a portlet, the `show` procedure runs a security check to determine whether the current user is allowed to see the portlet.

```
procedure show
...
    -- Perform a security check
    if (not is_runnable(
         p_provider_id     =>  p_portlet_record.provider_id
        ,p_reference_path  =>  p_portlet_record.reference_path)
    ) then
        wwerr_api_error.add(
                  DOMAIN, SUBDOMAIN,
                  'securityerr', 'services_portlet.show');
        raise wwpro_api_provider.PORTLET_SECURITY_EXCEPTION;
    end if;
...
end show;
```

6. Find the `is_runnable` function. `is_runnable` is the place where you implement your security checks. In this example, the security check is quite simple. If the user is logged on (that is, not in a public session), then the function returns `true` and the portlet is displayed to the user. For your own purposes, you could, of course, code much more complex security checks in the `is_runnable` function.

```
function is_runnable
(
     p_provider_id in integer
    ,p_reference_path in varchar2
)
return boolean
is
begin
    --
    -- Portlet security check.  It allows the portlet to be visible
    -- if the user is logged on (that is, the current session is not a
    -- public session).
    --
    return wwctx_api.is_logged_on;
end is_runnable;
```

7. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 6.3.2, "Implementing the Provider Package" for information on how to add it.
8. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in Section 7.6.2, "Adding Portlets," of the *Oracle Application Server Portal User's Guide*.

6.8 Improving Portlet Performance with Caching

OracleAS Portal provides for the caching of PL/SQL portlets. This functionality permits PL/SQL portlets to cache their Web content on the middle tier. Subsequent requests for the content may be retrieved from the cache, with or without validation from the database, decreasing the database workload.

OracleAS Portal provides three types of caching for your PL/SQL portlets:

- **Validation-based caching** compares a key value to check whether the contents of the cache are still valid. If the key value does not change, it uses the cached content. Otherwise, it makes a round trip to the portal node to fetch the portlet content.
- **Expiry-based caching** uses a given expiration period for the contents of the cache when rendering the portlet. This form of caching is useful for content that changes infrequently or at very regular intervals (for example, every day at the close of business).
- **Invalidation-based caching** is the most complex form of caching but also the most flexible. The objects in OracleAS Web Cache are considered valid as long as they are not invalidated explicitly. You can also combine invalidation-based caching with either expiry-based or validation-based caching.

Because OracleAS Portal supports user customization of pages and portlets, the view of a page can vary from one user to another. OracleAS Portal's caching is designed to allow content to vary on a per-user basis, even if the URL is the same across all users. Therefore, portal objects can be cached at either the user level or the system level:

- **User-level caching** is for a specific user. The cache entries are unique for that user and cannot be accessed by other users.
- **System-level caching** is for all users. One cache entry is used for all users. Examples of content that might be suitable for system-level caching are page banners and news portlets.

When a database provider issues a request for a portlet, the request is sent to the portlets's `show` procedure. This procedure accepts the `portlet_runtime_record` as a parameter. This record structure contains fields that can be examined and set by the portlet to enable caching. The caching control fields of this record are:

- **`caching_key`**: This value is communicated in the `ETAG` header for this request and returned back to the portlet provider in subsequent requests. Setting this field enables validation-based caching.
- **`caching_period`**: This field enables expiry-based caching. The value is the number of minutes the content should be held in the cache. This mode overrides validation-based caching. If a value is set for this field, then the `caching_key` field is ignored.
- **`caching_level`**: This field defines whether the content is meant for general use or for a specific user. The valid values are `SYSTEM` and `USER`.

6.8.1 Using Caching

The general model for working with portlet caching varies according to the type of caching you choose. To a great extent, the type of caching you choose depends on the portlet content. If the portlet content changes at fairly regular intervals (for example, at the close of business every day), then it probably makes sense to use expiry-based caching. If the portlet content changes at irregular intervals, then validation- or invalidation-based caching is probably best.

6.8.1.1 Validation-Based Caching

If you choose validation-based caching, the general model is as follows:

1. Set the `caching_key` field of the `portlet_runtime_record` parameter. Add a check to compare the value of the current key with the value of the `caching_key` field of the `portlet_runtime_record` parameter. Note that the first time the `show` procedure is called, the key is null and its value must be set.
2. Determine whether you want to use system or user level caching. Set the `caching_level` field of the `portlet_runtime_record` parameter accordingly.

6.8.1.2 Expiry-Based Caching

If you choose expiry-based caching, the general model is as follows:

1. Set the `caching_period` field of the `portlet_runtime_record` parameter to the desired interval for the cache (in minutes).
2. Determine whether you want to use system or user level caching. Set the `caching_level` field of the `portlet_runtime_record` parameter accordingly.

6.8.1.3 Invalidation-Based Caching

If you choose invalidation-based caching, the general model is as follows:

1. Indicate to OracleAS Portal that it must generate specific headers for OracleAS Web Cache by calling `wwpro_api_provider.USE_INVALIDATION`.
2. Determine whether you want to use system or user level caching. Set the `caching_level` field of the `portlet_runtime_record` parameter accordingly.
3. Optionally, set up validation- or expiry-based caching as well.
4. Add invalidation logic to your portlet where needed (for example, when the portlet is customized) and make appropriate calls to `wwpro_api_invalidation`.

6.8.2 Configuring and Monitoring the Cache

The *Oracle Application Server Portal Configuration Guide* describes how to configure caching as well as how to monitor and tune performance.

6.8.3 Implementing Validation-Based Caching

The caching example, located in `..\pdkplsql\pdk\plsql\cache` in PDK-PL/SQL (`pdkplsql.zip`), illustrates how you can implement validation and expiry-based caching. You can browse through this example as follows to see how the validation-based functions are implemented in a portlet:

1. Open the `validcache_portlet.pkb` file in an editor.
2. At the very top of the file, notice the aliases for the caching level constants.

```
CREATE OR REPLACE
package body VALIDCACHE_PORTLET
is
    -- Caching Constants
    CACHE_LEVEL_SYSTEM constant varchar2(10) := 'SYSTEM';
    CACHE_LEVEL_USER   constant varchar2(10) := 'USER';
```

3. Find the show procedure. Notice first that the p_portlet_record is an in and out parameter for this procedure.

```
procedure show
(
    p_portlet_record   in out     wwpro_api_provider.portlet_runtime_record
)
```

4. In the procedure's security check, the caching fields of p_portlet_record are set to null if the security check fails.

```
begin
    if (not is_runnable(
         p_provider_id      => p_portlet_record.provider_id
        ,p_reference_path   => p_portlet_record.reference_path)
    ) then
        -- Set it to null so that cache does not get used even if exists
        p_portlet_record.caching_level := null;
        p_portlet_record.caching_key := null;
        raise wwpro_api_provider.PORTLET_SECURITY_EXCEPTION;
    end if;
```

5. After that, the procedure calls the get_cache_key function to get the cache key's value and assign it to a temporary value:

```
    --
    -- CACHE IS VALID?
    --
    l_cache_key := get_cache_key();
```

6. Find the get_cache_key function, which is referenced from the show procedure. This function generates a key for the portlet. You can implement your own logic here based upon your portlet's requirements.

```
function get_cache_key
return varchar2
is
    l_date date;
begin
    select sysdate into l_date from  dual;
    return trim(substr(to_char(l_date, 'YYYY:MM:DD:HH:MI:SS'),1,18));
exception
    when others then
        null;
end get_cache_key;
```

7. Now return to the show procedure. Notice how the code checks your portlet_runtime_record parameter for the current values of the caching_key and the caching_level. This same piece of code can compare your caching_key values.

```
    if p_portlet_record.caching_level = CACHE_LEVEL_SYSTEM then
        if l_cache_key is not null then
            -- Cache exists for the user, overwrite it
            p_portlet_record.caching_level := CACHE_LEVEL_USER;
            p_portlet_record.caching_key := l_cache_key;
        else
            return; -- System cache is still valid.
        end if;
    elsif p_portlet_record.caching_level = CACHE_LEVEL_USER then
```

```
            if p_portlet_record.caching_key != l_cache_key then
                -- cache has expired. reset it
                p_portlet_record.caching_key := l_cache_key;
            else
                return; -- User cache is good as gold
            end if;
        elsif p_portlet_record.caching_level is null then
            if p_portlet_record.caching_key is not null then
                -- Cache does not exists for the user, create it
                p_portlet_record.caching_level := CACHE_LEVEL_USER;
                p_portlet_record.caching_key := l_cache_key;
            else
                -- Define a sytem cache. This can happen only once!
                -- the first time the portlet is rendered.
                p_portlet_record.caching_level := CACHE_LEVEL_SYSTEM;
                p_portlet_record.caching_key := 'MY_INITIAL_CACHE_KEY';
            end if;
        else
            p_portlet_record.caching_level := CACHE_LEVEL_SYSTEM;
            p_portlet_record.caching_key := 'MY_INITIAL_CACHE_KEY';
        end if;
```

8. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 6.3.2, "Implementing the Provider Package" for information on how to add it.
9. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in Section 7.6.2, "Adding Portlets," of the *Oracle Application Server Portal User's Guide*.

6.8.4 Implementing Expiry-Based Caching

The caching example, located in `..\pdkplsql\pdk\plsql\cache` in PDK-PL/SQL (`pdkplsql.zip`), illustrates how you can implement expiry-based caching. You can browse through this example as follows to see how the expiry-based functions are implemented in a portlet:

1. Open the `expirycache_portlet.pkb` file in an editor.
2. At the very top of the file, notice the aliases for the caching level constants.

```
CREATE OR REPLACE
package body VALIDCACHE_PORTLET
is
    -- Caching Constants
    CACHE_LEVEL_SYSTEM constant varchar2(10) := 'SYSTEM';
    CACHE_LEVEL_USER   constant varchar2(10) := 'USER';
```

3. Find the `show` procedure. Notice first that the `p_portlet_record` is an `in` and `out` parameter for this procedure.

```
procedure show
(
    p_portlet_record   in out     wwpro_api_provider.portlet_runtime_record
)
```

4. In the procedure's security check, the caching fields of `p_portlet_record` are set to null if the security check fails.

```
begin
    if (not is_runnable(
         p_provider_id      => p_portlet_record.provider_id
        ,p_reference_path   => p_portlet_record.reference_path)
    ) then
        -- Set it to null so that cache does not get used even if exists
        p_portlet_record.caching_level := null;
        p_portlet_record.caching_key := null;
        raise wwpro_api_provider.PORTLET_SECURITY_EXCEPTION;
    end if;
```

5. After that, the procedure sets the value of the caching period in minutes in a temporary variable. The `get_cache_key` function to get the cache key's value and assign it to a temporary value:

   ```
   -- Set the Caching Period to one minute
   l_cache_period := 1;
   ```

6. Next, notice how the code checks `your portlet_runtime_record` parameter for the current values of the `caching_period` and sets the `caching_period` accordingly. This same piece of code can compare your `caching_period` values.

   ```
   if p_portlet_record.caching_level = CACHE_LEVEL_SYSTEM then
            -- Cache does not exists for the user, create it
            p_portlet_record.caching_level := CACHE_LEVEL_USER;
            p_portlet_record.caching_period := l_cache_period;
   elsif p_portlet_record.caching_level = CACHE_LEVEL_USER then
            -- Cache exists for the user, overwrite it
            p_portlet_record.caching_period := l_cache_period;
   elsif p_portlet_record.caching_level is null then
        if p_portlet_record.caching_period  is not null then
            -- Cache does not exists for the user, create it
            p_portlet_record.caching_level := CACHE_LEVEL_USER;
            p_portlet_record.caching_period := l_cache_period;
        else
            -- Define a sytem cache. This can happen only once!
            p_portlet_record.caching_level := CACHE_LEVEL_SYSTEM;
            p_portlet_record.caching_period := l_cache_period;
        end if;
   else -- p_portlet_record.caching_level value is messed up!
        p_portlet_record.caching_level := CACHE_LEVEL_SYSTEM;
        p_portlet_record.caching_period := l_cache_period;
   end if;
   ```

7. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 6.3.2, "Implementing the Provider Package" for information on how to add it.
8. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in Section 7.6.2, "Adding Portlets," of the *Oracle Application Server Portal User's Guide*.

6.8.5 Implementing Invalidation-Based Caching

Suppose you have a portlet that displays a map of the world, `map_portlet.pkb` and `map_portlet.pks`. You would go about adding invalidation-based functions to it as follows:

1. In the `show` procedure, you need to add a call to `wwpro_api_provider.use_invalidation`. This call indicates to OracleAS Portal that the portlet content should be cached by OracleAS Web Cache. Note that we have also specified that

the content be cached at the user level and that expiry-based caching be used as well (that is, an expiration interval of one minute has been set).

```
procedure show
...
    if (p_portlet_record.exec_mode = wwpro_api_provider.MODE_SHOW) then
        p_portlet_record.caching_invalidation :=
          wwpro_api_provider.use_invalidation;
        p_portlet_record.caching_level := 'USER';
        p_portlet_record.caching_period := 1;
...
```

2. Create a procedure in your `map_portlet.pkb` file that invalidates the cache. For example:

```
procedure map_invalidation
(
p_provider_id in number,
p_portlet_id in number,
p_instance_id in varchar2,
p_page_url in varchar2
)
is
begin
 wwpro_api_invalidation.invalidate_by_instance
   (p_provider_id => p_provider_id,
    p_portlet_id =>  p_portlet_id,
    p_instance_id => p_instance_id,
    p_user => wwctx_api.get_user);
 owa_util.redirect_url(p_page_url);
end map_invalidation;
```

3. In the `show` procedure, add a link for refreshing the portlet before the code that draws the map. For example:

```
/* Draw the Refresh Me link */
htp.anchor(
  curl => wwctx_api.get_user||
    '.map_invalidation?p_provider_id='||p_portlet_record.provider_id||
    '&p_portlet_id='||p_portlet_record.portlet_id||
    '&p_instance_id='||p_portlet_record.reference_path||
    '&p_page_url='||utl_url.escape(
                    url => p_portlet_record.page_url,
                    escape_reserved_chars => TRUE),
  ctext => wwui_api_portlet.portlet_text(
    p_string =>'Refresh Me',
    p_level => 1)
);
```

4. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 6.3.2, "Implementing the Provider Package" for information on how to add it.

5. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in Section 7.6.2, "Adding Portlets," of the *Oracle Application Server Portal User's Guide*.

6.9 Implementing Error Handling

OracleAS Portal provides the capability for you to trap erroneous input and return meaningful error messages. It manages the internal error stack by tracking the raised exceptions and retaining information about them. OracleAS Portal also includes a set of APIs for presenting errors in a standardized way.

Error handling services are available through the `wwerr_api_error` and `wwerr_api_error_ui` packages. These error handling services include the following key features:

- **Error stack**. OracleAS Portal uses an error stack to keep track of the error messages. When an error occurs, the error message is pushed onto an error stack. Whenever procedures or function calls are nested, the error stack keeps track of all the error messages. You can choose to retrieve only the top error (or most recent error) by using the `wwerr_api_error.get_top` method. Alternatively, you can get all the error messages on the stack using the `wwerr_api_error.get_errors` function. The stack can also be checked for the presence of any errors by calling the `wwerr_api_error.is_empty` function.
- **Error messages**. Error handling services provide a way to define meaningful error messages. To define your own error messages, you need to define its name space. The name space consists of the following:
  - **Name** is the error name.
  - **Domain** is the area of the product where the error occurred.
  - **Subdomain** is the subsystem where the error occurred.
  - **Context** is the name of the function where the error occurred.

  The name space uniquely identifies your error message. If it does not do so, a `wwc-0000` error message is generated.

  The default domains include the portal (`WWC`), application (`WWV`), and page groups (`WWS`). Each domain is further classified into subdomains, which define the object types. The portal domain includes the portlet, page, and document object types. The application domain includes object types such as forms, menus, reports, and charts. The page group domain includes object types such as folders, items, categories, and perspectives. If you need to define an error that does not fall within these classifications, you can define your own domain with subdomains for your errors.
- **Message parameters**. The other language strings that you create for your errors can take substitution parameters for your messages. The `p1`, `p2`, `p3`... parameters can be used to pass substitution parameters to the error messages. For example, for this string:

  ```
  (domain='yahoo', subdomain='provider', name='generalerror', string='Error: %1')
  ```

  an error can be added as follows:

  ```
  wwerr_api_error.add(p_domain=>'yahoo',  p_sub_domain=>'provider',
    p_name=>'generalerror',  p_context=>'yahoo.show',  p1=> sqlerrm);
  ```
- **Error display**. The `wwerr_api_error_ui` package provides a means to generate a standard user interface for displaying the errors in OracleAS Portal. The error messages can be displayed in two different ways:

- **Full screen user interface:** These error messages are displayed in a full screen mode. You may want to display full screen errors when the system encounters fatal or show-stopper errors.
- **Inline user interface:** These error messages are displayed within the current page itself. You may use inline errors for minor errors or warnings.

Additionally, you can choose the output format of the display (HTML, XML, or ASCII text).

6.9.1 Using Error Handling

In general, you set up error handling as follows:

1. On detecting error conditions, add the error message, with an appropriate domain and sub-domain combination, to the stack using the `wwerr_api_error.add` procedure.
2. When necessary (for example, at the end of a routine), expose the error messages using the `wwerr_api_error_ui` procedures. To display full screen messages, use the procedures `show_html`, `show_xml`, or `show_text` depending on your preferred output type. To display inline messages, use the procedures `show_inline_html`, `show_inline_xml`, or `show_inline_text`, depending on the output type you desire.

6.9.1.1 Guidelines for Error Handling

While implementing error handling, keep in mind the following:

- While defining your own error messages, use your own error domain for these messages. Never use the `WWC`, `WWV`, or `WWS` domain for your error messages. You will need to write a small loader script to load these into the other language tables.
- Avoid unnecessary error messages. If you do not want to do anything in a function, just return `null` rather than an error. For example, suppose you are coding a `copy_portlet` procedure for your portlet because the provider calls it for all of its other portlets. If you do not wish the `copy_portlet` procedure for this particular portlet to do anything, then simply have it return `null`. If you return errors, it will unnecessarily disrupt the portlet functionality.
- A maximum of ten error messages is kept on the stack. Beyond ten, messages are ignored when a call to `wwerr_api_error.add` is made.
- Use the API as a programmatic way of finding the problem. You can use the non-user-interface format for this purpose. For example, when programmatically registering a provider, the exception block can use `get_text_stack` to get the error messages and print them. This approach helps when debugging calls to public APIs since all of them add errors to the stack for exceptions.
- Remember to seed the other language strings for your error messages. For more information, refer to Section 6.11, "Writing Multi-Lingual Portlets".
- The standard user interface for error messages provides a navigation link back to the previous page. It also includes a Help icon for the specified help URL.

6.9.2 Adding Error Handling

The services example, located in `..\pdkplsql\pdk\plsql\svcex` in PDK-PL/SQL (`pdkplsql.zip`), illustrates how you can implement error handling. You can browse through this example as follows to see how the error handling functions are implemented in a portlet:

1. Open the `services_portlet.pkb` file in an editor.
2. The domain and subdomain definitions for your error messages are provided with aliases in the constants part of your portlet definition.

```
DOMAIN           constant varchar2(30) := 'provider';
SUBDOMAIN        constant varchar2(32) := 'services';
PORTLET_PATH     constant varchar2(256):= 'oracle.portal.pdk.servicesportlet';
PREFNAME_STRING  constant varchar2(30) := 'services_string';
PREFNAME_TITLE   constant varchar2(30) := 'services_title';
```

3. Find the `show` procedure. This procedure performs a security check and, if an error condition arises, it calls `wwerr_api_error.add` to push the `securityerr` error message onto the stack.

```
procedure show
(
    p_portlet_record        wwpro_api_provider.portlet_runtime_record
)
is
...
begin
    -- Perform a security check
    if (not is_runnable(
         p_provider_id     =>  p_portlet_record.provider_id
        ,p_reference_path  =>  p_portlet_record.reference_path)
    ) then
        wwerr_api_error.add(
                  DOMAIN, SUBDOMAIN,
                  'securityerr', 'services_portlet.show');
        raise wwpro_api_provider.PORTLET_SECURITY_EXCEPTION;
    end if;
```

4. The `show` procedure also checks for any other kind of execution mode and generates an appropriate error message for an invalid display mode.

```
if (p_portlet_record.exec_mode = wwpro_api_provider.MODE_SHOW) then
...
elsif (p_portlet_record.exec_mode = wwpro_api_provider.MODE_SHOW_EDIT)
...
else
    wwerr_api_error.add(DOMAIN, SUBDOMAIN,
        'invaliddispmode', 'services_portlet.show');
    raise wwpro_api_provider.PORTLET_EXECUTION_EXCEPTION;
end if;
```

5. Lastly, the `show` procedure implements a general error message in the exception handler to catch any errors not trapped by the preceding conditions.

```
exception
    when others then
        wwerr_api_error.add(
            DOMAIN, SUBDOMAIN,
            'generalerr', 'services_portlet.show');
        raise wwpro_api_provider.PORTLET_EXECUTION_EXCEPTION;
end show;
```

6. Error handling is also implemented in the `save_prefs` and `save_default_prefs` procedures. They check whether the error stack is empty and, if it is not, the portlet makes a call to `wwerr_api_error.show_html` to display the error in full screen mode.

```
exception
    when INVALID_TEXT_EXCEPTION then
         l_information := l_user||'%'||l_time
             ||'%INVALID_TEXT_EXCEPTION%'||p_string;
         l_action      := LOG_FAILED;
         wwlog_api.log (p_domain      => DOMAIN,
                        p_subdomain   => SUBDOMAIN,
                        p_name        => l_user,
                        p_action      => l_action,
                        p_information => l_information,
                        p_url         => l_url,
                        p_row_count   => 0,
                        p_elapsed_time=> l_elapsed_time);
         wwerr_api_error.add(DOMAIN, SUBDOMAIN,
             'invalid_text', 'services_portlet.save_prefs');
     if (not wwerr_api_error.is_empty) then
         wwerr_api_error_ui.show_html;
     end if;
end save_prefs;
```

7. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 6.3.2, "Implementing the Provider Package" for information on how to add it.
8. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in Section 7.6.2, "Adding Portlets," of the *Oracle Application Server Portal User's Guide*.

6.10 Implementing Event Logging

OracleAS Portal can log events that occur during transactions with its objects. It stores these logs in the database, which makes them available through standard SQL calls and reporting tools.

You can choose the events you would like to log and organize them categorically based on user-defined domains and subdomains. For the logged events, you can view information about the event, the time the event started and stopped, the host or IP address of the remote user, the browser type, and the language.

Event logging services are available through the `wwlog_api` and `wwlog_api_admin` packages. These services include the following key features:

- Event logs are useful for tracking specific usage of the portal. To track such information, you create a log event. Log events require a name space that consists of:
  - **Name** is the event name.
  - **Domain** is the area of the product where the event occurred.
  - **Subdomain** is the subsystem that generated the event.

  The default domains include the portal (`WWC`), application (`WWV`), and page group (`WWS`). Each domain is further classified into subdomains which define the object types. The portal domain includes the portlet, page, and document object types. The application domain includes object types such as forms, menus, reports, and charts. The page group domain includes object types such as folders, items, categories, and perspectives. Events themselves could be of types such as add, delete, customize, hide, copy, execute, and export. If you need to define an event that does not fall within these classifications, you can define your own domain with subdomains for your events.

- Logs can track information in two different ways:
  - Interval logging calculates the elapsed time for the action performed (for example, the time taken to render a portlet).
  - Event logging logs the occurrence of a single step event you care about (for example, whenever a user customizes a portlet).
- Log switching enables you to set a switch interval that defines how long you want to maintain your existing log records. The log information stored in the database uses two different tables. The log records are purged based on the value entered for the **Activity Log Interval** in the **Configuration** tab of **Global Settings** (accessible from the Services portlet in the **Administer > Portal** tab). When the log interval (in days) is reached, the logging switches between the two logging tables in the database (for example, A and B). Logs first go into A. When the log interval is reached the first time, the logs are written to B. When the log interval is reached again, the logs go back to A. A is emptied in preparation to store the new log records. If you set your log interval to 14 (the default setting), the logs will switch every 14 days, thus preserving for you, at any point in time, records dated between 14 and 28 days old.

6.10.1 Using Event Logging

In general, you can set up event logging as follows:

1. Add the event object, with an appropriate domain and subdomain combination, using `wwlog_api_admin.add_log_event`. Adding the event ensures that lists of values and other user interface components invoked when the user is monitoring the events show this new event in their lists.
2. Register the log event record by using `wwlog_api_admin.add_log_registry`. The log registry record represents the events you want to log in the future and provides a means to filter the events that need to be logged.
3. Use `start_log` and `stop_log` to mark the events you want to log in your code. Alternatively, for entering single step event log information, just call the log method to mark that event.

6.10.1.1 Guidelines for Event Logging

While implementing event logging, keep in mind the following:

- Log only what you really care about to improve performance. You don't want to flood the system with log messages about which you do not care. If events are logged in Show mode, then multiple instances of these portlets mean additional hits to the database.
- Choose your domain, subdomain, and log events carefully. While using the log APIs, do not use the OracleAS Portal domains like `WWC`, `WWV`, or `WWS` for your log messages. Organize your domains and subdomains hierarchically ensuring that they are unique across portlets. If other portlets happen to use the same domains or subdomains, you will see those log messages interspersed with your own.
- Create log events that show up in the pop-up lists of values monitoring the logs. You can simply create log registry records that filter the events that would be actually logged, either by specifying particular events or using the generic filters with wild cards (`%`). Apart from creating log registry records, we recommend that you create log events for events that you want to monitor. This way the lists of values in the user interface show these records for additional functions like monitoring.

- Provide required privileges to users or user groups who need to monitor the logs. Any logs created by a user can be viewed by that user, the Portal Administrator, and any user with the Edit privilege on the `ANY_LOGS` object type.

6.10.2 Adding Event Logging

The services example, located in `..\pdkplsql\pdk\plsql\svcex` in PDK-PL/SQL (`pdkplsql.zip`), illustrates how you can implement event logging. You can browse through this example as follows to see how the event logging functions are implemented in a portlet:

1. Open the `services_portlet.pkb` file in an editor.
2. The domain and subdomain definitions for your log messages are provided with aliases in the constants part of your portlet definition.

```
DOMAIN           constant varchar2(30) := 'provider';
SUBDOMAIN        constant varchar2(32) := 'services';
PORTLET_PATH     constant varchar2(256):= 'oracle.portal.pdk.servicesportlet';
PREFNAME_STRING  constant varchar2(30) := 'services_string';
PREFNAME_TITLE   constant varchar2(30) := 'services_title';
```

3. Find the `save_prefs` procedure. This procedure provides customizable functionality where you can personalize text and the portlet title in Edit mode. `save_prefs` stores these customizations in the database. While saving the changes, it is advisable to log them. Hence, this procedure provides an ideal example of implementing the logging service. A single step event is logged using `wwlog_api.log`. The first instance of wwlog_api.log logs the event of personalizing text. The second instance logs the event of personalizing the portlet title.

```
procedure save_prefs
...
begin
...
    if (l_prefs.string_id is null or to_number(l_prefs.string_id) = 0)
    then
        l_action := LOG_INSERT;
...
    else  -- string exists in at least one language so update it
        l_action := LOG_UPDATE;
...
    end if;
-- Log this transaction
l_information := l_user||'%'||l_time||'%completed%'||p_string;
        wwlog_api.log (p_domain      => DOMAIN,
                       p_subdomain   => SUBDOMAIN,
                       p_name        => l_user,
                       p_action      => l_action,
                       p_information => l_information,
                       p_url         => l_url,
                       p_row_count   => l_row_count,
                       p_elapsed_time=> l_elapsed_time);
...
    if (l_prefs.title_id is null or to_number(l_prefs.title_id) = 0)
    then
       l_action := LOG_INSERT;
...
    else
       l_action := LOG_UPDATE;
```

```
...
-- Log this transaction
       l_information := l_user||'%'||l_time||'%completed%'||p_title;
       wwlog_api.log (p_domain      => DOMAIN,
                      p_subdomain   => SUBDOMAIN,
                      p_name        => l_user,
                      p_action      => l_action,
                      p_information => l_information,
                      p_url         => l_url,
                      p_row_count   => l_row_count,
                      p_elapsed_time=> l_elapsed_time);
...
end save_prefs;
```

4. The `save_prefs` procedure also logs an event with `wwlog_api.log` when an exception occurs.

```
exception
    when INVALID_TEXT_EXCEPTION then
         l_information := l_user||'%'||l_time
             ||'%INVALID_TEXT_EXCEPTION%'||p_string;
         l_action      := LOG_FAILED;
         wwlog_api.log (p_domain      => DOMAIN,
                        p_subdomain   => SUBDOMAIN,
                        p_name        => l_user,
                        p_action      => l_action,
                        p_information => l_information,
                        p_url         => l_url,
                        p_row_count   => 0,
                        p_elapsed_time=> l_elapsed_time);
...
```

5. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 6.3.2, "Implementing the Provider Package" for information on how to add it.
6. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in Section 7.6.2, "Adding Portlets," of the *Oracle Application Server Portal User's Guide*.

6.11 Writing Multi-Lingual Portlets

OracleAS Portal has a robust set of APIs for interacting with OracleAS Portal multi-lingual storage facility. This storage facility provides a mechanism for the storing and retrieving of strings in different languages. These APIs abstract the native multi-lingual functionality and provide developers with a powerful storage mechanism for developing providers that support different language environments.

Multi-lingual services are available through the `wwnls_api` package. These services include the following key features:

- The multi-lingual APIs enable the provider to load several translations for the strings displayed in their portlets. Once the strings have been loaded, the provider can call the APIs to retrieve the strings from the multi-lingual table as needed.
- Context APIs retrieve the user's language and the appropriate translation for that language. The Context APIs determine the user's language environment from the language setting in the browser. When a requested translation does not exist, the APIs return the base language translation.

For example, assume that the provider's `register` procedure loads US and French translations for the portlet title. When the portlet is rendered, the provider implementation retrieves the portlet title string from the table and displays the following results:

- A request for a French string causes the portlet title to appear in French.
- A request for a US string causes the portlet title to appear in US English.
- A request for a Chinese string causes the portlet title to appear in US English because we did not load a translation for the Chinese language.

6.11.1 Using Multi-Lingual Support

In general, you can set up multi-lingual support as follows:

1. Load your string definitions into the database using the string equivalents for each language you intend to use. For this purpose, call the `wwnls_api.add_string` or `wwnls_api.set_string` with an appropriate domain, subdomain, error message name, and error text combination.
2. Retrieve the strings you require with `wwnls_api.get_string` for the language that you desire.

6.11.2 Adding Multi-Lingual Support

To add multi-lingual support, you need to perform the following tasks:

- Loading Language Strings
- Retrieving Language Strings

6.11.2.1 Loading Language Strings

Language strings can be loaded by a script that is part of the provider installation. This script calls `add_string` and `set_string` to create equivalent strings for different languages.

OracleAS Portal uniquely identifies language strings using a combination of domain, subdomain, and name. The domain and subdomain provide a way to categorize the strings. The domain and subdomain should be unique enough to reasonably preclude conflicts with other users of the APIs.

- A *domain* is a particular area of the product. An example of a domain could be provider or page group.
- A *subdomain* is a subsystem of the domain. For example, the subdomain could be the provider name (for example, `HelloProvider`) or subpage name (for example, `HelloPage`).

The services example, located in `..\pdkplsql\pdk\plsql\svcex` in PDK-PL/SQL (`pdkplsql.zip`), illustrates how you can implement multi-lingual support. You can browse through this example as follows to see how to load strings for multi-lingual support:

1. Open the `services_seed.sql` file in an editor.
2. Notice the `add_string` call with the parameters for domain name, subdomain name, string name, language, and the actual string text. It returns the String ID for the language string. For setting equivalent strings in other languages, `set_string` is called with the same parameters.

```
set serveroutput on size 1000000
set define off

declare
    l_string_id  integer;
    l_person_id  integer;
    l_group_id   integer;
begin
...
-- strings for portlet record fields
l_string_id := wwnls_api.add_string(
 'provider','services','ptldefname','us','DatabaseServicesPortlet');
wwnls_api.set_string(
 'provider','services','ptldefname','d','DatenbankServicesPortlet-d');
l_string_id := wwnls_api.add_string(
 'provider','services','ptldeftitle','us','Database Services Portlet');
wwnls_api.set_string(
 'provider','services','ptldeftitle','d','Datenbank Services Portlet - d');
l_string_id := wwnls_api.add_string(
 'provider','services','ptldefdesc','us','This is the database services
portlet implemented in PL/SQL. It displays 6 show modes.');
wwnls_api.set_string(
 'provider','services','ptldefdesc','d','Dies ist das Datenbank Service
Portlet, erstellt in PL/SQL. Es stellt 6 Anzeigemodi dar. - d');
l_string_id := wwnls_api.add_string(
 'provider','services','ptldevtmmsg','us','Web Services Portlet Timed
Out.');
wwnls_api.set_string(
 'provider','services','ptldevtmmsg','d','Zeitüeberschreitung aufgetreten
in Web Services Portlet. -d');
```

6.11.2.2 Retrieving Language Strings

The services example, located in `..\pdkplsql\pdk\plsql\svcex` in PDK-PL/SQL (`pdkplsql.zip`), illustrates how you can implement multi-lingual support. You can browse through this example as follows to see how to retrieve strings for multi-lingual support:

1. Open the `services_portlet.pkb` file in an editor.
2. The domain and subdomain definitions for your language strings are provided with aliases in the constants part of your portlet definition.

```
DOMAIN           constant varchar2(30) := 'provider';
SUBDOMAIN        constant varchar2(32) := 'services';
PORTLET_PATH     constant varchar2(256):= 'oracle.portal.pdk.servicesportlet';
PREFNAME_STRING  constant varchar2(30) := 'services_string';
PREFNAME_TITLE   constant varchar2(30) := 'services_title';
```

3. Find the `get_portlet_info` procedure. Notice the calls to `wwnls_api.get_string` to populate the portlet title, name, and description.

```
function get_portlet_info
(
     p_provider_id in integer
    ,p_language in varchar2
)
return wwpro_api_provider.portlet_record
is
     l_portlet  wwpro_api_provider.portlet_record;
begin
    l_portlet.id := services_provider.SERVICES_PORTLET_ID;
```

```
        l_portlet.provider_id := p_provider_id;
        l_portlet.language := p_language;
        l_portlet.title :=
            wwnls_api.get_string(
                 p_domain       =>  DOMAIN
                ,p_sub_domain   =>  SUBDOMAIN
                ,p_name         =>  'ptldeftitle'
                ,p_language     =>  p_language
                );
        l_portlet.description :=
            wwnls_api.get_string(
                 p_domain       =>  DOMAIN
                ,p_sub_domain   =>  SUBDOMAIN
                ,p_name         =>  'ptldefdesc'
                ,p_language     =>  p_language
                );
        l_portlet.name :=
            wwnls_api.get_string(
                 p_domain       =>  DOMAIN
                ,p_sub_domain   =>  SUBDOMAIN
                ,p_name         =>  'ptldefname'
                ,p_language     =>  p_language
                );
...
```

4. Browse the rest of the file to examine other usage examples of`wwnls_api.get_string`, which is used in several other places in `services_portlet.pkb`.
5. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 6.3.2, "Implementing the Provider Package" for information on how to add it.
6. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in Section 7.6.2, "Adding Portlets," of the *Oracle Application Server Portal User's Guide*.

Part IV

Appendixes

Part IV contains the following appendixes:

- Appendix A, "Building Portlets with the Portlet Builder"
- Appendix B, "Troubleshooting OracleAS Portal"

A

Building Portlets with the Portlet Builder

For developers who want quick solutions for building components, OracleAS Portal provides the Portlet Builder. The Portlet Builder wizards guide you step-by-step through the process of creating a portlet. The wizards enable even the novice business developer to begin creating portlets right away.

After answering a few basic questions, you can decide whether to continue with the wizard or to create the portlet based on information already collected. A Finish button appears as soon as the wizard has collected enough information, allowing you to create the portlet without having to traverse all the wizard's pages. When you're ready for other users to run the portlet, you need only assign portlet access privileges, publish the portlet, and add it to a page.

This appendix describes the process of creating a portlet through a wizard and steps you through creating, editing, managing, and running different types of portlets. It contains the following sections:

- Using a Wizard to Build a Portlet
- Editing a Portlet Builder Component
- Managing Portlets
- Managing Versions
- Managing Portlet Security
- Performing Test Runs on a Portlet
- Referencing the OracleAS Portal Schema
- Coding Additional Functionality
- Using Shared Components to Create a Look and Feel
- Example: Building Charts and Reports

A.1 Using a Wizard to Build a Portlet

Portlets must be built under the ownership of a provider. This means that before you can build a portlet, you must first create a provider—or select an existing provider—to host (own or contain) the portlet. If you're creating a new provider, you may first want to create a schema against which to build the provider.

Although every portlet you create using OracleAS Portal requires that you build it under the ownership of a provider, it is not necessary that you build a provider each time you build a portlet. In fact, you can use an existing provider, or create one dedicated to the ownership of your locally built portlets, and then never have to create another provider again.

One provider can be used as a container that manages multiple portlets. A good rule is to use one provider per logical application, for example, a Human Resources application that consists of reports, forms, and charts.

This section steps you through the process of creating a schema, creating a provider, and building a portlet under the ownership of that provider in OracleAS Portal. It includes the following subsections:

- Creating a Schema in OracleAS Portal
- Creating a Provider for Locally Built Portlets
- Creating Portlets Using OracleAS Portal Wizards

A.1.1 Creating a Schema in OracleAS Portal

A schema is an Oracle database user account under the ownership of which you can store database objects, applications, and components and control the schema's database privileges. Use a Portal schema to enforce password access to the schema, to set aside tablespace specifically for your locally built portlets, and to identify the temporary tablespace the schema should use.

Creating a schema involves three main tasks:

- Creating a Schema
- Granting and Revoking Privileges on Database Objects
- Enrolling the Schema in One or More Roles

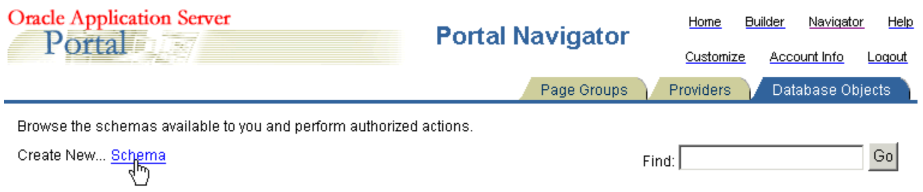
This section steps you through each of these tasks.

A.1.1.1 Creating a Schema

To create a schema in OracleAS Portal:

1. Log in to OracleAS Portal.
2. Click the **Navigator** link at the top of the page.
3. In the Navigator, click the **Database Objects** tab to bring it forward.
4. At the top of the page, click the **Schema** link next to **Create New...** (Figure A–1)

Figure A–1 The Schema Link on the Database Objects Tab in the Portal Navigator

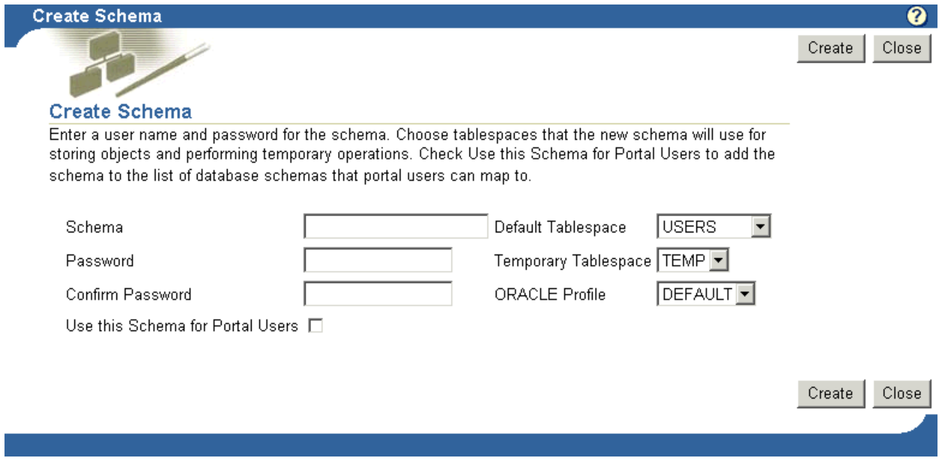


Browse the schemas available to you and perform authorized actions.

5. On the Create Schema page (Figure A–2), enter a name to identify the schema in the **Schema** field.

   The name must be unique within the portal. Blank and special characters are not allowed in the name. Type an underscore character to add a space in a name. For example, you can name a schema `PORTLETS_SCHEMA`, but not `PORTLETS SCHEMA`. Additionally, you cannot name a schema `PORTLETS*SCHEMA` nor `PORTLETS%SCHEMA`.

Figure A–2 The Create Schema Page



Create Schema

Enter a user name and password for the schema. Choose tablespaces that the new schema will use for storing objects and performing temporary operations. Check Use this Schema for Portal Users to add the schema to the list of database schemas that portal users can map to.

Schema		Default Tablespace	USERS
Password		Temporary Tablespace	TEMP
Confirm Password		ORACLE Profile	DEFAULT
Use this Schema for Portal Users ☐			

6. In the **Password** field, enter the password you will assign to the schema.

   Asterisks appear for each character you enter. Users will enter this password when logging into the Oracle database as this schema.

7. Reenter the password in the **Confirm Password** field.

8. Select a default tablespace from the **Default Tablespace** list.

   This will be used for storing any database objects created by the schema. We recommend that you select a tablespace created specifically for storing locally build portlet applications. If necessary, talk about this with your database administrator.

9. Select a temporary tablespace from the **Temporary Tablespace** list.

   This will be used by the schema to create temporary storage for operations such as sorting table rows.

10. Select an Oracle resource profile for the new schema from the **ORACLE Profile** list.

    The default profile is DEFAULT. Feel free to use this. If you plan to create a new profile, you must use Oracle SQL commands to do so. Refer to the Oracle database documentation for more information. You'll find this on the Oracle Technology Network at `http://www.oracle.com/technology/index.html`.

11. Select the **Use this Schema for Portal Users** check box to add the schema you are creating to the list of database schemas to which portal users can map for administrative purposes.

    Every OracleAS Portal user must be associated with a database schema. By default, the name of this schema is <*portal*>_public, where <*portal*> is the name of the schema in which OracleAS Portal is installed. If you select this check box, a portal user can be mapped instead to the schema you are creating here.

12. Click **Create** to save your changes and return to the Portal Navigator.

    Alternatively, click the link to your new schema that now appears at the top of the page to begin editing the new schema's properties. See "Granting and Revoking Privileges on Database Objects" and "Enrolling the Schema in One or More Roles" for more information.

A.1.1.2 Granting and Revoking Privileges on Database Objects

You may need to increase or limit the access your users will have on some database objects associated with this schema. Once they are able to log in to this schema, users' access privileges are influenced by the roles of which this schema is a member (see "Enrolling the Schema in One or More Roles"). You can use grants to select specific database objects and specify which of those access privileges should be further limited or increased.

For example, imagine that you have a BASIC_USER role. This role allows users to SELECT from all tables in the database. You may care to restrict selections on some tables. You can use grants to select one or more database objects (such as specific tables, views, and the like) and remove the SELECT privilege. Conversely, if a role does not include all the access privileges you want a schema to have on one or more database objects, you can use grants to increase access privileges.

Note: By default, OracleAS Portal allows only SELECT on database objects. Other privileges must be granted explicitly.

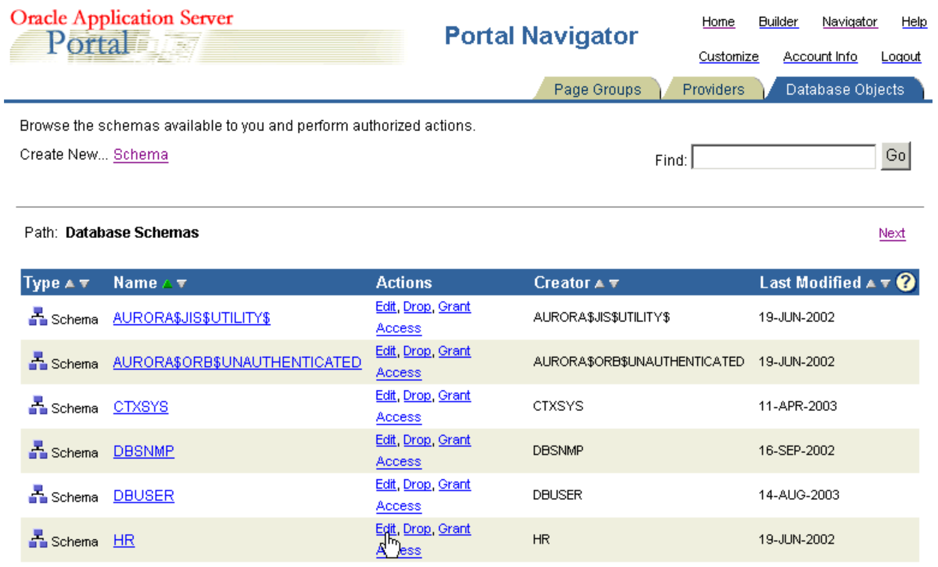
This section describes how to further control privileges on objects associated with your schema.

Note: If you have arrived at this task by clicking a link to edit the schema that appeared at the top of the Create Schema page, skip to step 6.

To grant privileges to database objects associated with a schema:

1. Log in to OracleAS Portal.
2. Click the **Navigator** link at the top of the page.
3. Click the **Database Objects** tab to bring it forward.
4. On the Database Objects tab, go to the **Name** column to locate the schema you will work with, and click the **Edit** link that is next to it.

Figure A–3 The Edit Link Next to a Schema in the Portal Navigator



Browse the schemas available to you and perform authorized actions.

Create New... Schema

Find: Go

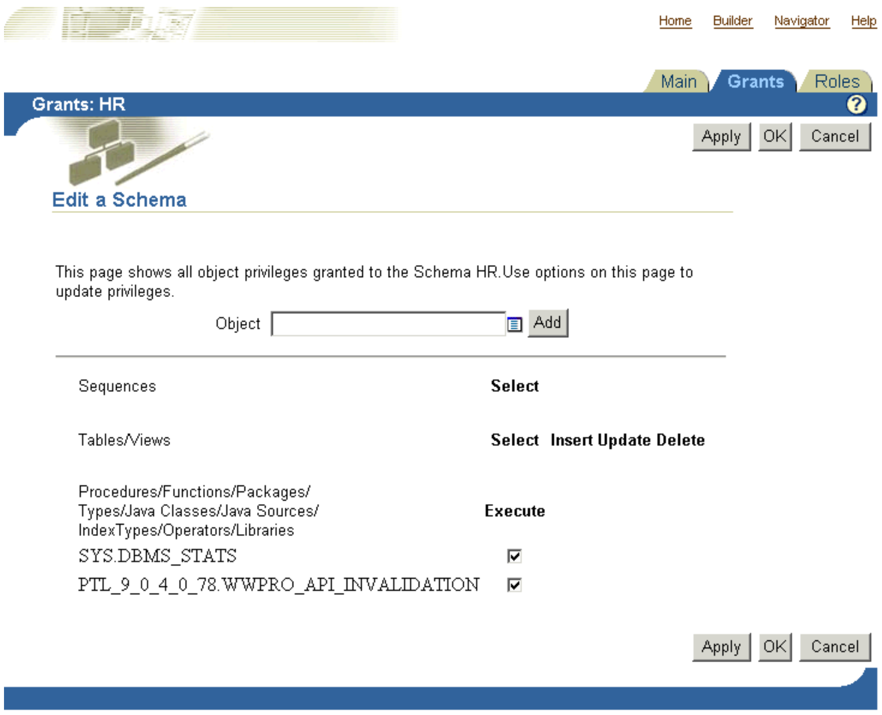
Path: Database Schemas

Next

Type	Name	Actions	Creator	Last Modified
Schema	AURORA$JIS$UTILITY$	Edit, Drop, Grant Access	AURORA$JIS$UTILITY$	19-JUN-2002
Schema	AURORA$ORB$UNAUTHENTICATED	Edit, Drop, Grant Access	AURORA$ORB$UNAUTHENTICATED	19-JUN-2002
Schema	CTXSYS	Edit, Drop, Grant Access	CTXSYS	11-APR-2003
Schema	DBSNMP	Edit, Drop, Grant Access	DBSNMP	16-SEP-2002
Schema	DBUSER	Edit, Drop, Grant Access	DBUSER	14-AUG-2003
Schema	HR	Edit, Drop, Grant Access	HR	19-JUN-2002

5. On the resulting page, click the **Grants** tab (Figure A–4).

Figure A–4 The Grants Tab



This page shows all object privileges granted to the Schema HR.Use options on this page to update privileges.

Object Add

Sequences	Select
Tables/Views	Select Insert Update Delete
Procedures/Functions/Packages/ Types/Java Classes/Java Sources/ IndexTypes/Operators/Libraries	Execute
SYS.DBMS_STATS	☑
PTL_9_0_4_0_78.WWPRO_API_INVALIDATION	☑

Apply OK Cancel

6. In the **Object** field, enter the name of a database object on which you want to control privileges.

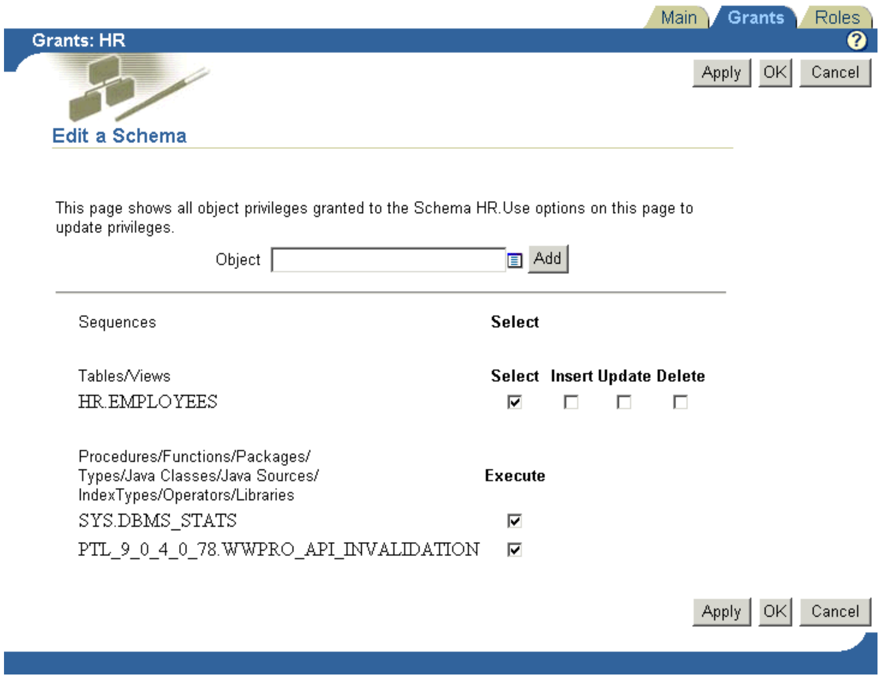
   Click the **Browse** icon to select from a list of objects. Prefix the name of the object with the schema that owns it. For example, add SCOTT to EMP (SCOTT.EMP) to indicate the EMP table in the SCOTT schema.

7. Click **Add** to add the object to the list at the bottom of the page.

Objects are grouped by object type. That is, tables are listed with tables, views with views, and so on.

8. Check or clear check boxes next to the object to increase or limit the access privileges on the object.

Figure A–5 The Check Boxes Next to a Table Object



This page shows all object privileges granted to the Schema HR.Use options on this page to update privileges.

Sequences	Select			
Tables/Views	Select	Insert	Update	Delete
HR.EMPLOYEES	✓	☐	☐	☐
Procedures/Functions/Packages/ Types/Java Classes/Java Sources/ IndexTypes/Operators/Libraries	Execute			
SYS.DBMS_STATS	✓			
PTL_9_0_4_0_78.WWPRO_API_INVALIDATION	✓			

9. Click **Apply** to save your changes, or click **OK** to save your changes and return to the Portal Navigator.

A.1.1.3 Enrolling the Schema in One or More Roles

You will want to enroll your schema in one or more roles to provide the desired level of overall database access privileges. If you think of a schema as a super-user, created from a population of users who have password access to the schema, you enroll a schema in one or more roles to provide access privileges to the database to all users who in turn have login access to the schema.

The access privileges provided through the schema's membership in one or more roles can be overridden on specific database objects through grants (see "Granting and Revoking Privileges on Database Objects").

Note: If you have arrived at this task by clicking a link to edit the schema that appeared at the top of the Create Schema page or if you are continuing from editing the schema's **Main** or **Grants** tab, skip to step 6.

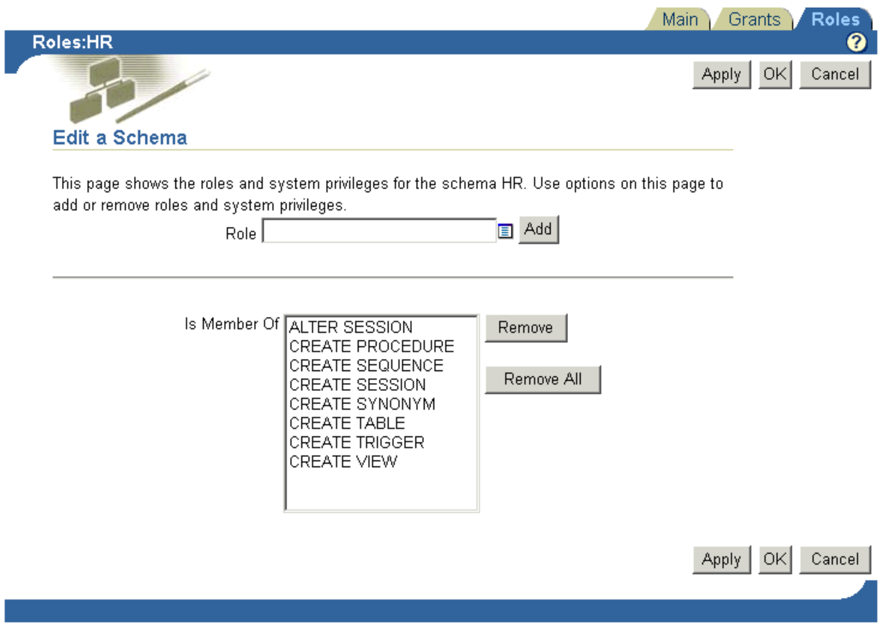
To enroll a schema in one or more roles:

1. Log in to OracleAS Portal.
2. Click the **Navigator** link at the top of the page.
3. Click the **Database Objects** tab to bring it forward.

4. On the Database Objects tab, go to the **Name** column to locate the schema you will work with and click the **Edit** link that is next to it.

5. On the resulting page, click the **Roles** tab to bring it forward.

Figure A–6 The Roles Tab



This page shows the roles and system privileges for the schema HR. Use options on this page to add or remove roles and system privileges.

Role [ ] Add

Is Member Of:
ALTER SESSION
CREATE PROCEDURE
CREATE SEQUENCE
CREATE SESSION
CREATE SYNONYM
CREATE TABLE
CREATE TRIGGER
CREATE VIEW

By default, the schema is enrolled in the CONNECT role, and may be enrolled in others.

6. Click the **Browse** icon next to the **Role** field.

7. Select a role to which you will enroll the schema.

8. Click the **Add** button next to the **Role** field to add the selected role to the **Is a Member Of** list.

   Continue selecting roles and clicking the **Add** button until you have enrolled the schema in all desired roles.

9. Click **Apply** to save your changes, or click **OK** to save your changes and return to the Portal Navigator.

A.1.2 Creating a Provider for Locally Built Portlets

Once you have created or identified an existing schema that will own your provider, you are ready to create a provider to host (own or contain) your locally built portlets.

To create a provider for locally build portlets:

1. Log in to OracleAS Portal.

2. Click the **Navigator** link at the top of the page.

3. In the Portal Navigator, click the **Providers** tab to bring it forward.

4. On the Providers tab, click the **Locally Built Providers** link.

5. On the resulting page, click the **Database Provider** link next to **Create New...**

Note: If you do not see the **Create New** link on the Locally Built Providers page, this means that the user name you used to log in does not have privileges in any schema. The capability to create a new provider is available only when the current user has some privilege in a database schema. In this case, you must have the Portal Administrator edit your user profile to grant you the privilege to create a schema. To do this:

1. Go to the **Portal User Profile** portlet on the **Administer** tab of the Portal Builder page.
2. Enter the user name and click **Edit**.
3. On the **Edit Portal User Profile** page, click the **Privileges** tab to bring it forward.
4. Under **Administration Privileges**, set **All Schemas** to **Create**.

Next, the Portal Administrator must add your user name to a group that has privileges to create database providers. To do this:

1. Go to the **Group** portlet on the **Administer** tab of the Portal Builder page.
2. Under **Edit/Delete Group**, enter the group name and click **Edit**.
3. On the resulting page, under **Members**, click **Add User**.
4. In the **Search** field, enter the user name, and click **Go**.
5. Select the newly added user name entry, then under **Roles Assignment**, select a group that has privileges to create database providers.

When these tasks are complete, you can log in again with your user name and create a schema on the **Database Objects** tab in the Portal Navigator. Once you have the privilege to create a schema, you will see the **Create New** link on the Locally Built Providers page.

---

6. In the **Portal DB Provider Name** field, enter an internal name to identify this provider to the portal.

   Blank characters and special characters are not allowed. Type an underscore character to add a space, for example, `portlet_provider`.
7. In the **Portal DB Provider Display Name** field, enter a display name to identify this provider to users.
8. Select a schema, for example, the one you just created, from the **Portal DB Provider Schema** drop-down list.

   This list of schemas includes only those with the **Use this Schema for Portal Users** check box checked. This check box appears on the **Main** tab.
9. Click **OK** to save your changes and return to the Portal Navigator.

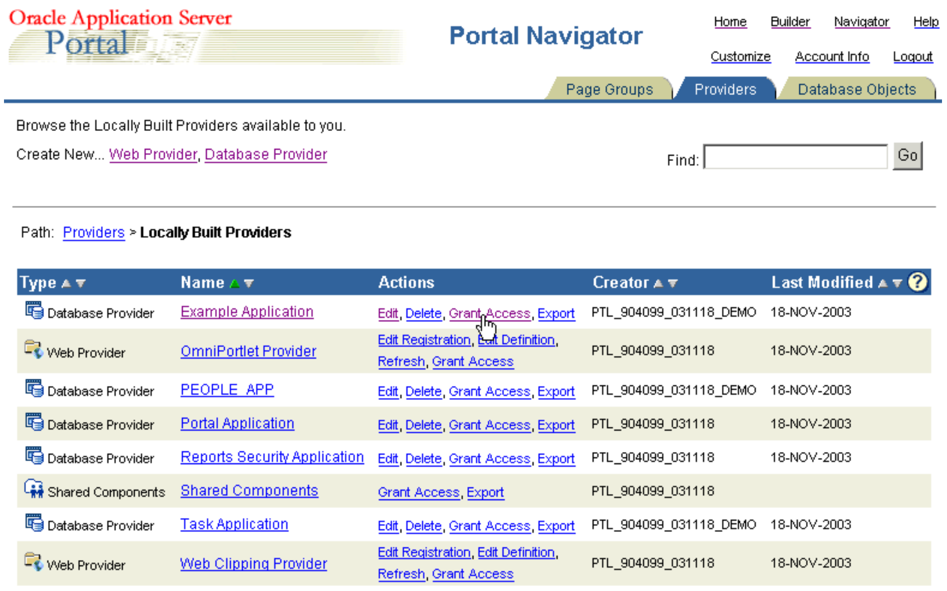
A.1.3 Exposing a Provider

Before you can create and publish portlets under the auspices of a provider, you must ensure that the provider has been identified as a provider to OracleAS Portal. To do this:

1. Create a provider, for example, as described in Section A.1.2.

2. Locate the listing for your new provider in the Portal Navigator.
3. Click the **Grant Access** link next to your new provider (Figure A–7).

Figure A–7 The Grant Access Link Next to a Provider in the Portal Navigator



Oracle Application Server Portal
Portal Navigator
Home Builder Navigator Help
Customize Account Info Logout
Page Groups Providers Database Objects
Find: Go

Browse the Locally Built Providers available to you.

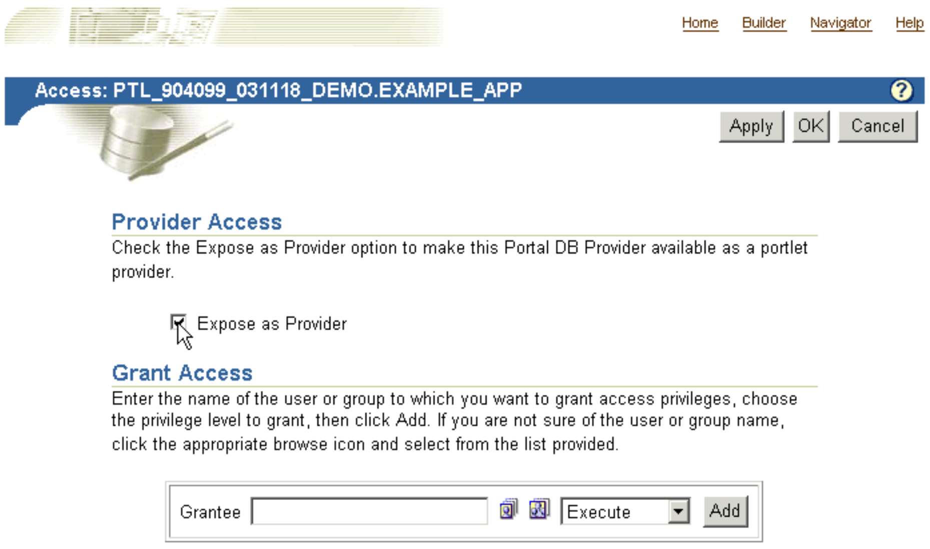
Create New... Web Provider, Database Provider

Path: Providers > **Locally Built Providers**

Type	Name	Actions	Creator	Last Modified
Database Provider	Example Application	Edit, Delete, Grant Access, Export	PTL_904099_031118_DEMO	18-NOV-2003
Web Provider	OmniPortlet Provider	Edit Registration, Edit Definition, Refresh, Grant Access	PTL_904099_031118	18-NOV-2003
Database Provider	PEOPLE_APP	Edit, Delete, Grant Access, Export	PTL_904099_031118_DEMO	18-NOV-2003
Database Provider	Portal Application	Edit, Delete, Grant Access, Export	PTL_904099_031118	18-NOV-2003
Database Provider	Reports Security Application	Edit, Delete, Grant Access, Export	PTL_904099_031118	18-NOV-2003
Shared Components	Shared Components	Grant Access, Export	PTL_904099_031118	
Database Provider	Task Application	Edit, Delete, Grant Access, Export	PTL_904099_031118_DEMO	18-NOV-2003
Web Provider	Web Clipping Provider	Edit Registration, Edit Definition, Refresh, Grant Access	PTL_904099_031118	18-NOV-2003

4. On the Grant Access page, verify that the box next to **Expose as Provider** is checked.

Figure A–8 The Expose as Provider Check Box on the Provider Grant Access Page



Access: PTL_904099_031118_DEMO.EXAMPLE_APP

Provider Access

Check the Expose as Provider option to make this Portal DB Provider available as a portlet provider.

☑ Expose as Provider

Grant Access

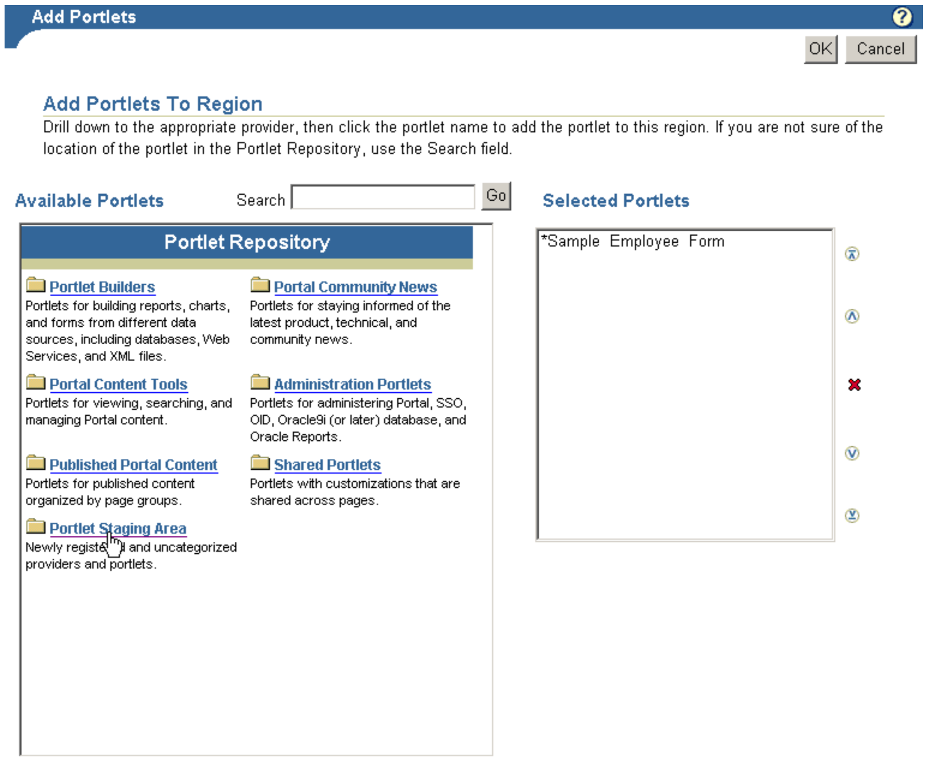
Enter the name of the user or group to which you want to grant access privileges, choose the privilege level to grant, then click Add. If you are not sure of the user or group name, click the appropriate browse icon and select from the list provided.

Grantee: Execute; Add

5. Click OK to save your change and exit the Grant Access page.

   Checking this check box enables users to publish portlets to the portlet repository under the auspices of this provider. Provided you have checked this box, the portlets you create using the Portlet Builder will be published to the portlet repository automatically (under the **Portlet Staging Area** node) when you finish them.

Figure A–9 The Portlet Staging Area Node in the Portlet Repository



Add Portlets To Region

Drill down to the appropriate provider, then click the portlet name to add the portlet to this region. If you are not sure of the location of the portlet in the Portlet Repository, use the Search field.

Portlet Builders
Portlets for building reports, charts, and forms from different data sources, including databases, Web Services, and XML files.

Portal Community News
Portlets for staying informed of the latest product, technical, and community news.

Portal Content Tools
Portlets for viewing, searching, and managing Portal content.

Administration Portlets
Portlets for administering Portal, SSO, OID, Oracle9i (or later) database, and Oracle Reports.

Published Portal Content
Portlets for published content organized by page groups.

Shared Portlets
Portlets with customizations that are shared across pages.

Portlet Staging Area
Newly registered and uncategorized providers and portlets.

*Sample Employee Form

A.1.4 Creating Portlets Using OracleAS Portal Wizards

This section lists and describes the types of portlets you can build using the portlet-building wizards available in OracleAS Portal. It provides general information on how to build a portlet and specific information on how to build forms, reports, charts, and lists of values.

Beginning with OracleAS Portal 10*g* (9.0.4), you can define portlets "in place" on a page. In other words, you can add a portlet to a page and define the complete portlet using the Portlet Builder without leaving the page. To add and create portlets in this way, you must have the required privileges: you must belong to the PORTAL_ADMINISTRATORS group and have the MANAGE CONTENT privilege on the page. This section describes how to define portlets outside of the page; then, you can modify and manage portlets independently, and later add the portlet to the page, as described in Section A.6.4, "Running the Component as a Portlet through the Portlet Customization Form".

This section contains the following subsections:

- Building Portlets Declaratively
- Building Forms Declaratively
- Building Reports Declaratively
- Building Forms and Reports against interMedia Rich Content
- Building Charts Declaratively
- Building Lists of Values Declaratively

You can build additional types of portlets through the portlet-building wizards. Table A–1 lists and describes the types of portlets you can build.

Table A–1 Types of Portlets Available Through Portlet-Building Wizards

Portlet Type	Description
Form	Displays a customized form that can be used as an interface for updating tables, executing stored procedures, and generating other customized forms. Through the OracleAS Portal portlet-building wizards, you can build three types of forms. ■ A **form based on a table or view** enables users to insert, update, and delete data in a database table or view. ■ A **master-detail form** displays a master row and multiple detail rows within a single HTML page. The form contains fields for updating values in two database tables or views. ■ A **form based on a procedure** enables users to insert, update, and delete data over a database stored procedure.
Report	Displays the data that you select from a database table or view in report format. The report can have tabular, form, or custom layout. You can build three types of reports: ■ **Query By Example (QBE) Report** allows users to query, insert, update and delete data in table and views. In the QBE report build wizard, you choose which data to display in the report. Or, you can allow end users to make their own queries in the QBE report's customization form. ■ **Reports from Query Wizard** guides you through building a report using a wizard to construct your SELECT statement. ■ **Reports from SQL Query** builds a report from your manually-constructed SQL query.
Chart	Displays data that you select from a database table or view as a bar chart.
Calendar	Displays the data that you select from a database table or view in calendar format.
Dynamic Page	Displays dynamically-generated HTML content on a Web page.
XML Component	Displays an XML page.
Hierarchy	Displays the data that you select from a database table or view as a graphical hierarchy of items containing up to three levels.
Menu	Displays an HTML-based menu that contains options that are linked to other menus, OracleAS Portal database portlets, or URLs.
URL	Renders the content of a URL target in a portlet.
Frame Driver	Displays a Web page with two frames. The queries entered in one frame control the content of the other.
Link	Displays a link that provides a hypertext jump between OracleAS Portal database portlets and other database portlets, database portlet customization forms, or any HTML page.
List of Values	Use LOVs when creating database portlets to preselect the possible values in an entry field. Users select a value from the list rather than enter it manually. You can build LOVs based on other LOVs.
Data Component	Displays data in a spreadsheet format.

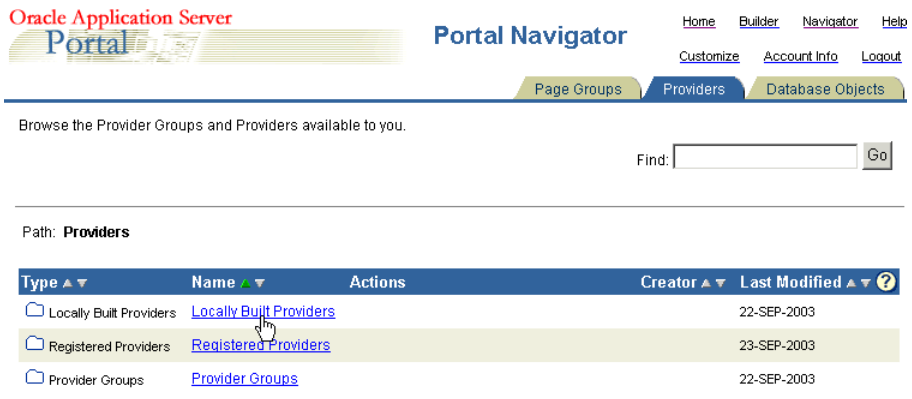
A.1.4.1 Building Portlets Declaratively

Note: To build a portlet, you must have at least the Edit privilege on the provider that will own the portlet. For more information, see "Creating a Provider for Locally Built Portlets".

To use a wizard to build a portlet:

1. Log in to OracleAS Portal.
2. Click the **Navigator** link at the top of the page.
3. Click the **Providers** tab to bring it forward.
4. Click the **Locally Built Providers** link.

Figure A–10 The Locally Built Providers Link in the Portal Navigator



Browse the Provider Groups and Providers available to you.

Path: **Providers**

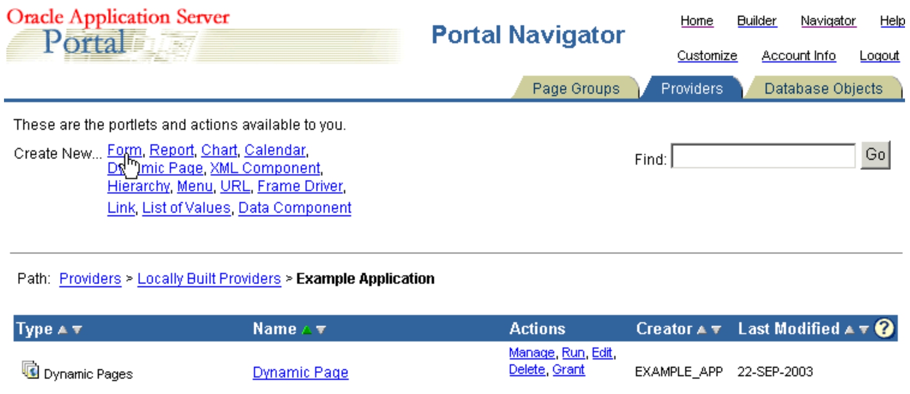
Type	Name	Actions	Creator	Last Modified
Locally Built Providers	Locally Built Providers			22-SEP-2003
Registered Providers	Registered Providers			23-SEP-2003
Provider Groups	Provider Groups			22-SEP-2003

5. Click the name of the OracleAS Portal database provider you want to host the new portlet.

   The **Name** column displays the names of all the providers on which you have privileges.

6. Next to **Create New...**, click the link for the type of portlet that you want to build, for example, **Chart** or **Form** (Figure A–11).

Figure A–11 The Form Link Next to Create New in the Portal Navigator



These are the portlets and actions available to you.

Create New... Form, Report, Chart, Calendar, Dynamic Page, XML Component, Hierarchy, Menu, URL, Frame Driver, Link, List of Values, Data Component

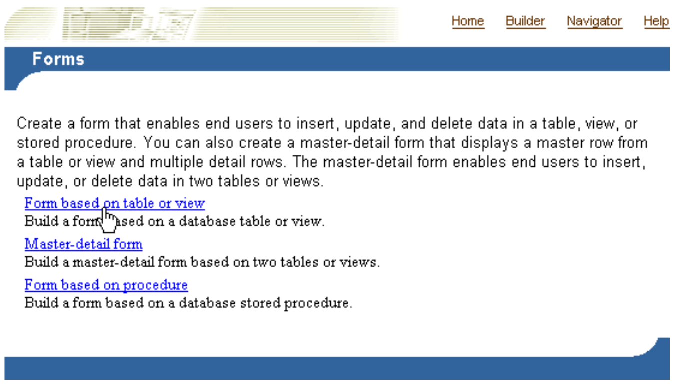
Path: Providers > Locally Built Providers > **Example Application**

Type	Name	Actions	Creator	Last Modified
Dynamic Pages	Dynamic Page	Manage, Run, Edit, Delete, Grant	EXAMPLE_APP	22-SEP-2003

See Table A–1 for a list and description of the types of portlets you can build.

Depending on the type of portlet you are building, OracleAS Portal displays either the first step of the portlet build wizard or a menu of additional choices. If you click **Form**, for example, you have the option of creating a form based on a table or view, a master-detail form, or a form based on a procedure (Figure A–12).

Figure A–12 Choices Available for Creating a Form in the Portlet Builder



Create a form that enables end users to insert, update, and delete data in a table, view, or stored procedure. You can also create a master-detail form that displays a master row from a table or view and multiple detail rows. The master-detail form enables end users to insert, update, or delete data in two tables or views.

Form based on table or view
Build a form based on a database table or view.

Master-detail form
Build a master-detail form based on two tables or views.

Form based on procedure
Build a form based on a database stored procedure.

7. When the first step of the portlet build wizard displays, follow the instructions shown on each step.

A progress indicator displays how many wizard steps are left to complete. The **Finish** button becomes available when you have provided the wizard with sufficient information to build the portlet. The fields you didn’t complete will populate with default values or will remain null.

If you are unsure of how to complete the fields on a particular step of the wizard, click the online help icon in OracleAS Portal for more information.

The next sections guide you through the specific steps required to build a particular type of portlet. For more information, see:

- Building Forms Declaratively
- Building Reports Declaratively
- Building Forms and Reports against interMedia Rich Content
- Building Charts Declaratively
- Building Lists of Values Declaratively

A.1.4.2 Building Forms Declaratively

Forms provide a means of inserting, updating, and deleting content from your database through a user-friendly interface. You control form layout through options available in the Forms build wizard. Wizard options enable you to control the look and feel of the overall form as well as the form’s individual entry fields and buttons.

You can perform JavaScript validation on user-specified values entered in any text field on the form. Additionally, you can add PL/SQL event handlers that run when a user clicks a button on the form. After successful submission of form content, you can specify a PL/SQL block or procedure that will execute.

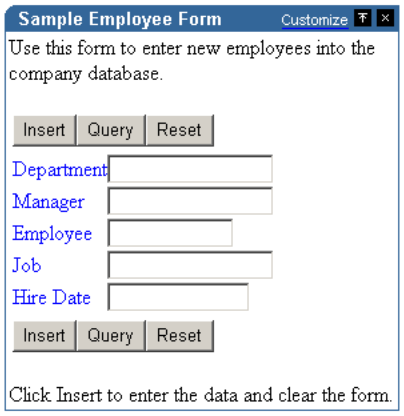
The portlet-building wizards in OracleAS Portal provide three construction methods for building forms:

- Forms based on tables or views
- Master-detail forms

- Forms based on a procedure

All of these forms are described in Table A–1. This section describes how to create a form based on a table or view (Figure A–13).

Figure A–13 Sample Form Based on a Table



Use this form to enter new employees into the company database.

Department	
Manager	
Employee	
Job	
Hire Date	

Click Insert to enter the data and clear the form.

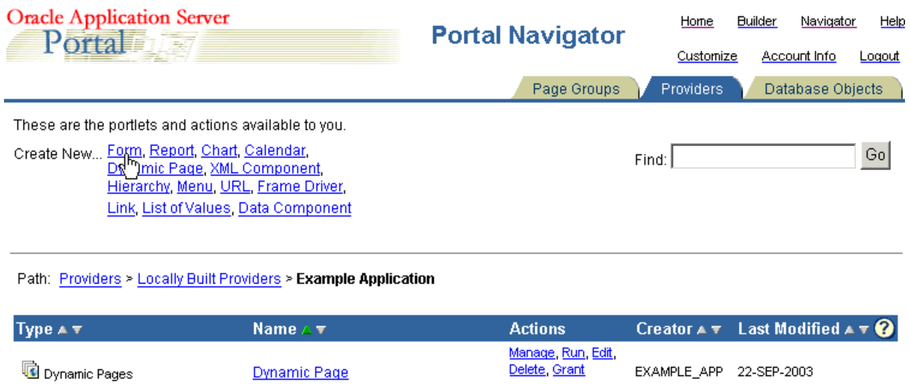
To create a form based on a table or view:

1. Follow the instructions detailed in "Building Portlets Declaratively".

   Return to this section once you complete step 6.

2. Click the **Form** link next to **Create New...** (Figure A–14).

Figure A–14 The Form Link Next to Create New in the Portal Navigator



These are the portlets and actions available to you.

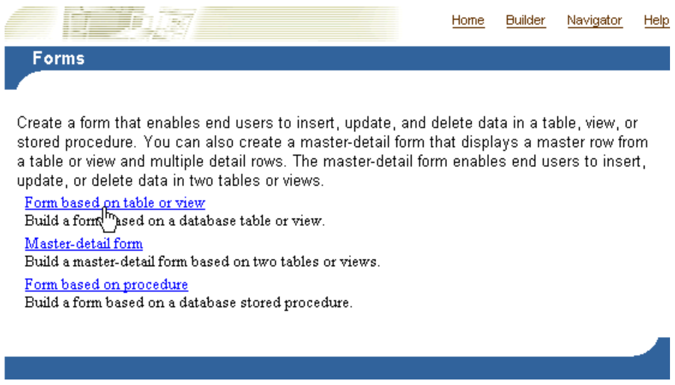
Create New... Form, Report, Chart, Calendar, Dynamic Page, XML Component, Hierarchy, Menu, URL, Frame Driver, Link, List of Values, Data Component

Path: Providers > Locally Built Providers > **Example Application**

Type	Name	Actions	Creator	Last Modified
Dynamic Pages	Dynamic Page	Manage, Run, Edit, Delete, Grant	EXAMPLE_APP	22-SEP-2003

3. On the Forms page, click **Form based on table or view**.

Figure A–15 The Form Based on Table or View Link in the Portlet Builder



Forms

Create a form that enables end users to insert, update, and delete data in a table, view, or stored procedure. You can also create a master-detail form that displays a master row from a table or view and multiple detail rows. The master-detail form enables end users to insert, update, or delete data in two tables or views.

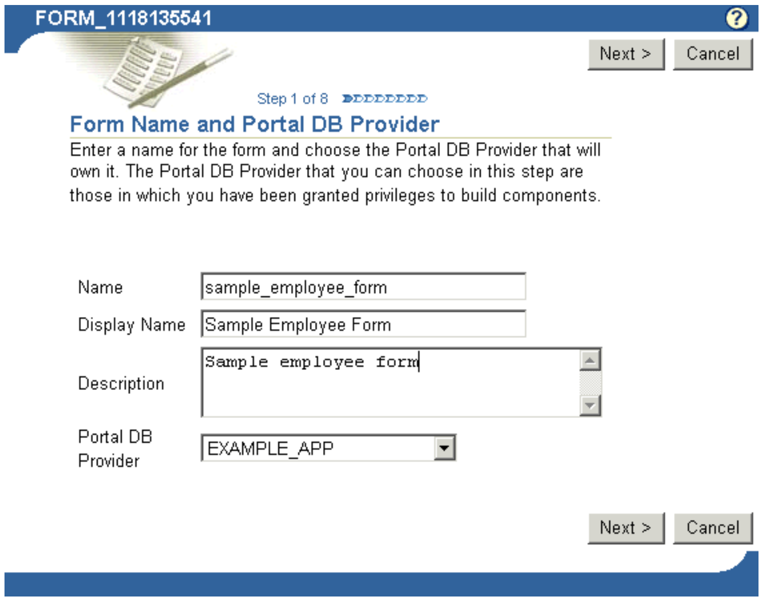
Form based on table or view
Build a form based on a database table or view.

Master-detail form
Build a master-detail form based on two tables or views.

Form based on procedure
Build a form based on a database stored procedure.

4. In the **Name** field, enter an internal name for the form (Figure A–16).

Figure A–16 Step One of the Form Wizard



Form Name and Portal DB Provider

Enter a name for the form and choose the Portal DB Provider that will own it. The Portal DB Provider that you can choose in this step are those in which you have been granted privileges to build components.

Name	sample_employee_form
Display Name	Sample Employee Form
Description	Sample employee form
Portal DB Provider	EXAMPLE_APP

The internal name is not published to users.

5. In the **Display Name** field, enter the name by which users will identify this form (Figure A–16).

   The display name is used at runtime. It displays as the title of the browser window as well as in the banner of the form. When the form is published as a portlet, the display name is used as the title of the portlet and in the List of Available Portlets in the Portlet Repository.

6. In the **Description** field, enter a description of this form (Figure A–16).

   The description displays below the portlet in the Portlet Repository. it can be a summary of the portlet's purpose, a classification of its type, or any other descriptive information. It may be useful within your organization to have a standard approach to what is included in the description.

7. From the list of **Portal DB Providers**, select the provider that will own this form (Figure A–16).

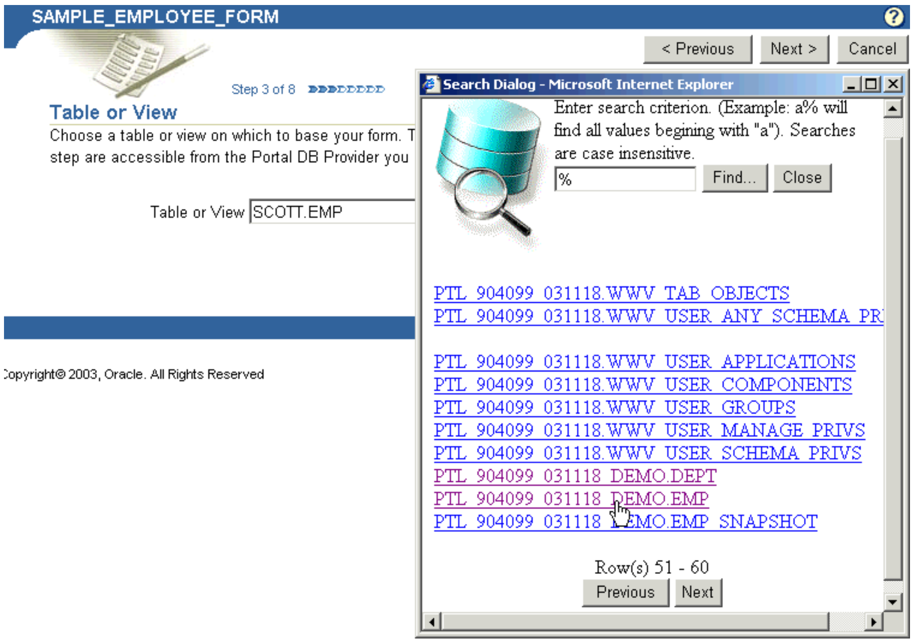
We recommend that you use a provider specifically created for owning, or hosting, your declaratively-built portlets. This list displays only those providers in which you have been granted privileges to build portlets.

8. Click **Next**.

9. Click the **Browse** icon next to the **Table or View** field, and choose a table or view on which to base your form (Figure A–17).

   You can use a constant in this field (for example, #APP_SCHEMA#.EMP, #PORTAL_SCHEMA#.DEPT) in lieu of a fixed value. For more information, see "Referencing the OracleAS Portal Schema".

Figure A–17   Selecting a Table or View



Choose a table or view on which to base your form. T
step are accessible from the Portal DB Provider you

Table or View SCOTT.EMP

Enter search criterion. (Example: a% will find all values begining with "a"). Searches are case insensitive.

%

PTL_904099_031118.WWV_TAB_OBJECTS
PTL_904099_031118.WWV_USER_ANY_SCHEMA_PR
PTL_904099_031118.WWV_USER_APPLICATIONS
PTL_904099_031118.WWV_USER_COMPONENTS
PTL_904099_031118.WWV_USER_GROUPS
PTL_904099_031118.WWV_USER_MANAGE_PRIVS
PTL_904099_031118.WWV_USER_SCHEMA_PRIVS
PTL_904099_031118_DEMO.DEPT
PTL_904099_031118_DEMO.EMP
PTL_904099_031118_DEMO.EMP_SNAPSHOT

Row(s) 51 - 60

The tables and views that you can choose in this step are those that are accessible from the Portal DB Provider you selected in the previous step. You can also enter the table name directly into the field.

10. Click **Next**.

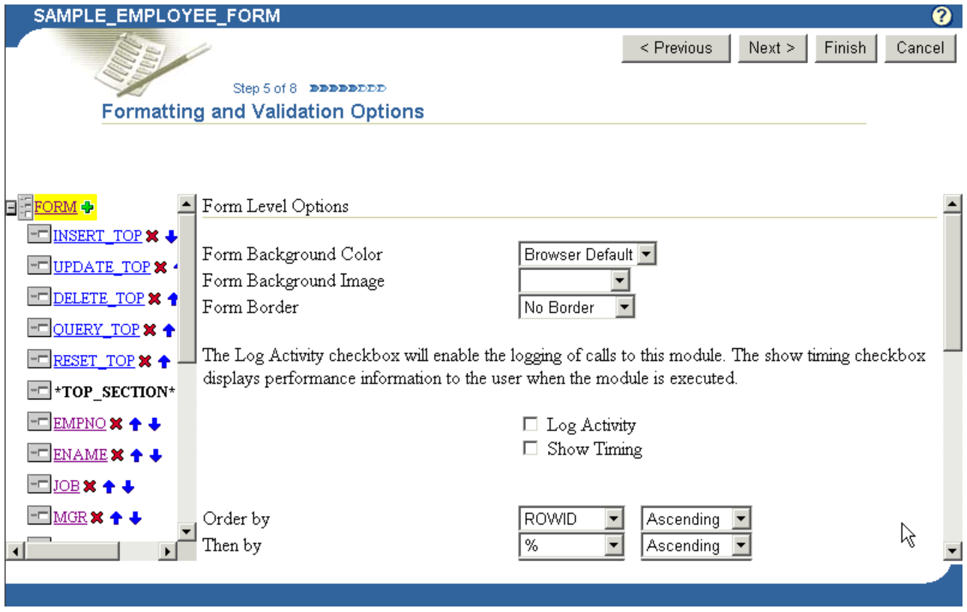
11. Select a form layout.

    Choose between **Tabular** or **Custom**. Tabular layouts are created automatically. Custom layouts are based on HTML code that you supply in a later step.

12. Click **Next**.

13. Set **Formatting and Validation Options** (Figure A–18):

Figure A–18 Form Formatting and Validation Options


SAMPLE_EMPLOYEE_FORM

< Previous | Next > | Finish | Cancel

Step 5 of 8

Formatting and Validation Options

FORM, INSERT_TOP, UPDATE_TOP, DELETE_TOP, QUERY_TOP, RESET_TOP, *TOP_SECTION*, EMPNO, ENAME, JOB, MGR

Form Level Options

Form Background Color: Browser Default
Form Background Image:
Form Border: No Border

The Log Activity checkbox will enable the logging of calls to this module. The show timing checkbox displays performance information to the user when the module is executed.

☐ Log Activity
☐ Show Timing

Order by: ROWID, Ascending
Then by: %, Ascending

a. Click **Form** in the left column to set up the formatting and validation options for the entire form (Figure A–18).

The formatting and validation options that appear in the right column relate to the object you have selected in the left column.

Table A–2 lists and describes form-level formatting and validation options.

Table A–2 Form-Level Formatting and Validation Options

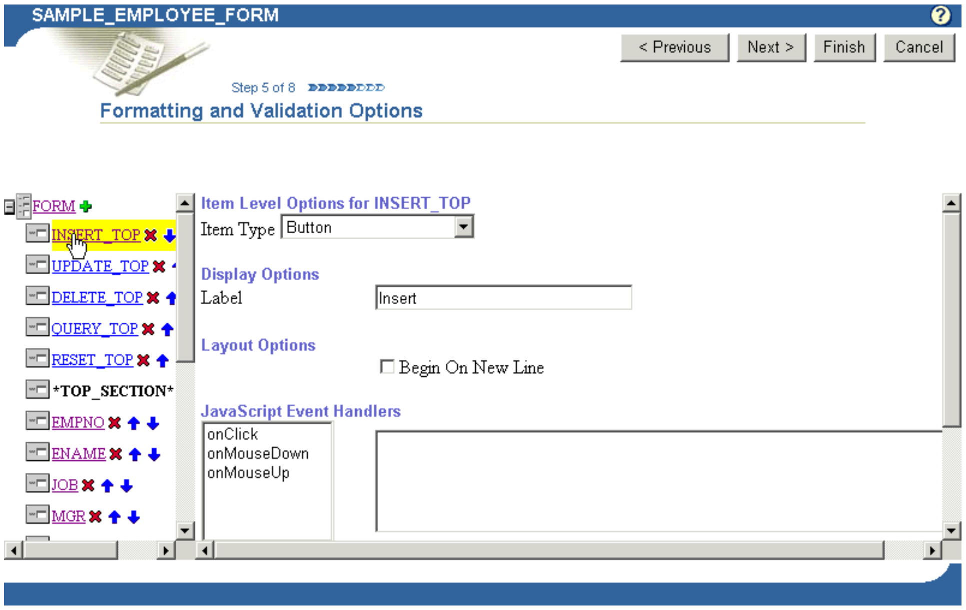
Option	Description
Form Background Color	Choose the background color of the form.
Form Background Image	Choose an image that will appear in the background of the form.
Form Border	Choose a style for the border around the form background.
Log Activity Check Box	Check to record activity in the OracleAS Portal log. Log information includes performance statistics as well as the names of users who request the form.
Show Timing Check Box	Check to display the time elapsed, starting from when the server receives the request to generate HTML for the form. Timing displays at the bottom of the form.
Order by	Choose the column by which you will order the data returned by the SELECT statement. Select the percent sign (%) if you do not wish to specify a column for ordering.
Then by	Choose the column(s) for subsequent ordering of the data returned by the SELECT statement. Select the percent sign (%) if you do not wish to specify a column for ordering.
Ascending	Choose **Ascending** (from a to z/0 to 9) or **Descending** (from z to a/9 to 0), depending on how you want the column values ordered.

Table A–2 (Cont.) Form-Level Formatting and Validation Options

Option	Description
On Successful Submission of a Form Execute this PL/SQL	Enter optional PL/SQL code that will execute after a user clicks a button on the form. The button must cause an operation, such as INSERT, to be performed on the table or view on which the form is based. For example, you might enter code that causes a message to display to the user when a table row is successfully updated.

b. Select a TOP_SECTION item in the left column (that is, INSERT_TOP, UPDATE_TOP, and the like) to set formatting and validation options in the right column for INSERT, UPDATE, and other like items that will display at the top of the form (Figure A–19).

Figure A–19 Button Formatting and Validation Options



Item Level Options for INSERT_TOP

Item Type: Button

Display Options

Label: Insert

Layout Options

☐ Begin On New Line

JavaScript Event Handlers

onClick
onMouseDown
onMouseUp

By default, insert, update, delete, query, and reset buttons appear at the top and bottom of each form. To prevent these buttons from displaying, click the Delete icon that appears next to the item in the left column.

Table A–3 lists and describes the formatting and validation options for buttons.

Table A–3 Formatting and Validation Options for Buttons

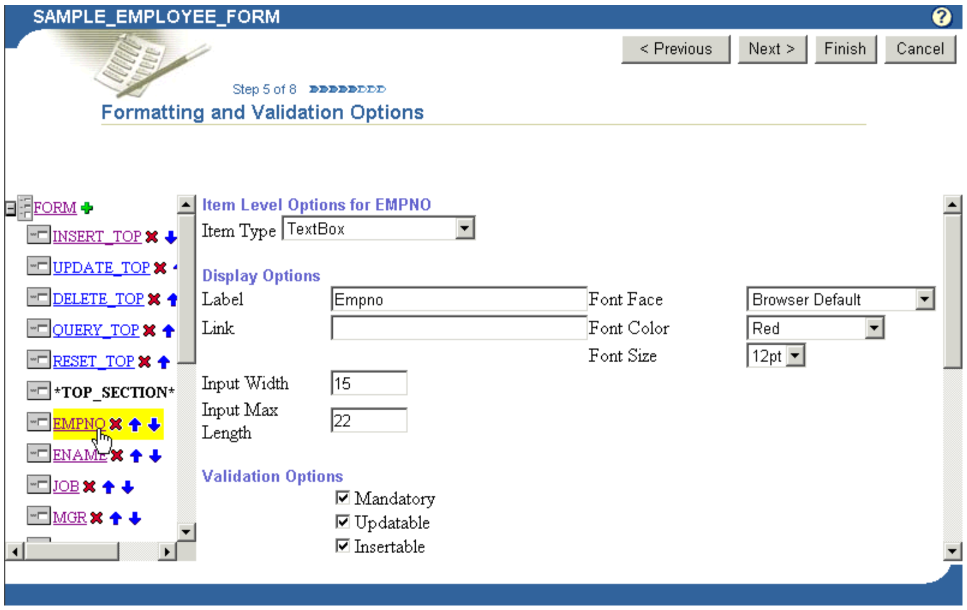
Option	Description
Item Type	Select the method for displaying the selected item type on your form. Note that the default for items, such as INSERT_TOP, is *button*. For INSERT, UPDATE, DELETE, QUERY, and RESET, you will likely most often want to use the default. If you choose another item type, you will need to add code to invoke the item's functionality (for example, an insert) through a JavaScript or PL/SQL Event Handler. Item types include: ■ **Blank**—Inserts a line break in between fields. ■ **Button**—Inserts a button. INSERT, UPDATE, DELETE, QUERY, and RESET items require the selection of an event handler further down on this page. ■ **CheckBox**—For INSERT, UPDATE, and so on, you must write a JavaScript handler for this to be meaningful. For columns, the underlying column must be at least varchar2(5) to accommodate entries of `TRUE` (checked) and `FALSE` (not checked). ■ **ComboBox**—Inserts a pop-list of values with an extra blank space that allows for manual entry of a value. You must already have defined a list of values to make this selection. You can enter a constant in the Default Value field, and define the default value as a constant in the Default Value Type field. For more information, see "Referencing the OracleAS Portal Schema". ■ **File Upload (Binary)**—Used to insert a BLOB column. ■ **File Upload (interMedia)**—Select this type for interMedia rich content, such as images, audio clips, and video clips. The table must have columns of the type ORDIMAGE, ORDAUDIO, or ORDVIDEO. For more information, see "Building Forms and Reports against interMedia Rich Content". ■ **Hidden**—Creates a hidden form field. ■ **Horizontal Rule**—Draws a horizontal line, similar to <hr>. ■ **Image**—Used to show an image. ■ **Label Only**—This shows only a prompt. ■ **Password**—This will have the behavior of a password field, where input will be hidden. ■ **TextArea**—Provides a large text area for the entry of multiple lines of text. ■ **TextBox**—Provides a text box.
Display Options	**Label**—Enter text that will display next to the button.
Layout Options	This section displays only if you selected **Tabular** in the Form Layout step. If you selected **Custom**, you can specify your own sophisticated layout in the next step using HTML code. **Begin on New Line**—Check this option to display the item on a new line on the form. Leaving this blank will display the item on the same line as the previous item or column field.
JavaScript Event Handlers	Choose an event and enter the JavaScript for the action you want to occur when that event happens. Refer to your JavaScript documentation for descriptions of the events on this list.

Table A–3 (Cont.) Formatting and Validation Options for Buttons

Option	Description
PL/SQL Button Event Handler	Choose a button event and enter the PL/SQL code for the action you want to occur when the event happens. For advanced customization, choose **Custom** from the drop-down list, and enter the PL/SQL code for the custom event.

c. Select a column name in the left column to set formatting and validation options in the right column for table or view columns that will display in the form (Figure A–20).

Figure A–20 Column Formatting and Validation Options



Item Level Options for EMPNO

Item Type: TextBox

Display Options

Label	Empno	Font Face	Browser Default
Link		Font Color	Red
		Font Size	12pt
Input Width	15		
Input Max Length	22		

Validation Options

- [✓] Mandatory
- [✓] Updatable
- [✓] Insertable

To prevent the display of a column, click the **Delete** icon next to it.

Table A–4 lists and describes the formatting and validation options available for table or view columns.

Table A–4 Formatting and Validation Options for Table or View Columns

Option	Description
Item Type	Defines how the column you've selected on the left will display on your form. Options are listed and described in Table A–3.

Table A–4 (Cont.) Formatting and Validation Options for Table or View Columns

Option	Description
Display Options	**Label**—Enter text to label this column on the form. Text will be display-only and will appear next to the field that represents this table or view column on the form. For example, you can add a label next to the field for the EMPNO column called `Employee Identification Number`. **Link**—Specify a link to another OracleAS Portal portlet or URL. If you specify this option, the Label appears on the form as a hypertext link. If you want to link the form to another URL, type the URL location in the Link text box. If you want to link to an OracleAS Portal portlet, you can type the name of the package containing the portlet, for example: `SCHEMA.portlet.SHOW` `SCHEMA` is the name of the schema that owns the portlet; `portlet` is the portlet name; and `SHOW` is the procedure used to display the portlet. You can also specify `SHOW_PARMS` to display the customization form for the portlet. **Font Face, Color, Size**—Specify the font characteristics for displaying text associated with the selected column on the form. **Input Width**—Enter the character width of the field associated with the selected column. **Input Max Length**—Enter the maximum length of the data that can be entered into the form field associated with the selected column.

Table A–4 (Cont.) Formatting and Validation Options for Table or View Columns

Option	Description
Validation Options	**Mandatory**—Check to require that the user specify a value in the field associated with the selected column before submitting the form. **Updatable**—Check to enable the user to update the column. Leaving this blank prevents updates. This feature is useful when you want to allow users to update some table columns in the form, but only view others. For example, by clearing **Updatable** for employee names and ID numbers, and checking **Updatable** for the employee's department, you can create a form that enables users to update an employee's department (for example, when the employee is transferred), but disables users from updating the employee name and ID number. When users display views, updatable columns display in black, non-updatable columns display in blue, and mandatory columns display in red. **Insertable**—Check to store the value of the column in the database when the user creates a new record. Leaving this blank prevents the value from being stored in the database (that is, it removes the column from the `INSERT` statement). **Default Value**—Enter a default value for the field associated with the selected column. Users can accept this value or specify their own. Specify a constant, function, expression, or SQL query. **Default Value Type**—Specify a type for the default value entered in the previous field (**Default Value**). **Format Mask**—Enter an Oracle display format for columns containing numeric and date data types. For example, you could enter `DD/MM/YYYY` to display dates according to this pattern; or you could enter `999,999,999.99` to place commas and decimals according to this pattern. **Note:** Refer to the Oracle database documentation (`http://www.oracle.com/technology/index.html`) for information about date and numeric formatting options. **Field Level Validation**—Choose a JavaScript validation routine that verifies whether the user enters a valid value in the field. For example, you could choose a JavaScript routine called IsNumber that verifies that a number has been typed in a SALARY field. Field validation providers are implemented in JavaScript and run when the OnChange condition occurs; for example, when the user presses the Return key after entering a value in the field.
	Form Level Validation—Choose a JavaScript validation provider that verifies whether the user enters a valid value in the field. Form validation providers run when the user submits the information, for example, after the user clicks the Insert button on the form.

Table A–4 (Cont.) Formatting and Validation Options for Table or View Columns

Option	Description
Layout Options	This section displays only if you selected Tabular in the Form Layout step. If you selected Custom, you can specify your own sophisticated layout in the next step using HTML code. **Begin on New Line**—Check to create a line break before the field associated with the selected column. Leaving this blank will display the column field on the same line as the previous column field. **Row Span**—Enter the number of HTML cells that can be used to display the field vertically on the browser page. **Col Span**—Enter the number of HTML cells that can be used to display the field horizontally on the browser page.
JavaScript Event Handlers	Choose an event and enter the JavaScript for the action you want to occur when the event takes place. Refer to your JavaScript documentation for descriptions of the events in the list.

d. Select a BOTTOM_SECTION item in the left column (that is, PREVIOUS, NEXT, INSERT_BOTTOM, UPDATE_BOTTOM, and the like) to set formatting and validation options in the right column for PREVIOUS, NEXT, INSERT, UPDATE, and other like items that will display at the bottom of the form.

   See step **b** of this task for more information.

14. Click **Next**.
15. If you selected a **Custom** layout:

    Enter the HTML code that will control the layout of your form. The text boxes on this page contain sample HTML code that creates a table. Using this model, each column you selected in the Tables or Views step of this wizard is formatted to display in a table row on the form.

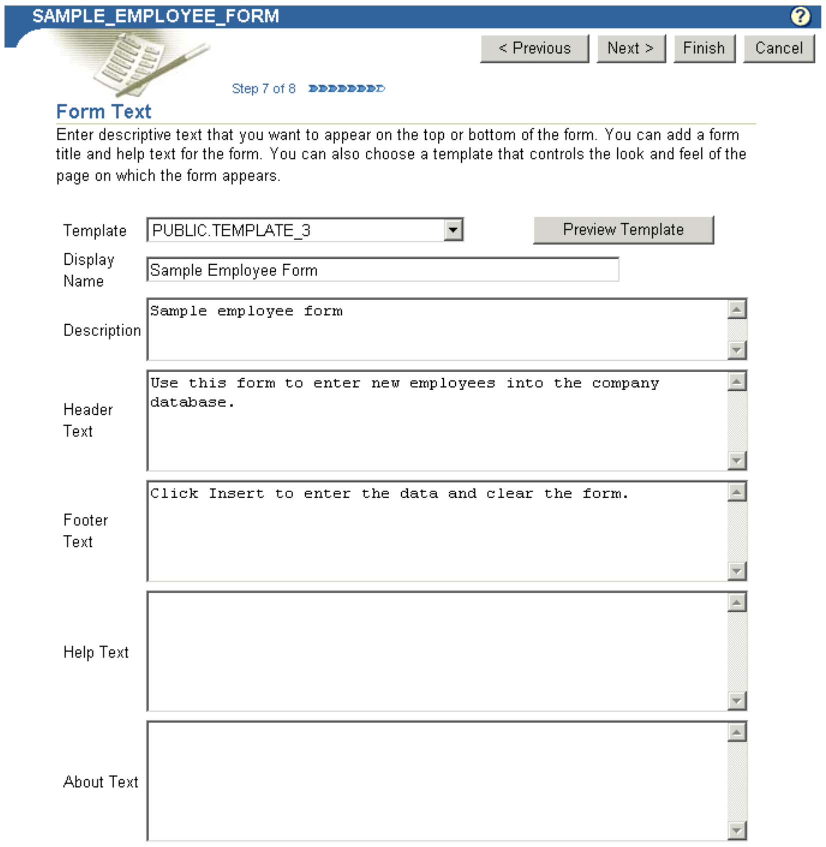
    Column *labels* are specified by the suffix `.LABEL` in the sample code. Column *values* are specified by the suffix `.ITEM` in the sample code.

    You can update the sample code with any HTML of your own, provided you do not change the column values (`.ITEM` suffix) or labels (`.LABEL` suffix). You can, however, delete column values from this code.

    If you selected a **Tabular** layout:

    Enter descriptive text that you want to appear on the top or bottom of the form (Figure A–21). You can add a form title and help text for the form. You can also choose a template that controls the look and feel of the page on which the form appears.

Figure A–21 Entering Header, Footer, and Other Text for a Form



Enter descriptive text that you want to appear on the top or bottom of the form. You can add a form title and help text for the form. You can also choose a template that controls the look and feel of the page on which the form appears.

Template	PUBLIC.TEMPLATE_3
Display Name	Sample Employee Form
Description	Sample employee form
Header Text	Use this form to enter new employees into the company database.
Footer Text	Click Insert to enter the data and clear the form.
Help Text	
About Text	

Table A–5 lists and describes the options available on this page.

Table A–5 Form Text Options on Tabular Format Forms

Option	Description
Template	Choose a template to set the look and feel of form elements such as background colors and images and the image that appears in the upper left corner of the page. Templates are used only when your form is displayed in full-page view, and not when it is displayed as a portlet. When a form is displayed as a portlet, the host page's style controls the look and feel.
Preview Template	Click to view the appearance of the template currently selected in the **Template** drop-down list.
Display Name	Edit the display name of the form. You can specify HTML in this field.
Header Text	Enter any introductory text that you want to display at the top of the form, just below the title, but above any buttons. You can specify HTML in this field.
Footer Text	Enter any text that you want to display at the bottom of the form. You can specify HTML in this field.

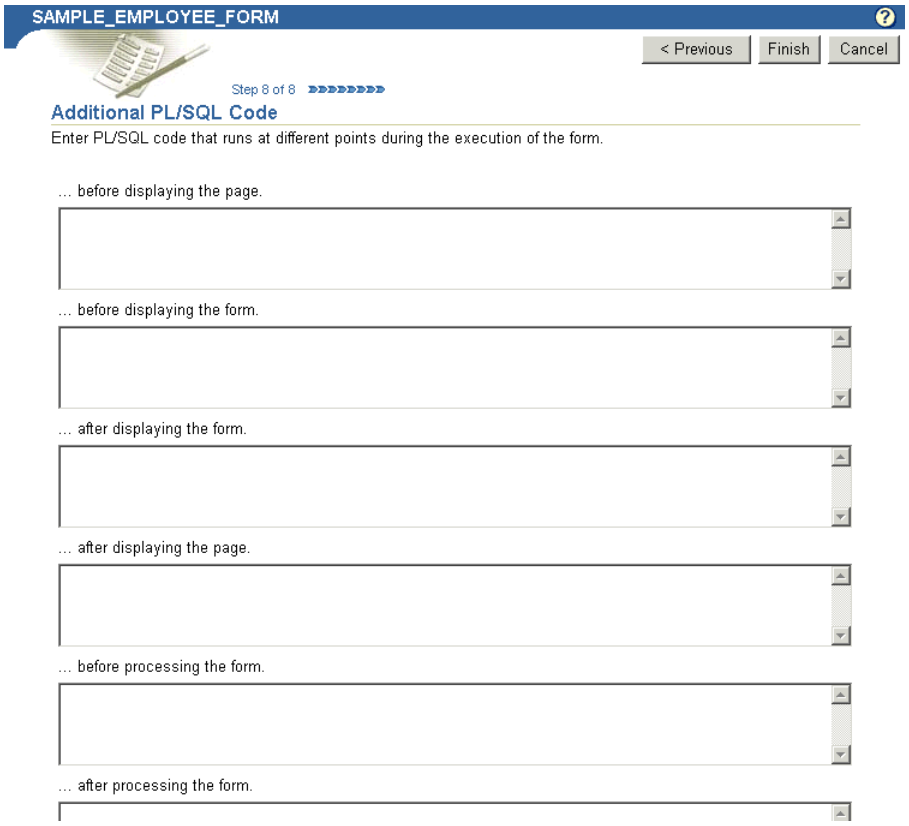
Table A–5 (Cont.) Form Text Options on Tabular Format Forms

Option	Description
Help Text	Enter any text that you want to display in a help page for the form. OracleAS Portal automatically adds a help button to the form. Users can click this button to link to a page displaying the help text that you enter here. You can specify HTML in this field.

16. Click **Next**.
17. Optionally, enter the PL/SQL that will run at different points during the execution of the HTML code that creates this form (Figure A–22).

    In the PL/SQL you enter into these fields, you can use constants for the OracleAS Portal schema, the application schema, and the application name. For more information, see "Referencing the OracleAS Portal Schema".

Figure A–22 Optional Fields for Entering Additional PL/SQL Code



Additional PL/SQL Code

Enter PL/SQL code that runs at different points during the execution of the form.

... before displaying the page.

... before displaying the form.

... after displaying the form.

... after displaying the page.

... before processing the form.

... after processing the form.

Table A–6 lists and describes the options on this page.

Table A–6 Additional PL/SQL Code Options

Option	Description
... before displaying the page.	Enter a PL/SQL procedure that will execute before the HTML for the form is generated (that is, before the <FORM> tag is generated). In spite of the name of this field, the PL/SQL procedure actually executes after the page itself is displayed.

Table A–6 (Cont.) Additional PL/SQL Code Options

Option	Description
… before displaying the form.	Enter a PL/SQL procedure that will execute after the <FORM> tag but before displaying any form elements.
… after displaying the form.	Enter a PL/SQL procedure that will execute before the ending </FORM> tag and after all form elements are displayed.
… after displaying the page.	Enter a PL/SQL procedure that will execute after the ending </FORM> tag.
… before processing the form.	Enter a PL/SQL procedure that will execute before the form is processed.
… after processing the form.	Enter a PL/SQL procedure that will execute after the form is processed.

Note: If you want to display output from a PL/SQL procedure, you must enter the PL/SQL in one of these four fields:

- before displaying the page
- before displaying the form
- after displaying the form
- after displaying the page

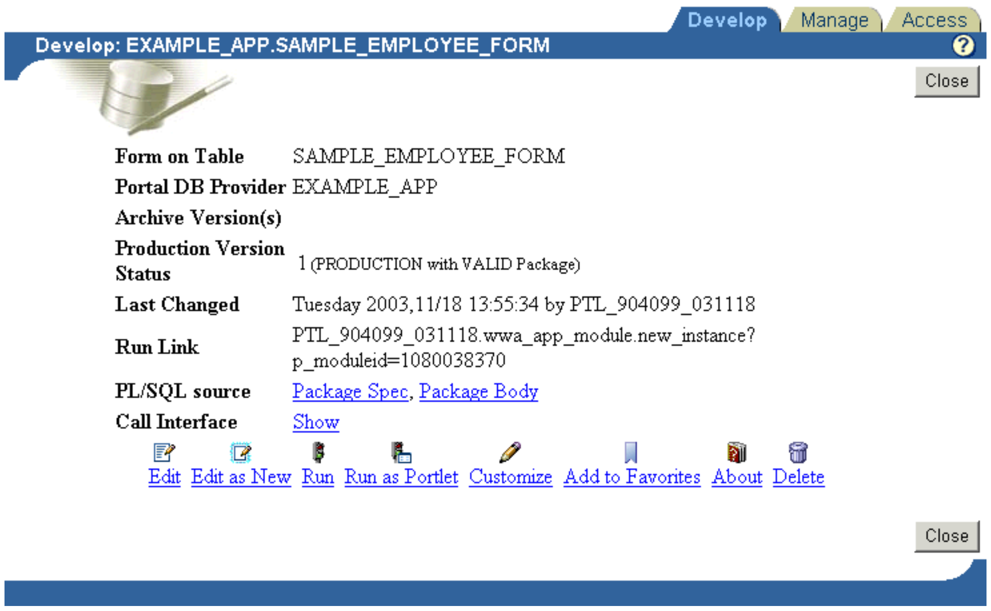
You cannot display output from a PL/SQL procedure in the before/after processing the form fields; the output will not display.

Additionally, if you want to display output from your PL/SQL procedure, you must specify the schema name with the procedure. This is required even if the form and the procedure are in the same schema.

18. Click **Finish** to save your changes.

Once you click **Finish**, Portal jumps to a summary page from which you can manage your form (Figure A–23). These tasks are discussed in later sections in this appendix.

Figure A–23 Form Summary Page



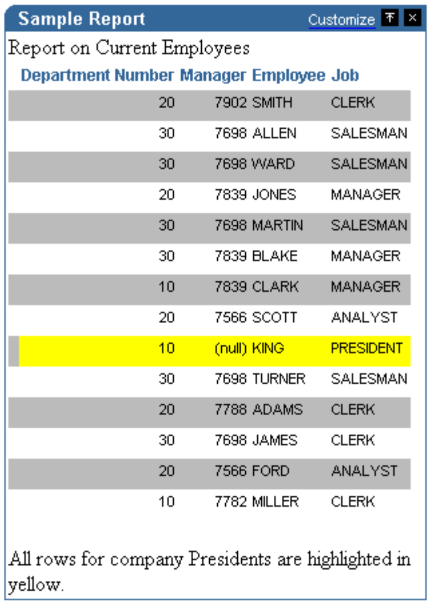
Develop: EXAMPLE_APP.SAMPLE_EMPLOYEE_FORM

Form on Table	SAMPLE_EMPLOYEE_FORM
Portal DB Provider	EXAMPLE_APP
Archive Version(s)	
Production Version Status	1 (PRODUCTION with VALID Package)
Last Changed	Tuesday 2003,11/18 13:55:34 by PTL_904099_031118
Run Link	PTL_904099_031118.wwa_app_module.new_instance?p_moduleid=1080038370
PL/SQL source	Package Spec, Package Body
Call Interface	Show

19. Click **Close** to return to the Portal Navigator.

A.1.4.3 Building Reports Declaratively

Figure A–24 A Sample Report



Report on Current Employees

Department Number	Manager	Employee	Job
20	7902	SMITH	CLERK
30	7698	ALLEN	SALESMAN
30	7698	WARD	SALESMAN
20	7839	JONES	MANAGER
30	7698	MARTIN	SALESMAN
30	7839	BLAKE	MANAGER
10	7839	CLARK	MANAGER
20	7566	SCOTT	ANALYST
10	(null)	KING	PRESIDENT
30	7698	TURNER	SALESMAN
20	7788	ADAMS	CLERK
30	7698	JAMES	CLERK
20	7566	FORD	ANALYST
10	7782	MILLER	CLERK

All rows for company Presidents are highlighted in yellow.

Reports display data from tables and views (Figure A–24). You can base a report on multiple tables or views using a JOIN condition. You can highlight data that satisfies conditions you specify. For example, you can display in bold red text the records of all employees who have been employed less than two years. You can create hypertext links from values displayed in the report to other OracleAS Portal database portlets or URLs.

Additionally, the wizards enable you to specify other options to achieve the following results:

- Limit data displayed in the report
- Sum column values
- Use logical operators to select data within the columns
- Format data in the report
- Format the report's customization form
- Specify PL/SQL code that executes at various points in the report.

The OracleAS Portal Portlet Builder wizards provide three construction methods for building reports:

- Query By Example (QBE) Reports
- Reports From Query Wizard
- Reports From SQL Query

These reports are described in Table A–1. This section describes how to build a report using the Query Wizard.

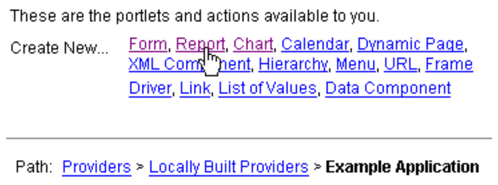
To build a report using the Query Wizard:

1. Follow the instructions detailed in "Building Portlets Declaratively".

   Return to this section once you complete step 6.

2. Click the **Report** link next to **Create New...**(Figure A–25)

Figure A–25 The Report Link in the Example Applications Portal DB Provider



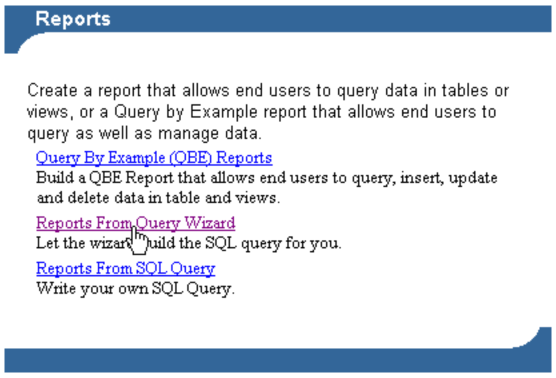
These are the portlets and actions available to you.

Create New... Form, Report, Chart, Calendar, Dynamic Page, XML Component, Hierarchy, Menu, URL, Frame Driver, Link, List of Values, Data Component

Path: Providers > Locally Built Providers > **Example Application**

3. On the Reports page, click **Reports From Query Wizard** (Figure A–26).

Figure A–26 The Reports From Query Wizard Link in the Portlet Builder



Reports

Create a report that allows end users to query data in tables or views, or a Query by Example report that allows end users to query as well as manage data.

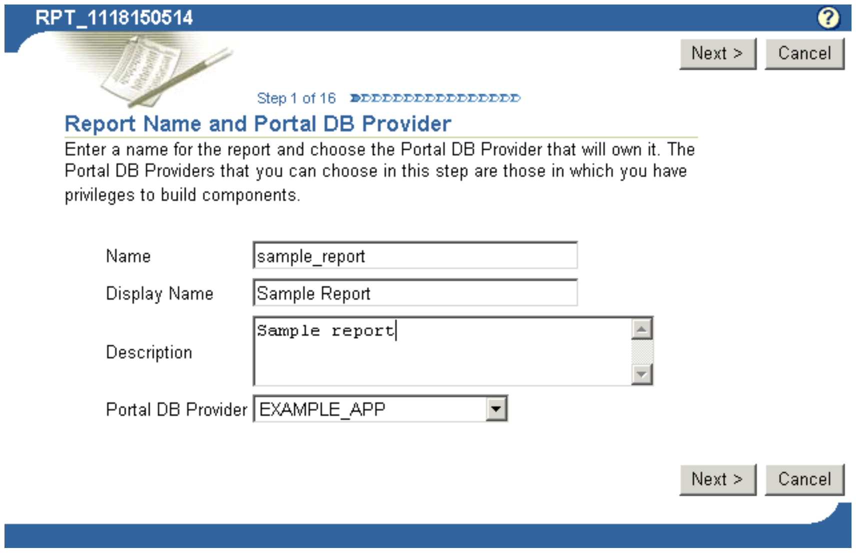
Query By Example (QBE) Reports
Build a QBE Report that allows end users to query, insert, update and delete data in table and views.

Reports From Query Wizard
Let the wizard build the SQL query for you.

Reports From SQL Query
Write your own SQL Query.

4. In the **Name** field, enter an internal name for the report (Figure A–27).

Figure A–27 The First Step in the Report From Query Wizard



Report Name and Portal DB Provider

Enter a name for the report and choose the Portal DB Provider that will own it. The Portal DB Providers that you can choose in this step are those in which you have privileges to build components.

Name	sample_report
Display Name	Sample Report
Description	Sample report
Portal DB Provider	EXAMPLE_APP

The internal name is not published to users.

5. In the **Display Name** field, enter a display name for the report (Figure A–27).

   The display name is used at runtime. It displays as the title of the browser window as well as in the banner of the report. When the report is published as a portlet, the display name is used as the title of the portlet and in the List of Available Portlets in the Portlet Repository.

6. From the list of **Portal DB Providers**, select the provider that will own this report (Figure A–27).

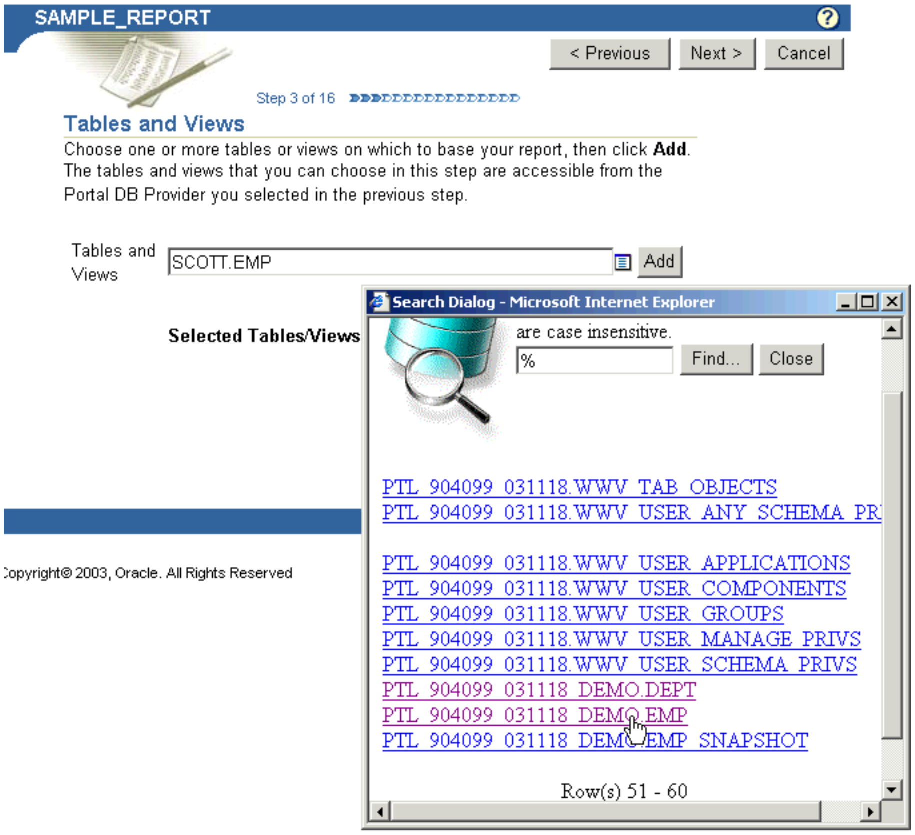
   We recommend that you use a provider specifically created for owning, or hosting, your declaratively-built portlets. This list displays only those providers in which you have been granted privileges to build portlets.

7. Click **Next**.

8. From the **Tables and Views** list, choose one or more tables or views on which to base this report (Figure A–28).

Note: In the **Tables and Views** field, you can use constants for the OracleAS Portal schema and the application schema (for example, enter #APP_SCHEMA#.EMP). For more information, see "Referencing the OracleAS Portal Schema".

Figure A–28 Selecting a Data Source for the Report



Tables and Views

Choose one or more tables or views on which to base your report, then click **Add**. The tables and views that you can choose in this step are accessible from the Portal DB Provider you selected in the previous step.

Tables and Views: SCOTT.EMP Add

Selected Tables/Views

Copyright© 2003, Oracle. All Rights Reserved

Search Dialog - Microsoft Internet Explorer

are case insensitive.

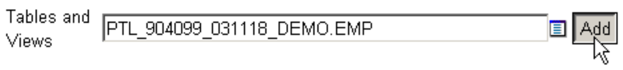
% Find... Close

PTL_904099_031118.WWV_TAB_OBJECTS
PTL_904099_031118.WWV_USER_ANY_SCHEMA_PR
PTL_904099_031118.WWV_USER_APPLICATIONS
PTL_904099_031118.WWV_USER_COMPONENTS
PTL_904099_031118.WWV_USER_GROUPS
PTL_904099_031118.WWV_USER_MANAGE_PRIVS
PTL_904099_031118.WWV_USER_SCHEMA_PRIVS
PTL_904099_031118_DEMO.DEPT
PTL_904099_031118_DEMO.EMP
PTL_904099_031118_DEMO.EMP_SNAPSHOT

Row(s) 51 - 60

Click **Add** after each selection (Figure A–29).

Figure A–29 Clicking Add to Add a Data Source


Tables and Views: PTL_904099_031118_DEMO.EMP Add

9. Click **Next**.

10. If you are basing your report on multiple tables or views, identify the columns that will be joined to one another.

This page displays only when you have selected multiple columns or views in the previous step. Default join conditions are automatically generated. You can accept or modify these defaults.

11. Click **Next**.

12. Select the columns you want to display in your report (Figure A–30).

Figure A–30 Selecting the Columns that Will Display in the Report

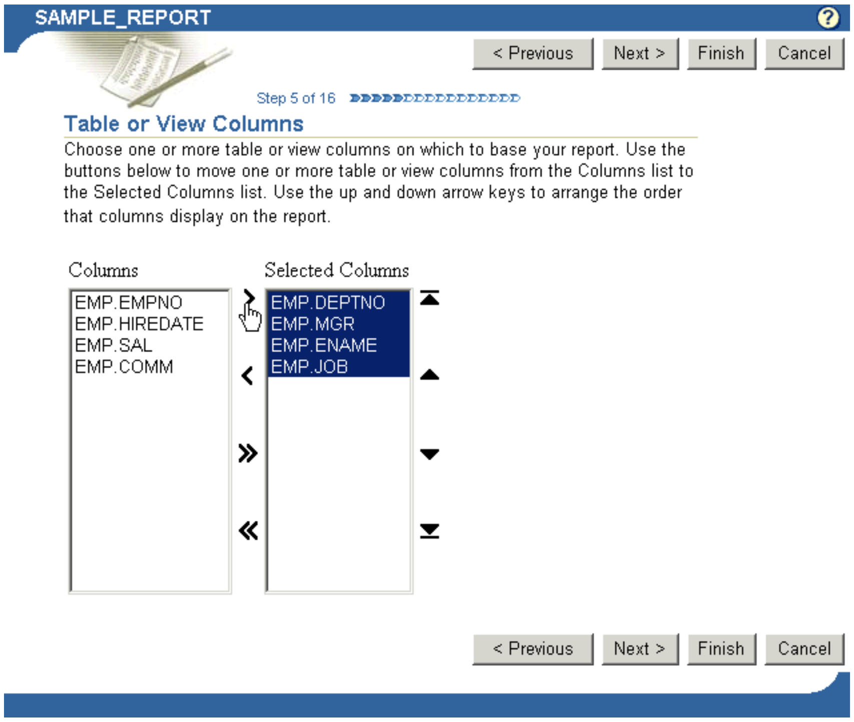


Table or View Columns

Choose one or more table or view columns on which to base your report. Use the buttons below to move one or more table or view columns from the Columns list to the Selected Columns list. Use the up and down arrow keys to arrange the order that columns display on the report.

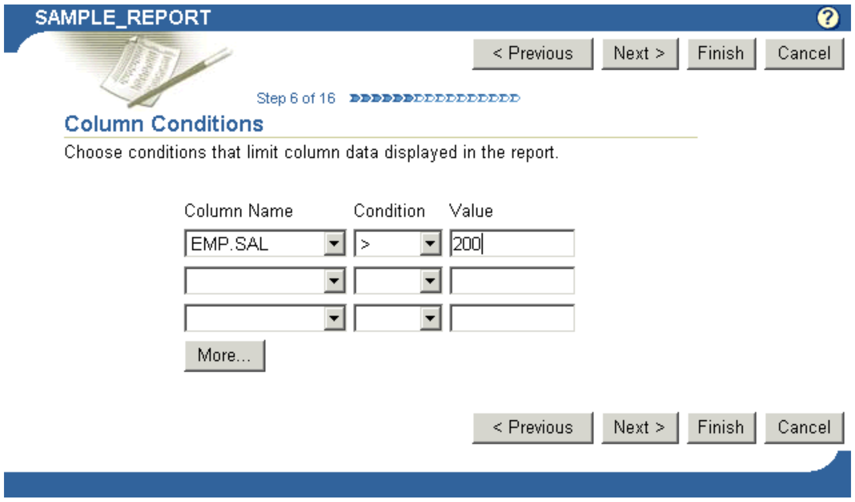
Columns	Selected Columns
EMP.EMPNO	EMP.DEPTNO
EMP.HIREDATE	EMP.MGR
EMP.SAL	EMP.ENAME
EMP.COMM	EMP.JOB

Click the arrows between the two columns to move individual selections or the entire list from one column to the other. Use the rearrange icons to the right of the right column to set the order of appearance of your selected columns. Select no more than 255 columns.

13. Click **Next**.

14. Optionally, specify conditions that limit the data displayed in the report (Figure A–31).

Figure A–31 Setting Column Conditions



Column Conditions

Choose conditions that limit column data displayed in the report.

Column Name	Condition	Value
EMP.SAL	>	200

Table A–7 lists and describes the options on this page.

Table A–7 Options on the Column Conditions Page

Option	Description
Column Name	Choose a column if you want to specify a condition for the data in the report. The column need not be one that was selected for display. For example, you can choose a salary column that is not selected for display to limit the displayed data to only those employees that meet your specified salary condition, such as salary must be greater than $200. To set this condition, choose (for example) `SAL` under **Column Name**, then choose a **Condition** of >, and enter `200` as your **Value**.
Condition	Choose conditions that will define **Column Name**'s relationship to **Value**. Choose from: ■ equal to (=) ■ greater than (>) ■ greater than or equal to (>=) ■ less than (<) ■ less than or equal to (<=) ■ like ■ not equal to (!=) ■ null ■ not null ■ in ■ not in To specify multiple values after an `IN` or `NOT IN` condition, type a colon (:) between each value, for example, 10:20:30.
Value	Enter a value that limits which rows of the identified column are displayed in the report.
More …	Click to display more fields for adding more conditions on this page.

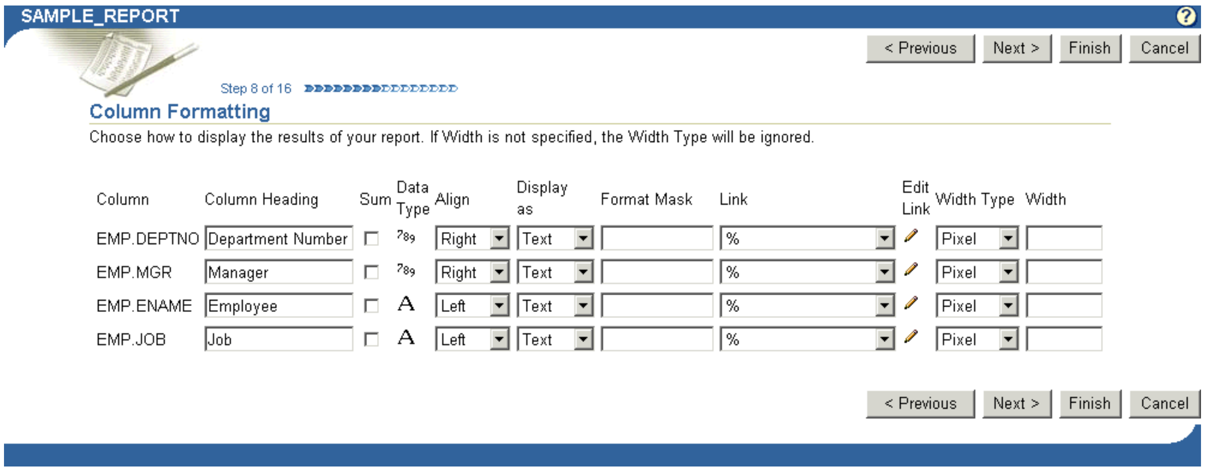
15. Click **Next**.
16. Select from three report layout types: Tabular, Form, or Custom.
    - **Tabular** layouts are based on options you choose in the Report Building wizard. Results are displayed in a table format, where each record represents a row in the underlying table or view, and each data column displays as a column in the table.
    - **Form** layouts are based on options you choose in the Report Building wizard. Results are displayed in rows, where each data column is displayed in a separate row.
    - **Custom** layouts are based on HTML code that you supply in a later wizard step. Because you specify your own HTML code, you will have a greater level of control over the appearance of your report than you would using a structured (tabular or form) layout.

    This description will proceed as if you have selected a tabular layout. Some of the wizard steps you would take with other types of layouts are not included in the construction of a tabular report. For example, with a custom layout, you would be given an opportunity to specify your custom HTML code. That step does not display in the wizard when you choose a tabular layout.

17. Click **Next**.

18. Specify column formatting (Figure A–32).

Figure A–32 The Column Formatting Page



Column Formatting

Choose how to display the results of your report. If Width is not specified, the Width Type will be ignored.

Column	Column Heading	Sum	Data Type	Align	Display as	Format Mask	Link	Edit Link	Width Type	Width
EMP.DEPTNO	Department Number	☐	789	Right	Text		%	✎	Pixel	
EMP.MGR	Manager	☐	789	Right	Text		%	✎	Pixel	
EMP.ENAME	Employee	☐	A	Left	Text		%	✎	Pixel	
EMP.JOB	Job	☐	A	Left	Text		%	✎	Pixel	

Table A–8 lists and describes the options presented on this page.

Table A–8 Column Formatting Options for a Tabular Report

Option	Description
Column	Displays the names of the table or view columns you selected in a previous wizard step.
Column Heading	Enter the name you will use to identify this column in your report. For example, if your table has an `EMPNO` column, you can enter `Employee ID Number`, in lieu of the column's actual name.
Sum	Select to add values together and display the result in the report.
Data Type	Displays the data type of the data in the associated column.
Align	Choose whether to align data to the left, center, or right margin of a report column. By default, numeric data aligns to the right, and alphabetical data aligns to the left.
Display as	Choose from: ■ **Text**—Displays the value in the report without interpreting any associated HTML tags. For example, the value <i>Name</i> displays as <i>Name</i>. ■ **HTML**—Displays the value in the report using associated HTML tags. For example, the value `<i>Name</i>` displays as *`Name`*. ■ **Hidden**—Hides the value from view in the report. For example, choose Hidden for columns whose values are included for link parameters that you do not want displayed in the report.
Format Mask	Enter an Oracle display format for columns that contain date and numeric data types. See Table A–9 and Table A–10 for examples of date and number format masks.

Table A–8 (Cont.) Column Formatting Options for a Tabular Report

Option	Description
Link	Create a link from the column to another OracleAS Portal portlet. The column value will display as a hypertext link. You can specify a link only if one has already been created for this column and has been stored in the database.
Edit Link	Click to display the Set Link Parameters window. In this window, you can set the values of the link parameters to pass to the target portlet. Enter a static value, such as 10 or KING, or select from the list of columns you identified in the table or view columns step earlier in this wizard. If you need the value of a column for a link parameter, but do not want that column displayed in the report, you can set the **Display as** value to **Hidden**.
Width Type	Specify one of the following: ▪ **Char**—Displays the output in the specified number of characters per line. For example, if you enter `20` in **Width**, the report displays 20 characters of the column data in each line. If the number of characters exceeds the specified number, the remaining characters are wrapped to the next line. Note, however, that if the output is ASCII format, **Char** is based on the column width. ▪ **Pixel**—Displays the output in the specified number of pixels per row. For example, if you enter `10` in **Width**, the column data displays 10 pixels of data per line. ▪ **Percent**—Displays the output in a column that takes up the specified percentage of the table. For example, if you enter `25` in **Width**, the column data displays in a column that takes up 25 percent of the overall table width.
Width	Enter the numeric value for the width type.

You cannot use format masks in reports generated in Excel format. Table A–9 and Table A–10 provide examples of date and number format masks, which you can use in the Format Mask field on this wizard page.

Table A–9 Examples of Date Format Masks

Sample Date Format	Date Displayed
MM/DD/RR	03/04/85
Mon. DD, RRRR	Mar. 4, 1985
Day Month DD fmHH:MI AM	Monday March 4 11:35 PM
Dy Mon ddth fmHH24:MI:SS	Mon Mar 4th 23:35:22
Day "the" ddthsp "of" Month	Monday the fourth of March

Table A–10 Examples of Number Format Masks

Sample Number Format	Number	Number Displayed
-0000	7934	"7934"
	-7934	"-7934"
-00000	7934	"07934"
-NNNN	7639	"7639"
	535	"535"

Table A–10 (Cont.) Examples of Number Format Masks

Sample Number Format	Number	Number Displayed
+NNNN	100	"100"
	-99	"-99"
-%NN	10	"%10"
$<NNNNN.00>	120014	" $1200.14"
+KKNNNNN.00	1950	"+ 1950.00"
$ <NNNNN.00>	1200	"$ 1200.00
	400	"$ 400.00"

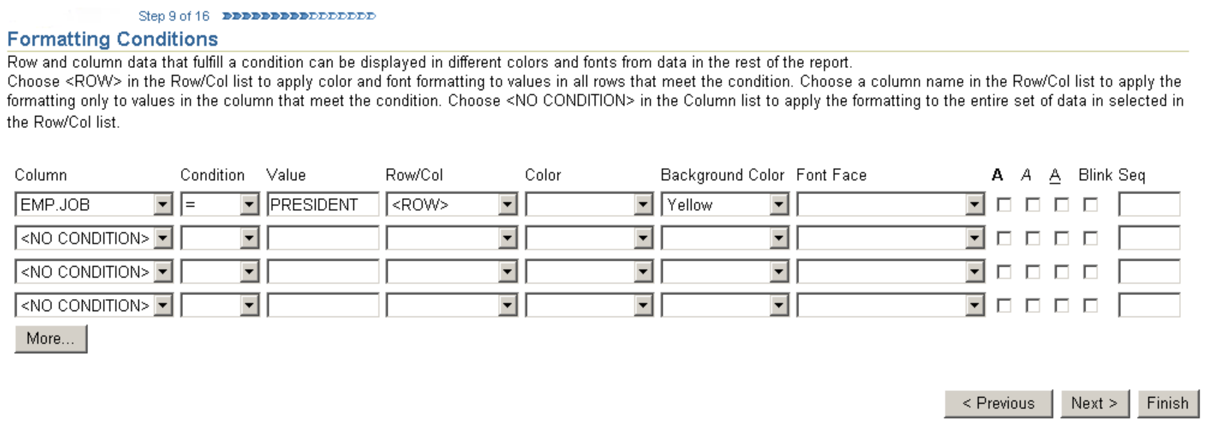
The quotation marks in Table A–10 will not appear in your output. They are used here to make it clear where there are leading spaces. See the Oracle Application Server documentation for more information on format masks. You'll find it on the Oracle Technology Network, `http://www.oracle.com/technology/index.html`.

19. Click **Next**.

20. Optionally, specify conditions and define special formatting for displayed report data that will be used whenever those conditions are met (Figure A–33).

    For example, specify that the background color of a particular row should become yellow whenever the JOB column equals PRESIDENT.

Figure A–33 The Formatting Conditions Page



Step 9 of 16

Formatting Conditions

Row and column data that fulfill a condition can be displayed in different colors and fonts from data in the rest of the report.
Choose <ROW> in the Row/Col list to apply color and font formatting to values in all rows that meet the condition. Choose a column name in the Row/Col list to apply the formatting only to values in the column that meet the condition. Choose <NO CONDITION> in the Column list to apply the formatting to the entire set of data in selected in the Row/Col list.

Column	Condition	Value	Row/Col	Color	Background Color	Font Face	A	A	A	Blink	Seq
EMP.JOB	=	PRESIDENT	<ROW>		Yellow		☐	☐	☐	☐	
<NO CONDITION>							☐	☐	☐	☐	
<NO CONDITION>							☐	☐	☐	☐	
<NO CONDITION>							☐	☐	☐	☐	

Table A–11 lists and describes the options on this page.

Table A–11 Options on the Formatting Conditions Page of the Report Wizard

Option	Description
Column	Choose a column on which the condition will be applied. The column must be one of those selected for display in an earlier wizard step.

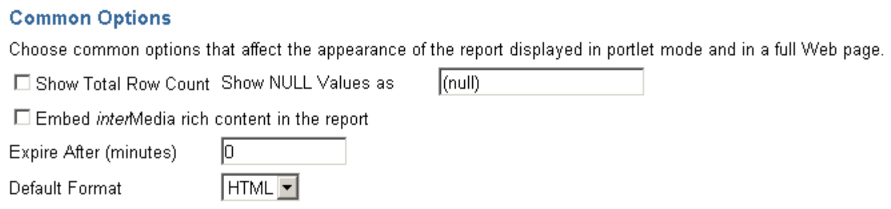
Table A–11 (Cont.) Options on the Formatting Conditions Page of the Report Wizard

Option	Description
Condition	Choose conditions that will define **Column**'s relationship to **Value**. Choose from: ■ equal to (=) ■ greater than (>) ■ greater than or equal to (>=) ■ less than (<) ■ less than or equal to (<=) ■ like ■ not equal to (!=) ■ null ■ not null ■ in ■ not in
Value	Enter a column value to trigger the condition. There is no need to apply formatting masks to numeric values. For example, in a table with an `EMPNO` column, if you want to apply special formatting to EMPNO values greater than 3000, select > for the **Condition**, and enter `3000` in this field. To specify multiple values after an `IN` or `NOT IN` condition, type a colon (:) between each value, for example, 10:20:30.
Row/Col	Choose whether to apply the conditional formatting to an entire row or to just the affected column data. If you specify a condition, you must choose **<Row>** or a column name here.
Color	Choose a text color for displaying data that meets the specified condition.
Background Color	Choose a background color for the column or row that contains data that meets the specified condition.
Font Face	Choose a font in which to display data that meets the specified condition.
A	Select to display results that meet the specified condition in bold.
A	Select to display results that meet the specified condition in italic.
<u>A</u>	Select to underscore results that meet the specified condition.
Blink	Select to cause results that meet the specified condition to blink on and off.
Seq	Enter a sequential number to establish an order of precedence for execution of the formatting. This is useful, for example, when a row contains data that meets more than one condition, and the formatting rules differ for each. The condition with the highest position in the order (that is, 1), takes precedence over all the others.

21. Click **Next**.
22. Define display options for the various possible views of the report:
    - Table A–12 describes display options common to all views (Figure A–34).

- Table A–13 describes display options for full-page views (Figure A–35).
- Options for portlet views are the same as those for full-page views, with a few noted exceptions. See Table A–13 and Figure A–35.
- Table A–14 describes display options for column breaks (Figure A–36).
- Table A–15 describes display options for row order (Figure A–37).
- Table A–16 describes display options for mobile devices (Figure A–38).

Figure A–34 Display Options Common to All Views


Common Options

Choose common options that affect the appearance of the report displayed in portlet mode and in a full Web page.

☐ Show Total Row Count Show NULL Values as (null)

☐ Embed *inter*Media rich content in the report

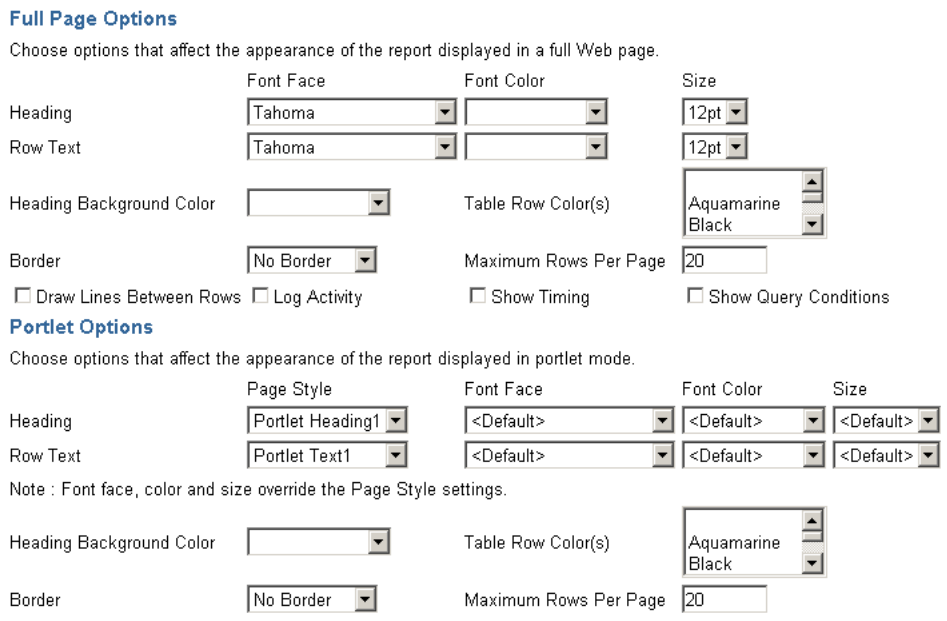
Expire After (minutes) 0

Default Format HTML

Table A–12 Report Display Options Common to All Views

Option	**Description**
Show Total Row Count	Select to display the total number of rows in the report.
Show Null Values as	Enter the text string you want to display for all null values in the report, for example, `(null)`.
Embed *inter*Media rich content in the report	Select to generate report output that shows any *inter*Media data values, such as audio, video, or images. For more information, see "Building Forms and Reports against interMedia Rich Content".
Expire After (minutes)	Enter the number of minutes after the initial display to keep the report in the cache. When the time expires, the report data is refreshed from the database.
Default Format	Choose a display format for the report: ■ **HTML**—Formats the report using HTML tables, and displays output on a new page in the Web browser. Portlets that contain large amounts of data may take longer to display in this format. ■ **Excel**—Formats the report for display in Microsoft Excel. When running the finished report, your local Excel settings may make it necessary for you to first save the report locally before it can be opened and viewed. ■ **ASCII**—Formats the report using the HTML PRE tag to display heading and values in the report as ASCII text. This option is useful for displaying large amounts of data.

Figure A–35 Report Display Options for Full Web-Page and Portlet Views


Full Page Options

Choose options that affect the appearance of the report displayed in a full Web page.

	Font Face	Font Color	Size
Heading	Tahoma		12pt
Row Text	Tahoma		12pt

Heading Background Color		Table Row Color(s)	Aquamarine Black
Border	No Border	Maximum Rows Per Page	20

☐ Draw Lines Between Rows ☐ Log Activity ☐ Show Timing ☐ Show Query Conditions

Portlet Options

Choose options that affect the appearance of the report displayed in portlet mode.

	Page Style	Font Face	Font Color	Size
Heading	Portlet Heading1	<Default>	<Default>	<Default>
Row Text	Portlet Text1	<Default>	<Default>	<Default>

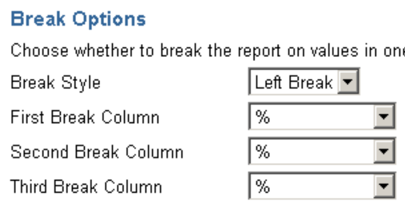
Note : Font face, color and size override the Page Style settings.

Heading Background Color		Table Row Color(s)	Aquamarine Black
Border	No Border	Maximum Rows Per Page	20

Table A–13 Report Display Options for Full Web-Page and Portlet Views

Option	Description
Font Face	Specify a font for report text.
Font Size	Specify a font size for report text. Use relative values, such as +1, +2, and so on. Relative font size is the last font size specified in the HTML code for the page plus the relative value. For example, the last font size specified in HTML code is 14 points; a new heading is coded at +2 font size; the heading will display in 16-point type.
Border	Specify the thickness of border to display around the report. Choose **No Border**, **Thin Border**, or **Thick Border**.
Font Color	Specify a font color for report text.
Heading Background Color	Specify a background color for report column headings.
Table Row Color(s)	Specify a background color for report rows. To choose more than one color use CTRL-Click (Windows). Rows will display in alternating colors.
Maximum Rows Per Page	Enter the maximum number of rows you want to display in the report.
Draw Lines Between Rows	Select to display lines between report rows.
Log Activity	(Full page only.) Check to enter performance information and the names of users who request the report in the OracleAS Portal activity log.
Show Timing	(Full page only.) Check to display report timing at the bottom of the report. Timing runs from when the server receives the request to generate the report to when the HTML for the report is generated.
Show Query Conditions	(Full page only.) Check to display user-specified parameters that are passed to the query that created the report and the time when the report was created.

Figure A–36 Report Display Options for Column Breaks


Break Options

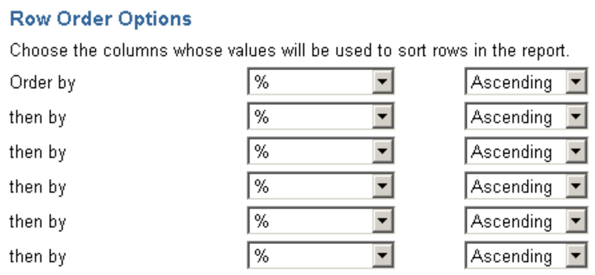
Choose whether to break the report on values in one, two or three column vlaues.

Break Style	Left Break
First Break Column	%
Second Break Column	%
Third Break Column	%

Table A–14 Report Display Options for Column Breaks

Option	Description
Break Style	Select a break style, and choose whether to break the report on values from one, two, or three columns.
First Break Column	Select the first column on which to break. For example, if you choose DEPTNO, the report will show all the rows associated with the first department number, followed by all the rows associated with the next department number, and so on. If you specify break columns, then you must also specify **Order by** values for the columns in the same order (see Table A–15).
Second Break Column	Select the second column on which to break. If you do not require a second level break, choose %.
Third Break Column	Select the third column on which to break. If you do not require a third level break, choose %.

Figure A–37 Report Display Options for Row Order


Row Order Options

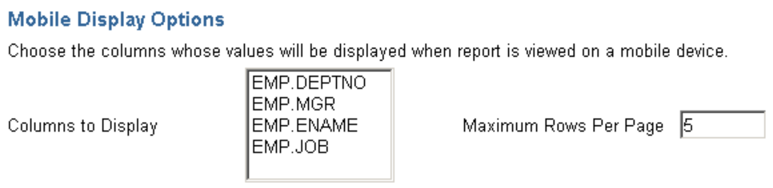
Choose the columns whose values will be used to sort rows in the report.

Order by	%	Ascending
then by	%	Ascending
then by	%	Ascending
then by	%	Ascending
then by	%	Ascending
then by	%	Ascending

Table A–15 Report Display Options for Row Order

Option	Description
Order by	Select the table or view column whose values will be used to sort rows in the report. Choosing this option is equivalent to specifying a SQL ORDER BY clause. Choose **Ascending** to sort A to Z or 0 to 9. Choose **Descending** to sort Z to A or 9 to 0.
then by	Select additional columns whose values will be used to sort report rows. For example, if you choose **Order by** Department ID, **then by** Employee Name, Oracle Portal sorts report rows numerically using department IDs. Rows containing the same Department ID are then sorted alphabetically using employee names.

Figure A–38 Report Display Options for Mobile Devices



Mobile Display Options

Choose the columns whose values will be displayed when report is viewed on a mobile device.

Columns to Display: EMP.DEPTNO, EMP.MGR, EMP.ENAME, EMP.JOB

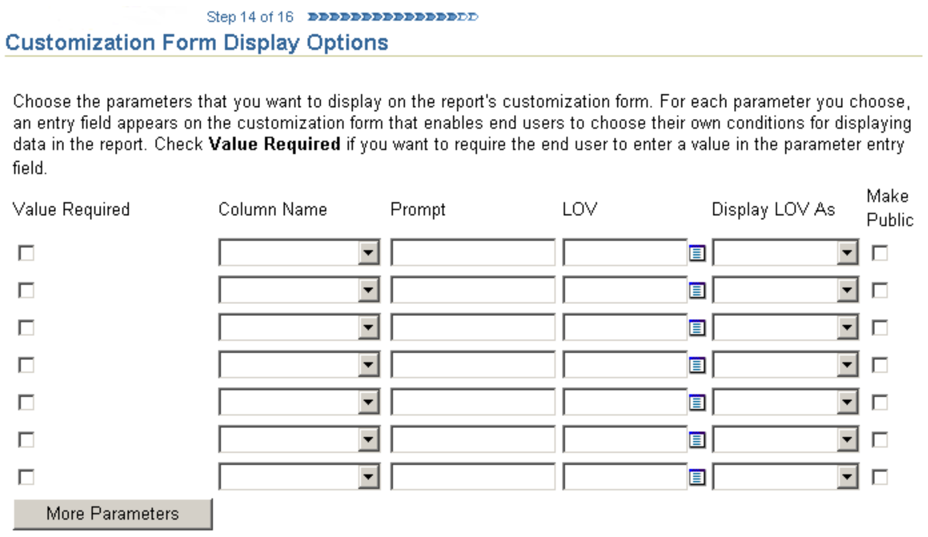
Maximum Rows Per Page: 5

Table A–16 Report Display Options for Mobile Devices

Option	Description
Columns to Display	Select the columns you want to display on a mobile device. Use this feature to reduce the size of the report to better fit onto the smaller screens available on mobile devices.
Maximum Rows Per Page	Enter the maximum number of rows you want to display on a single screen.

23. Click **Next**.
24. In the **Customization Form Display Options** section, choose the parameters that you want to display on the report's customization form (Figure A–39).

Figure A–39 Detail of the Customization Form Display Options Page



Step 14 of 16

Customization Form Display Options

Choose the parameters that you want to display on the report's customization form. For each parameter you choose, an entry field appears on the customization form that enables end users to choose their own conditions for displaying data in the report. Check **Value Required** if you want to require the end user to enter a value in the parameter entry field.

For each parameter you choose, an entry field appears on the customization form that enables users to choose their own conditions for displaying data in the report. Check **Value Required** if you want to require the user to enter a value in the parameter entry field. If you wish, you can use bind variables to allow users to specify their own amounts. See "Using Bind Variables" for more information.

Table A–17 lists and describes the display options for the report customization form.

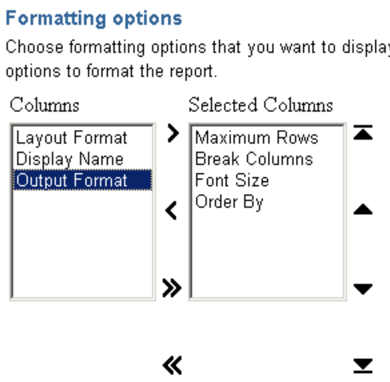
Table A–17 Display Options for the Customization Form Display Options Page

Option	Description
Value Required	Select to require that the user specify a value for the column's entry field on the report customization form. If you do not check this box, the user will not be required to specify a value.
Column Name	Select a table or view column. An entry field will be added to the report customization form. Users can specify values that limit the data displayed in the report. Entry fields appear only for the columns you select here.
Prompt	Enter the prompt text you want to display next to the entry field. The prompt text tells users what to enter in the field. For example: `Enter a Department Number:`
LOV	Select a list of values (LOV) for the column's entry field. Users of the customization form can choose values from this list to limit the data displayed in the report. Click the Browse icon next to this field to locate an LOV. Note that the LOV must already be built and stored in the Portal Repository. For more information, see "Building Lists of Values Declaratively". In the **LOV** field, you can use a constant for the application name (for example, enter #APP_NAME#.dept_lov). For more information, see "Referencing the OracleAS Portal Schema".
Display LOV As	Select a format for displaying your list of values. Choose from: ▪ **<Blank>**—No option is selected. ▪ **Check box**—Each selection displays with a check box next to it. Users check or do not check the box to indicate their preferred value. ▪ **Combo box**—Combines a pop-up list and a manual entry field. Users can either select a provided value or enter their own. ▪ **Pop up**—Provides a pop-up list from which users select a value. ▪ **Radio group**—Lists each selection with a radio button next to it. Users select or do not select the button to indicate their preferred value. ▪ **Multiple Select**—Allow users to choose more than one value.
Make Public	Click to allow values to be made public, that is, to allow non-authenticated users to see them. Be sure to make public any element that will be a variable in the list of portlet parameters associated with a page parameter.
More Parameters	Click to display more fields on this page to include more table or view columns on the report customization form.

25. In the **Formatting options** section, choose formatting options to include on the report customization form (Figure A–40).

    Including one or more of these selections enables users to have a greater level of control over the final appearance of the report.

Figure A–40 The Formatting Options Section of the Customization Form Display Options Page

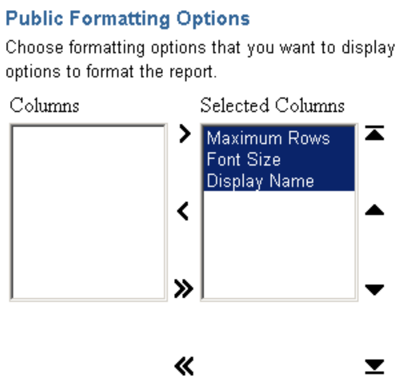

Formatting options

Choose formatting options that you want to display on the report's customization form. End users can choose these options to format the report.

Columns	Selected Columns
Layout Format	Maximum Rows
Display Name	Break Columns
Output Format	Font Size
	Order By

26. In the **Public Formatting Options** section, choose formatting options to include on the parameter tab for the page where the report is displayed (Figure A–41).

Figure A–41 The Public Formatting Options Section of the Customization Form Display Options Page


Public Formatting Options

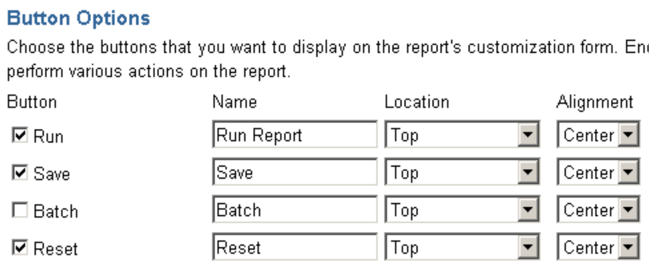
Choose formatting options that you want to display on the page's parameter tab. End users can choose these options to format the report.

Columns	Selected Columns
	Maximum Rows
	Font Size
	Display Name

27. In the **Button Options** section, select the buttons that will display on the report customization form (Figure A–42).

    Table A–18 lists and describes button display options for the report customization form.

Figure A–42 The Button Options Section of the Customization Form Display Options Page


Button Options

Choose the buttons that you want to display on the report's customization form. End users can click the buttons to perform various actions on the report.

Button	Name	Location	Alignment
☑ Run	Run Report	Top	Center
☑ Save	Save	Top	Center
☐ Batch	Batch	Top	Center
☑ Reset	Reset	Top	Center

Table A–18 Button Display Options for the Customization Form Display Options Page

Option	Description
Button	Select one or more buttons to display on the report customization form: ▪ **Run**—Displays the URL portlet with the options the user has specified in the customization form. ▪ **Save**—Saves users' current selections to enable them to rerun a report using the same parameters without having to reenter them into the report customization form. When you include this button, the report customization form also displays a **Reset to Defaults** button that resets all values in the report customization form to their default values. ▪ **Batch**—Runs the URL portlet in batch mode and saves the results in the database. ▪ **Reset**—Undoes any changes made to the report customization form since the last Save, Reset, or Reset to Defaults event.
Name	Enter the label that will display on the selected button. Keep the name short to avoid displaying large buttons. You cannot change the label on the **Reset to Defaults** button.
Location	Specify the location of the button on the report customization form. Choose from **Top**, **Top and Bottom**, or **Bottom**.
Alignment	Specify the alignment of the button on the report customization form. Choose from **Left**, **Center**, or **Right**.

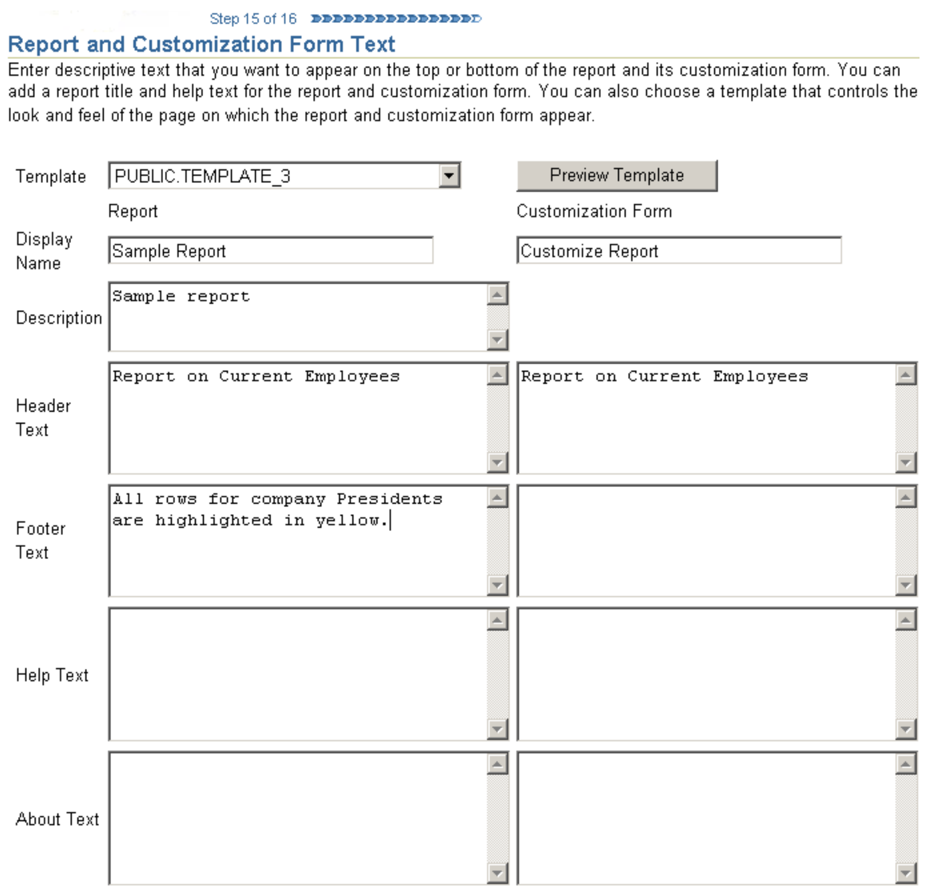
28. Click **Next**.
29. Optionally, select a template to set the look and feel of the report and the report customization form when they are displayed on a full Web page.

    The template you choose here applies to both the report and the report customization form when they are displayed on a full Web page. The template is not used when the report and report customization form are displayed as portlets; in this case, the style of the page is used.

    The template controls the look and feel of the page on which the report appears. In contrast, the display options you have specified on other pages in this wizard control the look and feel of the report itself.

    Click the **Preview** button to see how the template elements will display (Figure A–43).

Figure A–43 Report and Customization Form Text Page



Report and Customization Form Text

Enter descriptive text that you want to appear on the top or bottom of the report and its customization form. You can add a report title and help text for the report and customization form. You can also choose a template that controls the look and feel of the page on which the report and customization form appear.

	Report	Customization Form
Template	PUBLIC.TEMPLATE_3	Preview Template
Display Name	Sample Report	Customize Report
Description	Sample report	
Header Text	Report on Current Employees	Report on Current Employees
Footer Text	All rows for company Presidents are highlighted in yellow.	
Help Text		
About Text		

30. Optionally, enter text for display on the top or bottom of the report or the report customization form (Figure A–43).

Table A–19 lists and describes the options for including text. Enter text for the report in the left column. Enter text for the report customization form in the right column.

Table A–19 Options for Including Text on a Report or a Report Customization Form

Option	**Description**
Display Name	Edit the display name for the report and the report customization form. You can specify HTML in this field.
Description	(Report only.) Provide a description of the purpose of this report. This is an attribute that may or may not display, depending on which attributes have been selected for display in the region where the report or report portlet is placed.
Header Text	Enter any introductory text for display at the top of the report or report customization form. Header Text displays below the display name or title. You can specify HTML in this field.
Footer Text	Enter any text for display at the bottom of the report or report customization form.You can specify HTML in this field.

Table A–19 (Cont.) Options for Including Text on a Report or a Report Customization

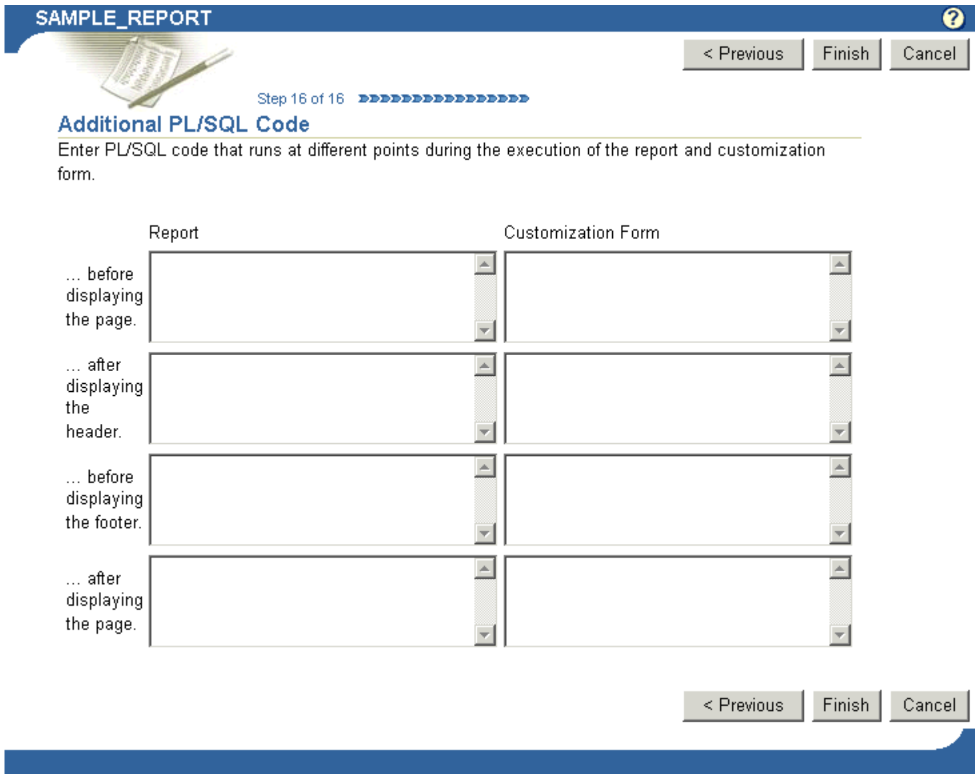
Option	Description
Help Text	Enter any text for inclusion on a help page for the report or report customization form. OracleAS Portal automatically includes a help button with the report and the report customization form when they are displayed on a full page. When they are displayed as a portlet, whether the help button displays is controlled by the way the host portlet region has been configured. Users click the help button to display the text you enter here. You can specify HTML in this field.
About Text	Enter any text for inclusion in an About box. OracleAS Portal automatically includes an About button with the report and the report customization form when they are displayed on a full page. When they are displayed as a portlet, whether the about button displays is controlled by the way the host portlet region has been configured. Users click the about button to display the text you enter here.

31. Click **Next**.
32. Optionally, enter PL/SQL code that runs during the execution of the report or report customization form HTML (Figure A–44).

    Table A–20 lists and describes the options available for running PL/SQL code at different points during the execution of the HTML code that creates the report or the report customization form.

    Note: In the PL/SQL you enter into these fields, you can use constants for the OracleAS Portal schema, the application schema, and the application name. For more information, see "Referencing the OracleAS Portal Schema".

Figure A–44 Additional PL/SQL Code Page



Additional PL/SQL Code

Enter PL/SQL code that runs at different points during the execution of the report and customization form.

	Report	Customization Form
... before displaying the page.		
... after displaying the header.		
... before displaying the footer.		
... after displaying the page.		

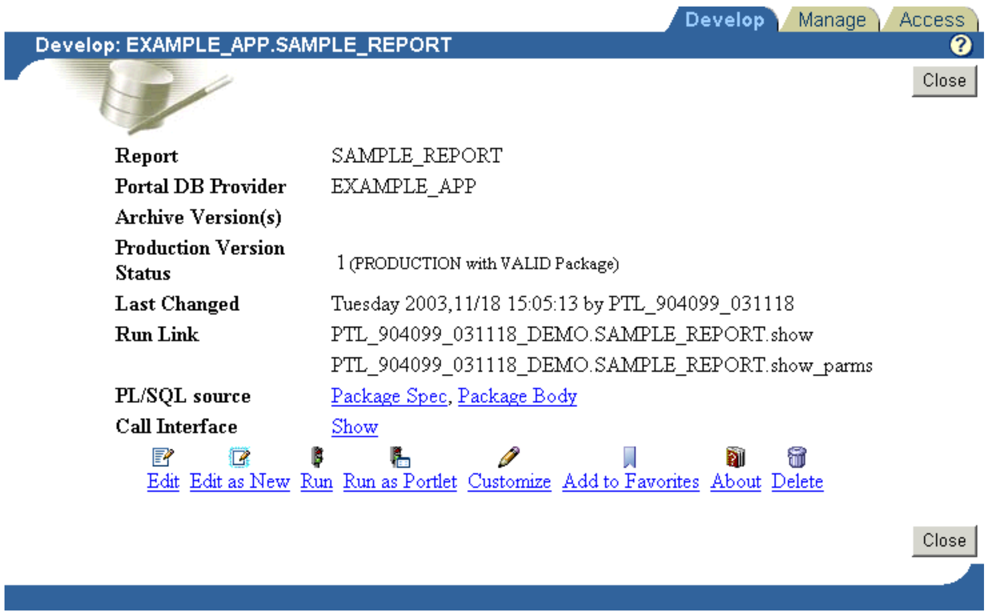
Table A–20 Options for Executing PL/SQL Code During Report Publication

Option	Description
… before displaying the page	Enter a PL/SQL procedure that will execute before the page containing the report or the report customization form displays.
… after displaying the header	Enter a PL/SQL procedure that will execute after the report or the report customization form header displays.
… before displaying the footer	Enter a PL/SQL procedure that will execute before the report or the report customization form footer displays.
… after displaying the page	Enter a PL/SQL procedure that will execute after the page containing the report or the report customization form displays.

33. Click **Finish**.

This saves the report and takes you to a summary page where you can manage and test run the report (Figure A–45). Subsequent sections in this appendix provide more detail about these tasks.

Figure A–45 The Report Summary Page



Report	SAMPLE_REPORT
Portal DB Provider	EXAMPLE_APP
Archive Version(s)	
Production Version Status	1 (PRODUCTION with VALID Package)
Last Changed	Tuesday 2003,11/18 15:05:13 by PTL_904099_031118
Run Link	PTL_904099_031118_DEMO.SAMPLE_REPORT.show PTL_904099_031118_DEMO.SAMPLE_REPORT.show_parms
PL/SQL source	Package Spec, Package Body
Call Interface	Show

A.1.4.4 Building Forms and Reports against *inter*Media Rich Content

You can integrate *inter*Media rich content, such as such as images, audio, and video, into the reports and forms you build with OracleAS Portal. These objects are stored in the Oracle9*i* Database Server.

The database column types for the *inter*Media rich content you can display include:

- ORDIMAGE—This object type supports the storage, management, and manipulation of image data, such as GIF, JPEG, and BMP.
- ORDAUDIO—This object type supports the storage and management of audio data, such as MP3, AU, WAV, and MPEG
- ORDVIDEO—This object type supports the storage and management of video data such as REAL, QuickTime 3/4, AVI, and MPEG.

In addition to allowing the use of these *inter*Media rich content type columns in an OracleAS Portal report or form, object attributes can be displayed in join conditions, formatting, column conditions, and so on. For example, display a video clip's size or duration in a report, or set a report condition that specifies that only objects modified after a certain date will display.

Keep in mind that, although *inter*Media supports a variety of content types and formats, the browser being used to view this material must natively support the relevant MIME type or have an installed plug-in that displays rich content that is not typically supported on the Web. For example, most browsers can natively display GIF and JPEG images, but TIFF images are not displayed without an installed plug-in.

*inter*Media and Reports

Two OracleAS Portal Reports Wizards offer the opportunity to include *inter*Media rich content:

- Reports from Query Wizard
- Query By Example Report (QBE)

With QBE reports, object-type attributes will not display. Consequently, trying to provide a value for an *inter*Media-based column in a report customization form results in an error. This means that you cannot use a report customization form for a QBE report that runs against *inter*Media object-type columns.

You can display interMedia rich content in a report in one of two ways:

- Embedded (inline)

  The image, audio, or video runs in place within the report.

- As a linked icon

  When a user clicks the link, the content is displayed in a new browser window, or it may be handled by the associated source application. For example, it may display in a RealPlayer window.

These display options apply to all columns that contain *inter*Media content in the OracleAS Portal report. They can be selected on the **Display Options** tab in the OracleAS Portal Report Wizard. By default, *inter*Media rich content is represented by icons in a report. Users click an icon to view the actual content. If you plan to rely on the default, you do not need to do anything further. If you plan to embed interMedia rich content, select the **Embed interMedia rich content in the report** check box on the **Display Options** page in the Report Wizard.

For more information about the Report Wizard, see "Building Reports Declaratively".

***inter*Media and Forms**

Two OracleAS Portal Form Wizards offer the opportunity to include *inter*Media rich content in your form:

- Form based on table or view
- Master-detail form

With either of these methods, you must make sure that the underlying database object you are changing through your form is based on columns of type ORDIMAGE, ORDAUDIO, or ORDVIDEO.

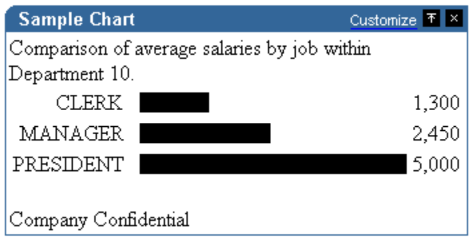
To include *inter*Media rich content in your form, on the **Formatting and Validation Options** page of the OracleAS Portal Form Wizard, select the column name in the left frame, and specify an **Item Type** of **File Upload (interMedia)** in the right frame.

Note: The assigning of column types is done outside of OracleAS Portal. Consult your database administrator if necessary to create a table with *inter*Media column types.

One advantage of uploading images, audio, and video clips into *inter*Media-based columns over uploading into BLOB columns is that the data is automatically parsed to extract several attributes, such as MIME type, length, and any user-defined metadata that might be included in the original media file. For example, a QuickTime file might contain close-captioning, or a RealVideo file might have copyright information. With *inter*Media-based columns, these attributes are automatically extracted and stored in the *inter*Media object for indexing and querying.

A.1.4.5 Building Charts Declaratively

Figure A–46 A Chart Built with the Portlet Builder Chart Wizard



Sample Chart

Customize

Comparison of average salaries by job within Department 10.

CLERK	1,300
MANAGER	2,450
PRESIDENT	5,000

Company Confidential

Charts display data from database tables or views as bar charts. They are based on at least two table or view columns:

- Values in the **Label** column provide a display name for the bars on the chart.
- Values in the **Values** column calculate the size of the bars on the chart relative to one another. Value columns must always contain numeric data.

You can also specify a Link column. Values in this column create hypertext links from the chart's labels to other OracleAS Portal database portlets or URLs. For example, with a link in place, users can click the label in a chart and jump to a form that enables them to update the data the chart is based on.

Charts are also an effective way to display aggregate data. You can use **Group By** and **Summary Options** in the wizard to sum the column values returned by your query, for example, the minimum, maximum, or average value, or the number of values in a column.

The portlet-building wizards in OracleAS Portal provide two construction methods for charts:

- Charts From Query Wizard
- Charts From SQL Query

The Query Wizard provides a greater level of assistance in constructing a chart. The SQL Query method provides you with a greater level of control over the query that will be used to fetch data for the chart.

This section describes how to build a chart using the Query Wizard.

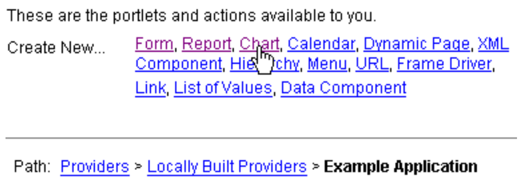
To create a chart using the Query Wizard:

1. Follow the instructions detailed in "Building Portlets Declaratively".

   Return to this section once you complete step 6.

2. Click the **Chart** link next to **Create New...** (Figure A–47).

Figure A–47 The Chart Link Next to Create New ...



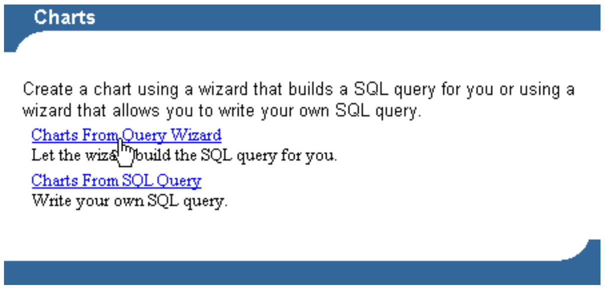
These are the portlets and actions available to you.

Create New... Form, Report, Chart, Calendar, Dynamic Page, XML Component, Hierarchy, Menu, URL, Frame Driver, Link, List of Values, Data Component

Path: Providers > Locally Built Providers > **Example Application**

3. On the Charts page, click **Charts From Query Wizard** (Figure A–48).

Figure A–48 The Charts From Query Wizard Link



Charts

Create a chart using a wizard that builds a SQL query for you or using a wizard that allows you to write your own SQL query.

Charts From Query Wizard
Let the wizard build the SQL query for you.

Charts From SQL Query
Write your own SQL query.

4. On the resulting page enter an internal name for the chart in the **Name** field (Figure A–49).

   The internal name is not published to users.

Figure A–49 The First Step of the Chart Wizard

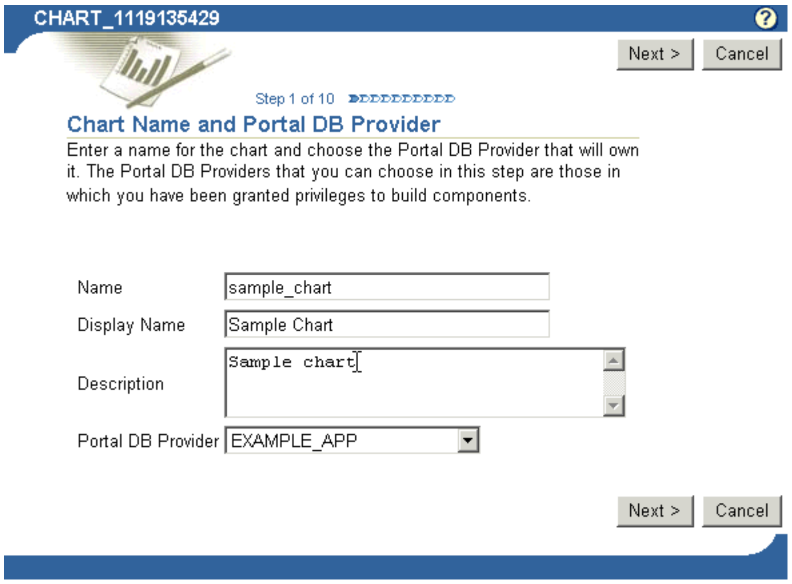


CHART_1119135429

Chart Name and Portal DB Provider

Enter a name for the chart and choose the Portal DB Provider that will own it. The Portal DB Providers that you can choose in this step are those in which you have been granted privileges to build components.

Name	sample_chart
Display Name	Sample Chart
Description	Sample chart
Portal DB Provider	EXAMPLE_APP

5. In the **Display Name** field, enter the name by which users will identify this chart (Figure A–49).

   The display name is used at runtime. It displays as the title of the browser window as well as in the banner of the chart. When the chart is published as a portlet, the display name is used as the title of the portlet and in the List of Available Portlets in the Portlet Repository.

6. In the **Description** field, enter a description of this chart (Figure A–49).

   The description displays below the portlet in the Portlet Repository. It can be a summary of the portlet's purpose, a classification of its type, or any other descriptive information. It may be useful within your organization to have a standard approach to what is included in the description.

7. From the list of **Portal DB Providers**, select the provider that will own this chart (Figure A–49).

   This list displays only those providers in which you have been granted privileges to build portlets. We recommend that you use a provider specifically created for owning, or hosting, your declaratively-built portlets.

8. Click **Next**.

9. Click the **List** icon next to the **Table or View** field, and choose a table or view on which to base your chart (Figure A–50).

Note: In the **Tables/Views** field, you can use constants for the OracleAS Portal schema and the application schema (for example, enter #APP_SCHEMA#.EMP). For more information, see "Referencing the OracleAS Portal Schema".

Figure A–50 Clicking the List Icon to Select a Table or View

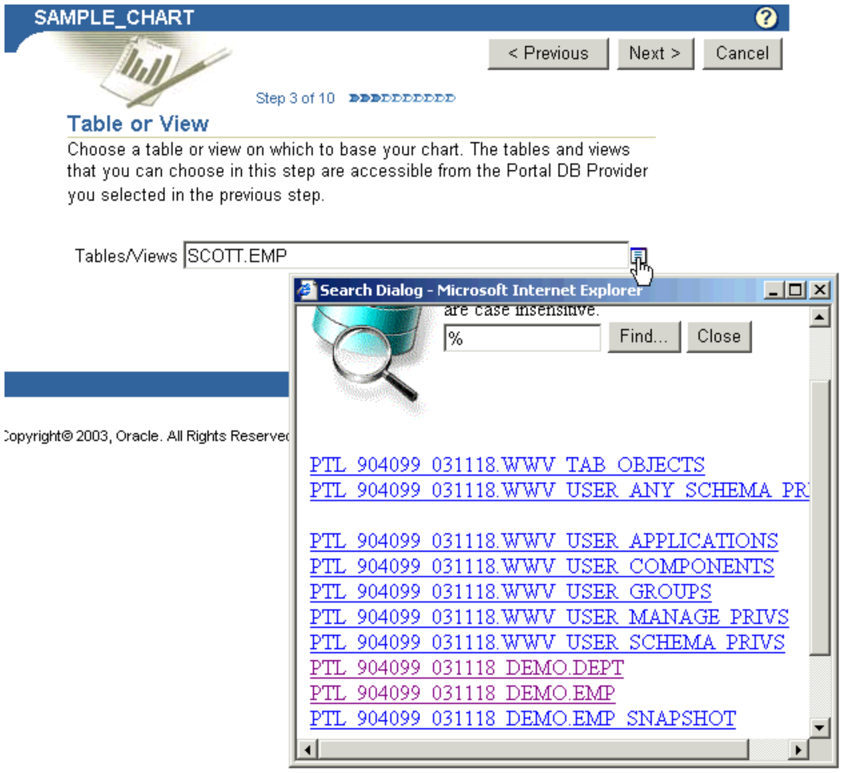


Table or View

Choose a table or view on which to base your chart. The tables and views that you can choose in this step are accessible from the Portal DB Provider you selected in the previous step.

Tables/Views SCOTT.EMP

are case insensitive.

%

Copyright© 2003, Oracle. All Rights Reserved

PTL_904099_031118.WWV_TAB_OBJECTS
PTL_904099_031118.WWV_USER_ANY_SCHEMA_PR
PTL_904099_031118.WWV_USER_APPLICATIONS
PTL_904099_031118.WWV_USER_COMPONENTS
PTL_904099_031118.WWV_USER_GROUPS
PTL_904099_031118.WWV_USER_MANAGE_PRIVS
PTL_904099_031118.WWV_USER_SCHEMA_PRIVS
PTL_904099_031118_DEMO.DEPT
PTL_904099_031118_DEMO.EMP
PTL_904099_031118_DEMO.EMP_SNAPSHOT

The tables and views that you can choose in this step are those that are accessible from the Portal DB Provider you selected in the previous step. Additionally, the list is limited to those tables and views on which you have the following privileges: all table views in the provider schema, granted select to public, granted select to the provider schema.

If clicking the **List** icon produces a "No response from Server" message, and you know the table you want to select, try entering the table name directly into the field.

Note: You can choose only one table or view using this wizard. If you want to base the chart on more than one table or view, you must write your own SQL to create the chart using the **Charts from SQL Query** wizard.

---

10. Click **Next**.

11. Choose the column(s) that contain the values you will display as Labels, Links, and Values in the chart (Figure A–51).

Figure A–51 Columns to Be Used for Labels, Links, and Values in the Chart

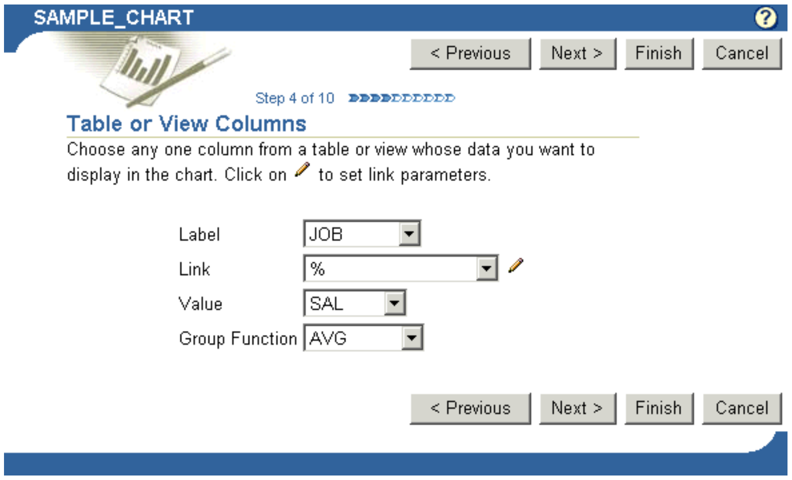


Table or View Columns

Choose any one column from a table or view whose data you want to display in the chart. Click on ✎ to set link parameters.

Label	JOB
Link	%
Value	SAL
Group Function	AVG

Table A–21 lists and describes the options on this page.

Table A–21 Data Selection Options for Charts

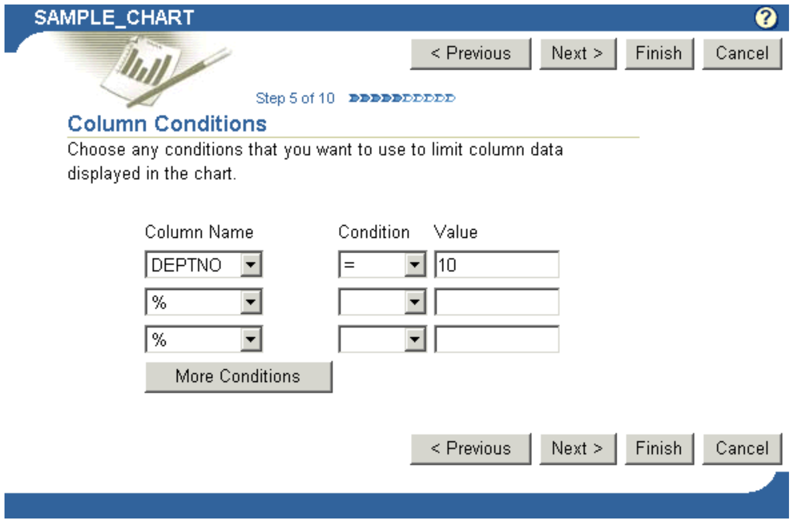
Option	Description
Label	Choose the column that contains the values you will use as labels for different bars on the chart. For example, if you were going to create a chart that compared salaries of each job title in the department, you might choose the JOB column as the **Label**, `SAL` as the **Value**, and AVG as the **Group Function**.
Link	Links must be preconstructed and stored in the database before you can use them here. Use this field to choose a preconstructed link that jumps from a chart bar's label to another OracleAS Portal portlet or URL. Click the Edit icon next to the **Link** field to set link parameters. If you do not set link parameters, the default values of the link are used.
Value	Choose the column that contains the values to be used to calculate the relative size of the bars in the chart. Value columns always contain numeric data. The default for **Value** is `1`. Specifying a value of `1` is useful if you also choose a group function. For example, you can choose the `JOB` column from `SCOTT.EMP` as the **Label**, `1` as the **Value**, and `COUNT` as the **Group Function**. This creates a chart that displays the number of employees in each job classification.
Group Function	A group function calculates a single summary value (%, sum, count, average, maximum, minimum, standard deviation, or variance) from groups of numeric values in the **Value** column. OracleAS Portal uses values in the **Label** column to determine these groupings. For example, you can choose the `JOB` column from `SCOTT.EMP` as the **Label**, `SAL` as the **Value**, and `AVG` as the **Group Function**. This creates a chart that displays the average salary for each job classification.

12. Click **Next**.

13. Optionally, specify conditions that limit the data displayed in the chart.

For example, to display data from all employees in Department 10, choose `DEPTNO` from **Column Name**, choose = or `like` from **Condition**, and enter `10` in the **Value** field (Figure A–52).

Figure A–52 Defining Column Conditions for a Chart



Column Conditions

Choose any conditions that you want to use to limit column data displayed in the chart.

Column Name	Condition	Value
DEPTNO	=	10
%		
%		

Table A–22 lists and describes the options that appear on the Column Conditions page.

Table A–22 Options on the Column Conditions Page

Option	**Description**
Column Name	Choose columns whose values will be used to limit the data displayed in the chart. For example, if you want to display values greater than 3000 from the `EMPNO` column of the `SCOTT.EMP` table, choose `EMPNO` as the **Column Name**, choose > from **Condition**, and enter `3000` in the **Value** field.
Condition	Choose conditions that will define **Column Name**'s relationship to **Value**. Choose from: ▪ equal to (=) ▪ greater than (>) ▪ greater than or equal to (>=) ▪ less than (<) ▪ less than or equal to (<=) ▪ like ▪ not equal to (!=) ▪ null ▪ not null ▪ in ▪ not in To specify multiple values after an `IN` or `NOT IN` condition, type a colon (:) between each value, for example, 10:20:30.
Value	Enter a value that will limit the data displayed in the chart.

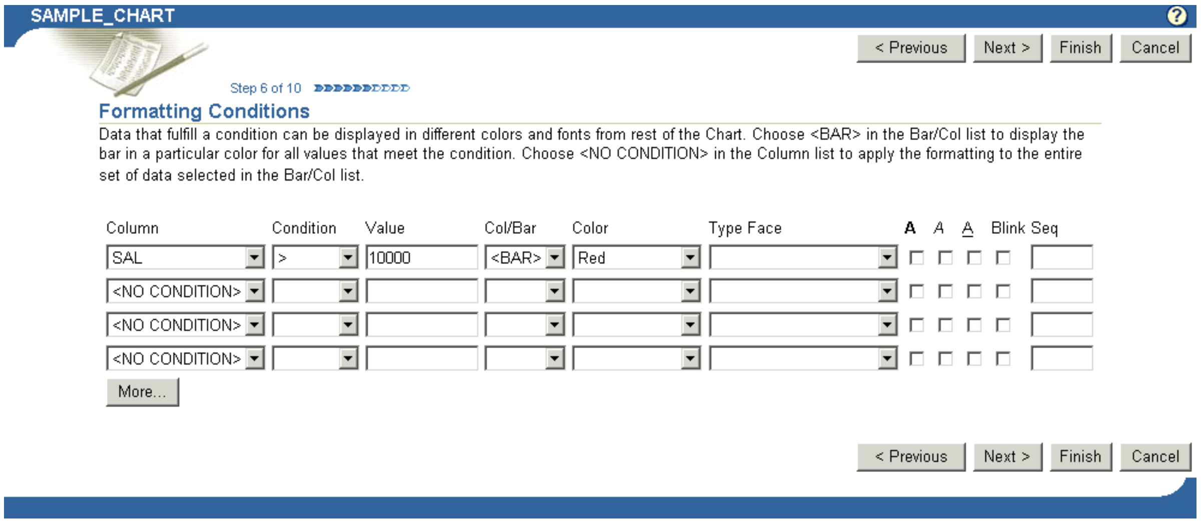
Table A–22 (Cont.) Options on the Column Conditions Page

Option	Description
More Conditions	Click to add more fields for specifying additional conditions.

14. Click **Next**.
15. Optionally, specify conditions and define special formatting for chart data that will be used whenever those conditions are met.

    For example, you can specify that a chart bar should be red its related column value exceeds a particular amount (Figure A–53).

Figure A–53 Defining Formatting Conditions



Formatting Conditions

Data that fulfill a condition can be displayed in different colors and fonts from rest of the Chart. Choose <BAR> in the Bar/Col list to display the bar in a particular color for all values that meet the condition. Choose <NO CONDITION> in the Column list to apply the formatting to the entire set of data selected in the Bar/Col list.

Column	Condition	Value	Col/Bar	Color	Type Face	A	A	A	Blink	Seq
SAL	>	10000	<BAR>	Red		☐	☐	☐	☐	
<NO CONDITION>						☐	☐	☐	☐	
<NO CONDITION>						☐	☐	☐	☐	
<NO CONDITION>						☐	☐	☐	☐	

Table A–23 lists and describes the formatting condition options on this page.

Table A–23 Options on the Formatting Conditions Page of the Chart Wizard

Option	Description
Column	Choose a column on which the condition will be applied.
Condition	Choose conditions that will define **Column**'s relationship to **Value**. Choose from: ▪ equal to (=) ▪ greater than (>) ▪ greater than or equal to (>=) ▪ less than (<) ▪ less than or equal to (<=) ▪ like ▪ not equal to (!=) ▪ null ▪ not null ▪ in ▪ not in ▪ between

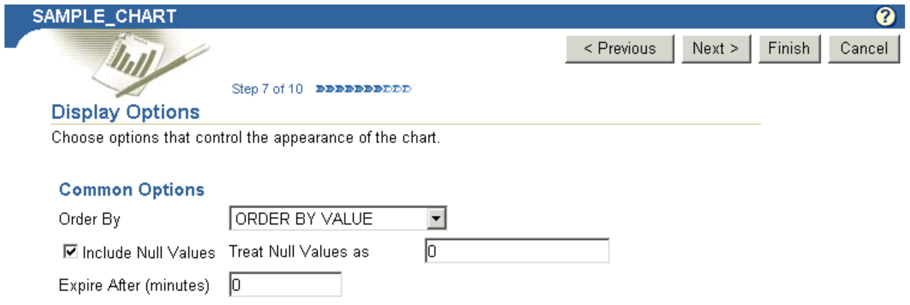
Table A–23 (Cont.) Options on the Formatting Conditions Page of the Chart Wizard

Option	Description
Value	Enter a column value to trigger the condition. There is no need to apply formatting masks to numeric values. For example, in a table with an `EMPNO` column, if you want to apply special formatting to EMPNO values greater than 3000, select > for the **Condition**, and enter `3000` in this field. Enter date values in nls_date_format. To specify multiple values after an `IN` or `NOT IN` condition, type a colon (:) between each value, for example, 10:20:30. To specify a BETWEEN condition, separate the two values by either a comma, a space, or the word *and*.
Col/Bar	Choose whether you want the conditional formatting to be applied to a displayed column name or to the `<BAR>` associated with value that meets the condition. For example, imagine that your chart uses the `SAL` and `ENAME` columns. `ENAME` is used to label each bar in the chart, and `SAL` is used to provide the value that drives the size of the bar. If you want a chart bar to turn red and blink if a salary is under 40,000, select `<BAR>`. If you want the `ENAME` to turn red and blink when the condition is met, select `ENAME`. If you want the salary, which is included in the chart along with the label and the bar, to turn red and blink when the condition is met, select `SAL`.
Color	Choose a color for displaying data that meets the specified condition. The color will be applied to text or to the bar, depending on the choice you made in the Col/Bar field.
Type Face	Choose a font in which to display data that meets the specified condition.
A	Select to display results that meet the specified condition in bold.
A	Select to display results that meet the specified condition in italic.
<u>A</u>	Select to underscore results that meet the specified condition.
Blink	Select to cause results that meet the specified condition to blink on and off.
Seq	Enter a sequential number to establish an order of precedence for execution of the formatting. This is useful, for example, when a row contains data that meets more than one condition, and the formatting rules differ for each. The condition with the highest position in the order (that is, 1), takes precedence over all the others.

16. Click **Next**.
17. Define display options for the various possible views of the chart:
    - Table A–24 describes display options common to all views of a chart (Figure A–54).
    - Table A–25 describes display options for full-page views of a chart (Figure A–55).
    - Table A–26 describes display options for charts displayed as portlets (Figure A–56).

- Table A–27 describes display options for charts displayed on mobile devices (Figure A–57).

Figure A–54 Display Options Common to All Chart Views



Display Options

Choose options that control the appearance of the chart.

Common Options

Order By: ORDER BY VALUE

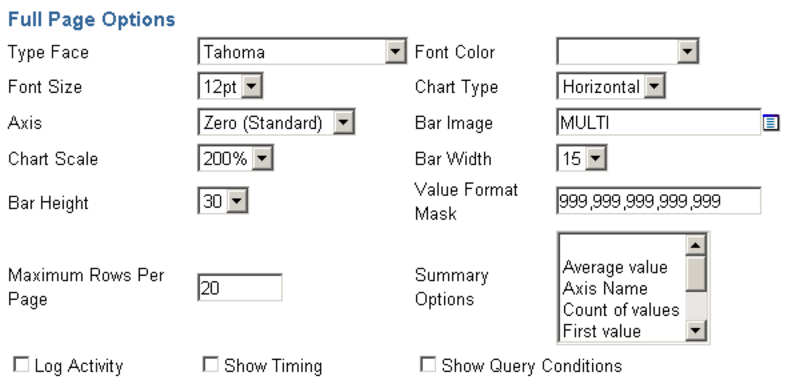
☑ Include Null Values Treat Null Values as: 0

Expire After (minutes): 0

Table A–24 Display Options Common to All Chart Views

Option	**Description**
Order By	Choose a method for ordering the chart's data: ■ **ORDER BY VALUE**—The bars in the chart are shown in the same numeric order as values in the table column that you specified in the **Value** entry field. ■ **ORDER BY VALUE DESC**—The bars in the chart are shown in reverse order of values in the table column that you specified in the **Value** entry field. ■ **ORDER BY LABEL**—The bars in the chart are shown in the same order as values in the table column that you specified in the **Label** entry field. ■ **ORDER BY LABEL DESC**—The bars in the chart are shown in the reverse order of values in the table column that you specified in the **Label** entry field.
Include Null Values	Select to display null values in the chart.
Treat Null Values as	Enter the text string you want to be displayed for any null values in the chart, for example: `(this field is null)`.
Expire After (minutes)	Enter the number of minutes after the initial display to keep the chart in the cache. When the time expires, the chart data is refreshed from the database.

Figure A–55 Display Options for Full Web Page Chart Views



Full Page Options

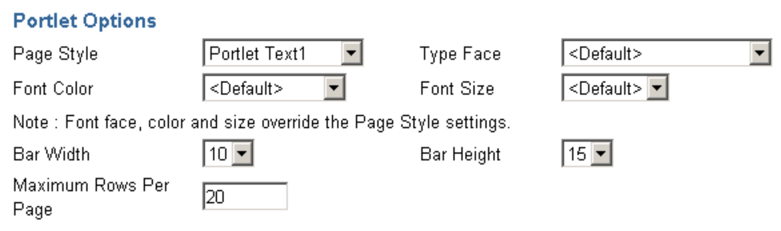
Type Face	Tahoma	Font Color	
Font Size	12pt	Chart Type	Horizontal
Axis	Zero (Standard)	Bar Image	MULTI
Chart Scale	200%	Bar Width	15
Bar Height	30	Value Format Mask	999,999,999,999,999
Maximum Rows Per Page	20	Summary Options	Average value Axis Name Count of values First value

☐ Log Activity ☐ Show Timing ☐ Show Query Conditions

Table A–25 Display Options Applying to Full Web-Page Chart Views

Option	Description
Type Face	Choose the font for displaying chart text.
Font Color	Choose the font color for displaying chart text. User-defined colors are not available for selection.
Font Size	Choose the font size for displaying chart text.
Chart Type	Choose whether to display the cart bars in a horizontal or vertical orientation.
Axis	Choose a method for displaying chart bars relative to the value of the chart's axis. For example, if you choose `Zero`, the value of the axis is set at 0. If you choose `Average Value`, the axis is set at the average of all values in the table or view column on which the **Value** section of the chart is based.
Bar Image	Choose an image that will be used to fill in the bars on the chart. Choose `MULTI` to display bars in different colors.
Chart Scale	Choose a percent (%) value to set the scale of chart bars relative to the Web page that will host the chart. Higher percentages display larger bars.
Bar Width	Choose a width in pixels for chart bars. This option applies to bars in both horizontally-and vertically-oriented charts.
Bar Height	Enter a height in pixels for chart bars. This option applies to bars in both horizontally-and vertically-oriented charts.
Value Format Mask	Enter a format for numeric values or dates that appear on the chart.
Maximum Rows Per Page	Enter the maximum number of bars you want to display per page.
Summary Options	Choose one or more options to display summary information about the chart. Each option you choose is included in the summary information box at the bottom of the chart. Windows users can choose more than one option by pressing the Ctrl key while clicking each option.
Log Activity	Select to enter information in the OracleAS Portal activity log. Information includes performance statistics and the names of users who request the chart.
Show Timing	Select to display the time when the server received the request to generate HTML for the chart. This information is displayed at the bottom of the chart.
Show Query Conditions	Select to display all user-specified parameters passed to the SQL query that created the chart and the time the chart was created. This information is displayed at the bottom of the chart.

Figure A–56 Display Options for Charts Viewed As Portlets



Portlet Options

Page Style	Portlet Text1	Type Face	<Default>
Font Color	<Default>	Font Size	<Default>

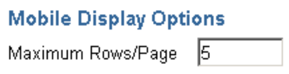
Note : Font face, color and size override the Page Style settings.

Bar Width	10	Bar Height	15
Maximum Rows Per Page	20		

Table A–26 Display Options for Charts Viewed as Portlets

Option	Description
Page Style	Choose the page style element to use for displaying chart text.
Type Face	Choose a font to override the font of the chosen page style element.
Font Color	Choose a font color to override the font color of the chosen page style element.
Font Size	Choose a font size to override the font size of the chosen page style element.
Bar Width	Choose a width in pixels for bars on the chart. This option applies to bars in both horizontally- and vertically-oriented charts.
Bar Height	Choose a height in pixels for bars on the chart. This option applies to bars in both horizontally- and vertically-oriented charts.
Maximum Rows Per Page	Enter the maximum number of bars you want to display in the chart.

Figure A–57 Display Options for Charts Viewed on Mobile Devices


Mobile Display Options

Maximum Rows/Page 5

Table A–27 Display Options for Charts Viewed on Mobile Devices

Option	Description
Maximum Rows Per Page	Enter the maximum number of bars you want to display in the chart.

18. Click **Next**.
19. Define the content for display on the chart customization form.

    Use the options on this page to give your users control over what data will appear in their view of the chart.

    For example, if you choose the `DEPTNO` column from the SCOTT.EMP table as a **Column Name** in this step, OracleAS Portal adds an entry field for `DEPTNO` to the chart's customization form. Users can type a department number in the field to limit display to data relevant to that department.

    Optionally, you can add a list of values to the entry files. Instead of requiring users to type a numeric value for `DEPTNO`, you could add a list of values that enables them to choose, for example, `10`, `20`, or `30`.

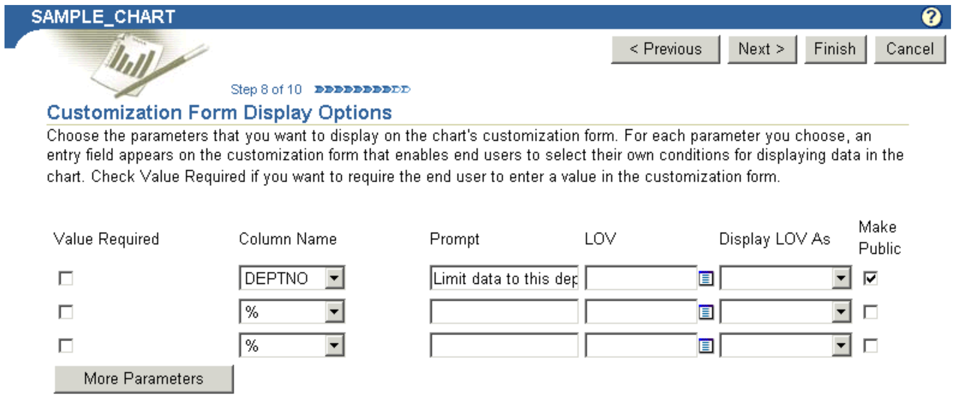
    In portlet implementations, the chart customization form displays when users click the **Customize** link at the top of the portlet. In full-page implementations, you can provide users with a link to run the chart; the link can target the chart customization form, which in turn can have a **Run** button. Users click this button to run the chart once they've set their customization options.

    If you wish, you can use bind variables to allow users to specify their own values in the customization form. See "Using Bind Variables" for more information.

    - Table A–28 describes chart customization form display options (Figure A–58).

- Table A–29 describes chart customization form formatting options (Figure A–59).
- Table A–30 describes chart customization form public formatting options (Figure A–60).
- Table A–31 describes chart customization form button options (Figure A–61).

Figure A–58 Chart Customization Form Display Options



Customization Form Display Options

Choose the parameters that you want to display on the chart's customization form. For each parameter you choose, an entry field appears on the customization form that enables end users to select their own conditions for displaying data in the chart. Check Value Required if you want to require the end user to enter a value in the customization form.

Value Required	Column Name	Prompt	LOV	Display LOV As	Make Public
☐	DEPTNO	Limit data to this dep			☑
☐	%				☐
☐	%				☐

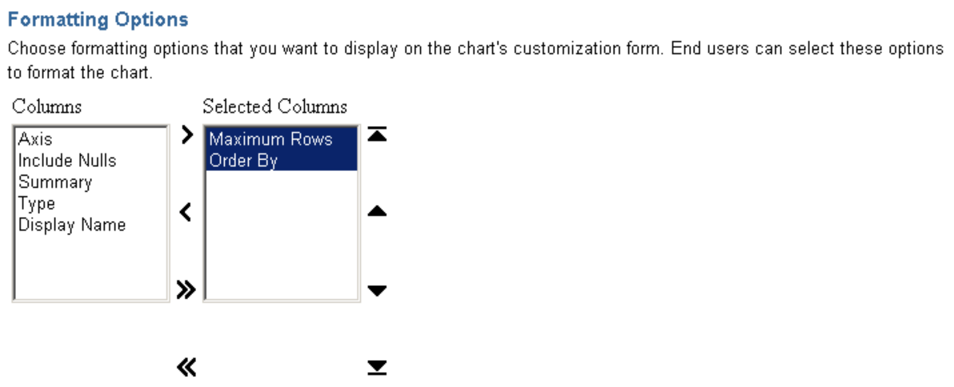
Table A–28 Chart Customization Form Display Options

Option	**Description**
Value Required	Select to require the user to specify a value for the column's entry field on the chart's customization form.
Column Name	Choose a table or view column for inclusion on the chart customization form. An entry field will display on the form that enables users to specify values that limit the data that will be displayed in the chart. If you do not choose a table or view column, no entry appears on the customization form.
Prompt	Enter prompt text to display next to the entry field. The prompt text tells users what to enter in the field, for example: `Choose the department number you want to display.`
LOV	Select a list of values (LOV) for the column's entry field. Users of the customization form can choose values from this list to limit the data displayed in the report. Click the Browse icon next to this field to locate an LOV. Note that the LOV must already be built and stored in the Portal Repository. For more information, see "Building Lists of Values Declaratively". In the **LOV** field, you can use a constant for the application name (for example, enter #APP_NAME#.dept_lov). For more information, see "Referencing the OracleAS Portal Schema".

Table A–28 (Cont.) Chart Customization Form Display Options

Option	Description
Display LOV As	Select a format for displaying your list of values. Choose from: ■ **<Blank>**—No option is selected. ■ **Check box**—Each selection displays with a check box next to it. Users check or do not check the box to indicate their preferred value. ■ **Combo box**—Combines a pop-up list and a manual entry field. Users can either select a provided value or enter their own. ■ **Pop up**—Provides a pop-up list from which users select a value. ■ **Radio group**—Lists each selection with a radio button next to it. Users select or do not select the button to indicate their preferred value. ■ **Multiple Select**—Allow users to choose more than one value.
Make Public	Check this field if you want this element to display on the chart customization form. Be sure to make public any element that will be a variable in the list of portlet parameters associated with a page parameter.
More Parameters	Click to add more entry fields on the customization form.

Figure A–59 Chart Customization Form Formatting Options


Formatting Options

Choose formatting options that you want to display on the chart's customization form. End users can select these options to format the chart.

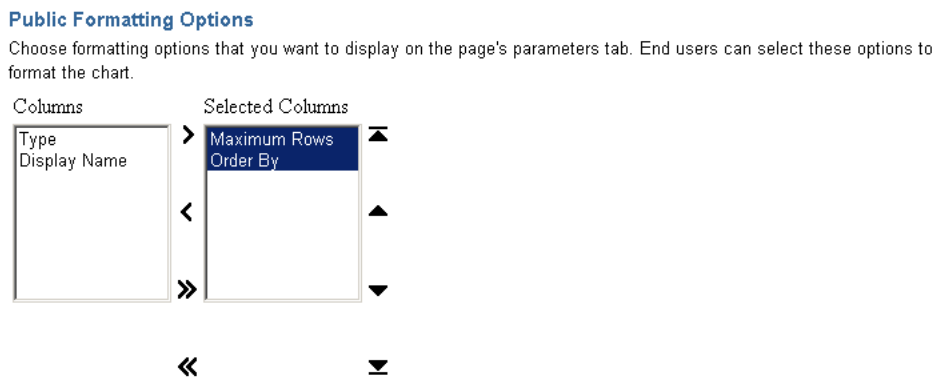
Table A–29 Chart Customization Form Formatting Options

Option	Description
Axis	Select to enable users of the chart customization form to choose a method for displaying chart bars relative to the value of the chart's axis. For example, if users choose `Zero`, the value of the axis is set at 0. If users choose `Average Value`, the axis is set at the average of all values in the table or view column on which the **Value** section of the chart is based.
Include Nulls	Select to enable users of the chart customization form to specify whether to display null values in the chart.
Maximum Rows/Page	Select to enable users of the chart customization form to specify the maximum number of bars to display in the chart.

Table A–29 (Cont.) Chart Customization Form Formatting Options

Option	Description
Summary	Select to enable users of the chart customization form to choose one or more options that display summary information about the chart. Each option that users choose is included in the summary information box at the bottom of the chart.
Type	Select to enable users of the chart customization form to choose a font for displaying chart text.
Display Name	Select to allow users of the chart customization form to edit the display name for the chart.
Order By	Select to enable users of the chart customization form to set the display order for chart data.

Figure A–60 Chart Customization Form Public Formatting Options



Public Formatting Options

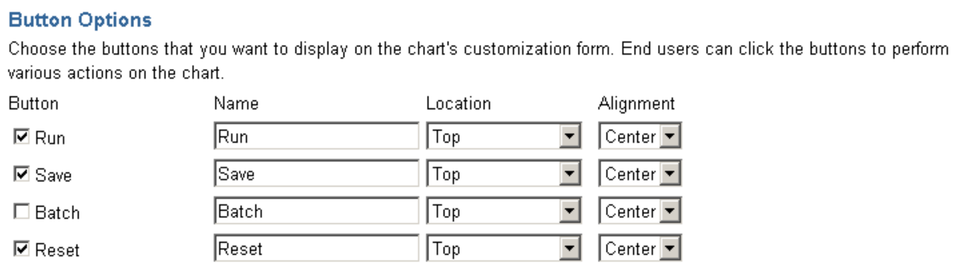
Choose formatting options that you want to display on the page's parameters tab. End users can select these options to format the chart.

Columns	Selected Columns
Type	Maximum Rows
Display Name	Order By

Table A–30 Chart Customization Form Public Formatting Options

Option	Description
Type	Select to enable unauthenticated users of the chart customization form to choose a font for displaying chart text.
Display Name	Select to enable unauthenticated users of the chart customization form to edit the display name of the chart.
Maximum Rows	Select to enable unauthenticated users of the chart customization form to specify the maximum number of bars to display in the chart.
Order By	Select to enable unauthenticated users of the chart customization form to set the display order for chart data.

Figure A–61 Chart Customization Form Button Options



Button Options

Choose the buttons that you want to display on the chart's customization form. End users can click the buttons to perform various actions on the chart.

Button	Name	Location	Alignment
☑ Run	Run	Top	Center
☑ Save	Save	Top	Center
☐ Batch	Batch	Top	Center
☑ Reset	Reset	Top	Center

Table A–31 Chart Customization Form Button Options

Option	Description
Button	(Full page only) Select one or more buttons to display on the chart customization form: ▪ **Run**—Displays the URL portlet with the options the user has specified in the chart customization form. ▪ **Save**—Saves users' current selections to enable them to rerun a chart using the same parameters without having to reenter them into the chart customization form. When you include this button, the chart customization form also displays a **Reset to Defaults** button that resets all values in the chart customization form to their default values. ▪ **Batch**—Runs the URL portlet in batch mode and saves the results in the database. ▪ **Reset**—Undoes any changes made to the chart customization form since the last Save, Reset, or Reset to Defaults event.
Name	Enter the label that will display on the selected button. Keep the name short to avoid displaying large buttons. You cannot change the label on the **Reset to Defaults** button.
Location	Specify the location of the button on the chart customization form. Choose from **Top**, **Top and Bottom**, or **Bottom**.
Alignment	Specify the alignment of the button on the chart customization form. Choose from **Left**, **Center**, or **Right**.

20. Click **Next**.

21. Optionally, select a template to define the look and feel of the page that will host the chart, and enter text for display at the top and/or bottom of the chart and the chart customization form.

    The template is applied only when the chart is displayed on a full Web page. If the chart is displayed as a portlet, the template you select here will be ignored and the style of the host page will be used.

    Table A–32 lists and describes the options available on this page (Figure A–62).

Figure A–62 Options Available on the Chart and Customization Form Text Page

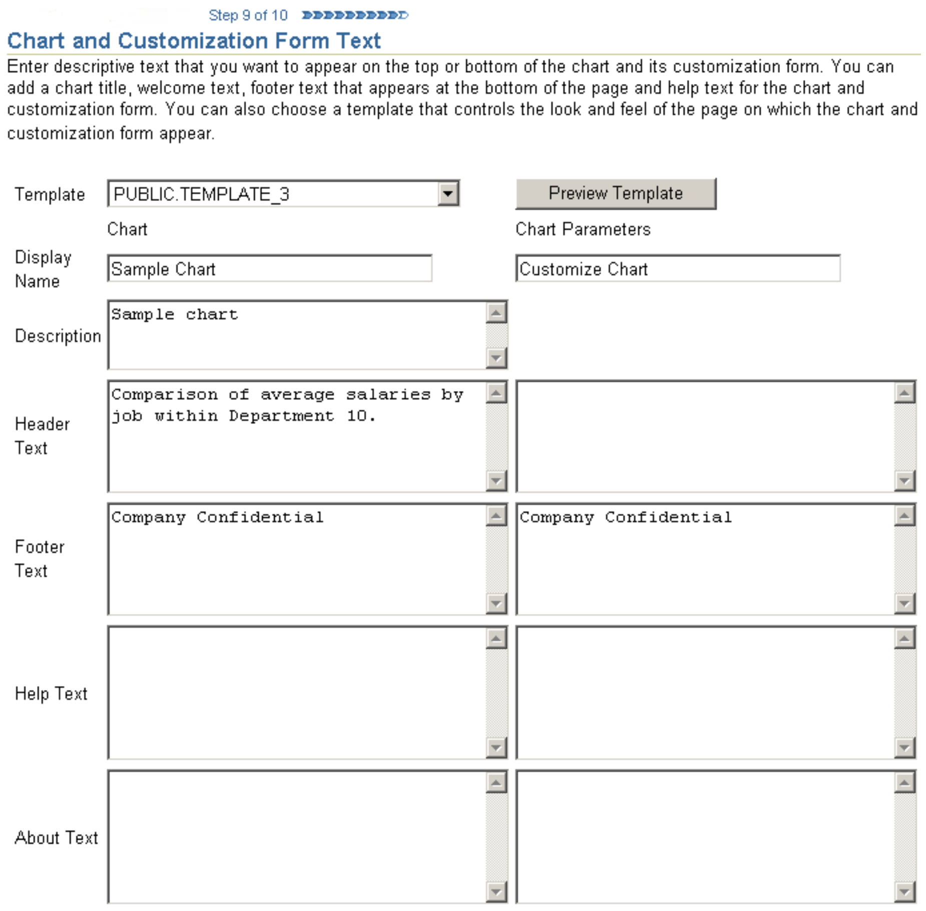

Step 9 of 10

Chart and Customization Form Text

Enter descriptive text that you want to appear on the top or bottom of the chart and its customization form. You can add a chart title, welcome text, footer text that appears at the bottom of the page and help text for the chart and customization form. You can also choose a template that controls the look and feel of the page on which the chart and customization form appear.

	Chart	Chart Parameters
Template	PUBLIC.TEMPLATE_3	Preview Template
Display Name	Sample Chart	Customize Chart
Description	Sample chart	
Header Text	Comparison of average salaries by job within Department 10.	
Footer Text	Company Confidential	Company Confidential
Help Text		
About Text		

Table A–32 Options Available on the Chart and Customization Form Text Page

Option	Description
Template	Choose a template to set the look and feel of the page that will host the chart. The template is applied only when the chart is displayed on a full Web page. If the chart is displayed as a portlet, the template you select here will be ignored and the style of the host page will be used. Display options, which you specified on previous wizard pages, control the look and feel of the chart itself.
Preview Template	Click to view the appearance of the template that is currently selected in the **Template** list.
Description	(Chart only.) Provide a description of the purpose of this chart. This is an attribute that may or may not display, depending on which attributes have been selected for display in the region where the chart or chart portlet is placed.
Display Name	Edit the chart's display name or the title for the chart customization form. You can specify HTML in this field.
Header Text	Enter any introductory text that you want to display at the top of the chart or the chart customization form. This text displays below the display name or title. You can specify HTML in this field.

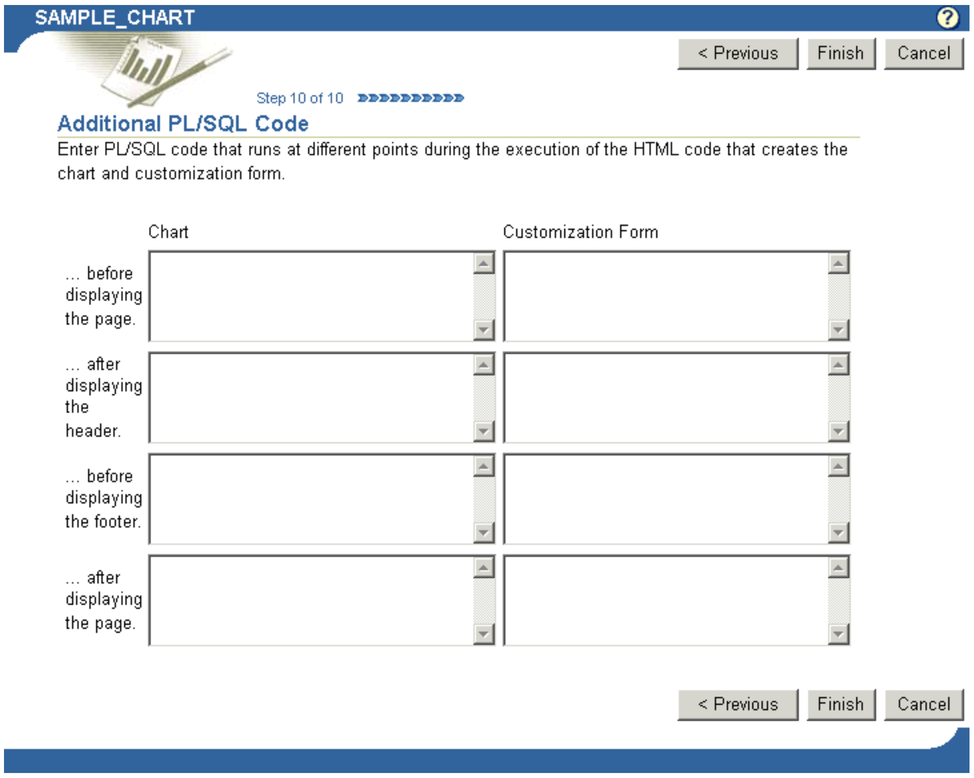
Table A–32 (Cont.) Options Available on the Chart and Customization Form Text Page

Option	Description
Footer Text	Enter any text that you want to display at the bottom of the chart or the chart customization form. You can specify HTML in this field.
Help Text	Enter any text that you want to display in a help page for the chart or chart customization form. OracleAS Portal automatically includes a help button with the chart and the chart customization form when they are displayed on a full page. When they are displayed as a portlet, whether the help button displays is controlled by the way the host portlet region has been configured. Users click the help button to display the text you enter here. You can specify HTML in this field.

22. Click **Next**.
23. Optionally, enter PL/SQL code to run at different points during the runtime creation of the chart or the chart customization form.

    Table A–33 lists and describes the options available on this page (Figure A–63).

Figure A–63 Additional PL/SQL Code Options for Charts



Additional PL/SQL Code

Enter PL/SQL code that runs at different points during the execution of the HTML code that creates the chart and customization form.

	Chart	Customization Form
... before displaying the page.		
... after displaying the header.		
... before displaying the footer.		
... after displaying the page.		

Table A–33 Options Available on the Additional PL/SQL Code Page

Option	Description
... before displaying the page.	Enter a PL/SQL procedure that will execute before the page containing the chart or the chart customization form displays.

Table A–33 (Cont.) Options Available on the Additional PL/SQL Code Page

Option	Description
... after displaying the header.	Enter a PL/SQL procedure that will execute after the chart or the chart customization form header displays. If you want to display output in the chart from a PL/SQL procedure, you must enter the PL/SQL in the **... after displaying the header** field.
... before displaying the footer.	Enter a PL/SQL procedure that will execute before the chart or customization form footer displays.
... after displaying the page.	Enter a PL/SQL procedure that will run after the page containing the chart or customization form displays.

24. Click **Finish**.

Clicking **Finish** saves your changes and takes you to a summary page where you can test your results (Figure A–64). See "Performing Test Runs on a Portlet" for more information on testing your results.

Figure A–64 Chart Summary Page

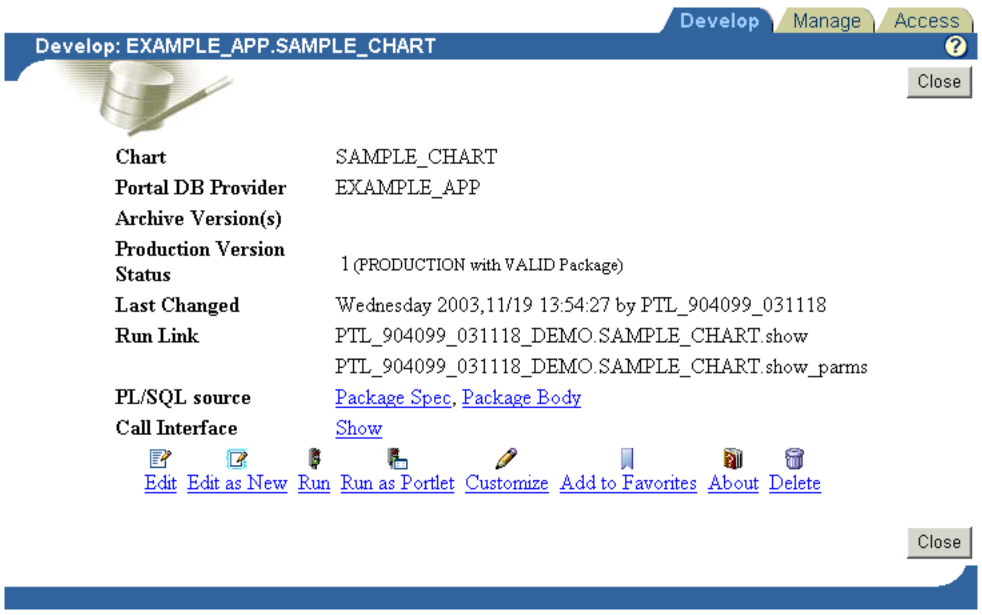
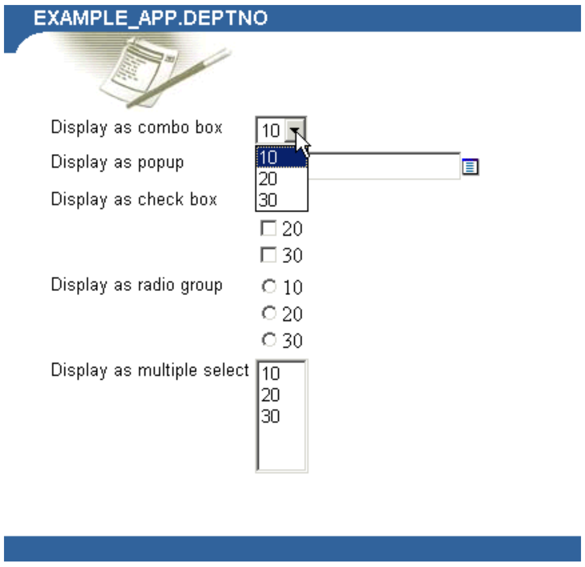


Chart	SAMPLE_CHART
Portal DB Provider	EXAMPLE_APP
Archive Version(s)	
Production Version Status	1 (PRODUCTION with VALID Package)
Last Changed	Wednesday 2003,11/19 13:54:27 by PTL_904099_031118
Run Link	PTL_904099_031118_DEMO.SAMPLE_CHART.show PTL_904099_031118_DEMO.SAMPLE_CHART.show_parms
PL/SQL source	Package Spec, Package Body
Call Interface	Show

A.1.4.6 Building Lists of Values Declaratively

Figure A–65 Different Display Types for a Dynamic List of Values



When you build a list of values using the OracleAS Portal List of Values Wizard, you have the option of creating two types of lists:

- Dynamic list of values
- Static list of values

A dynamic list of values takes its content from a SELECT statement you build through the wizard. The content of the list may change, depending on the data that is available for selection from the source database. A static list of values takes its content from values you enter in the wizard. These values do not change unless you edit the list and specifically alter them.

This section describes how to create a dynamic list of values.

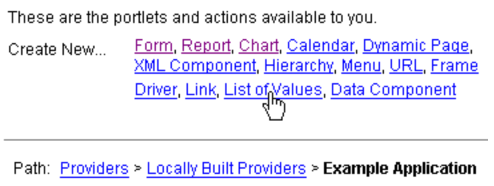
To create a dynamic list of values:

1. Follow the instructions detailed in "Building Portlets Declaratively".

   Return to this section once you complete step 6.

2. Click the **List of Values** link next to **Create New...** (Figure A–66).

Figure A–66 The List of Values Link Next to Create New ...

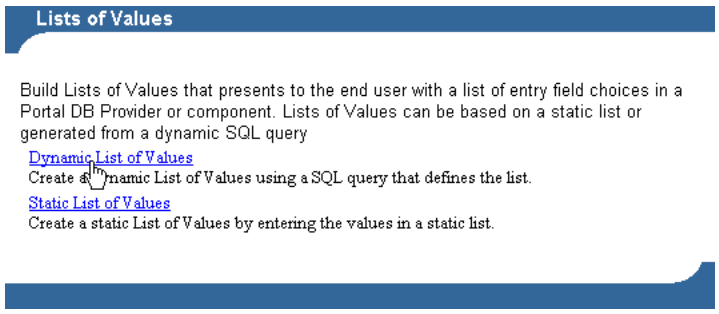

These are the portlets and actions available to you.

Create New... Form, Report, Chart, Calendar, Dynamic Page, XML Component, Hierarchy, Menu, URL, Frame Driver, Link, List of Values, Data Component

Path: Providers > Locally Built Providers > **Example Application**

3. On the List of Values page, click **Dynamic List of Values** (Figure A–67).

Figure A–67 The Dynamic List of Values Link



Lists of Values

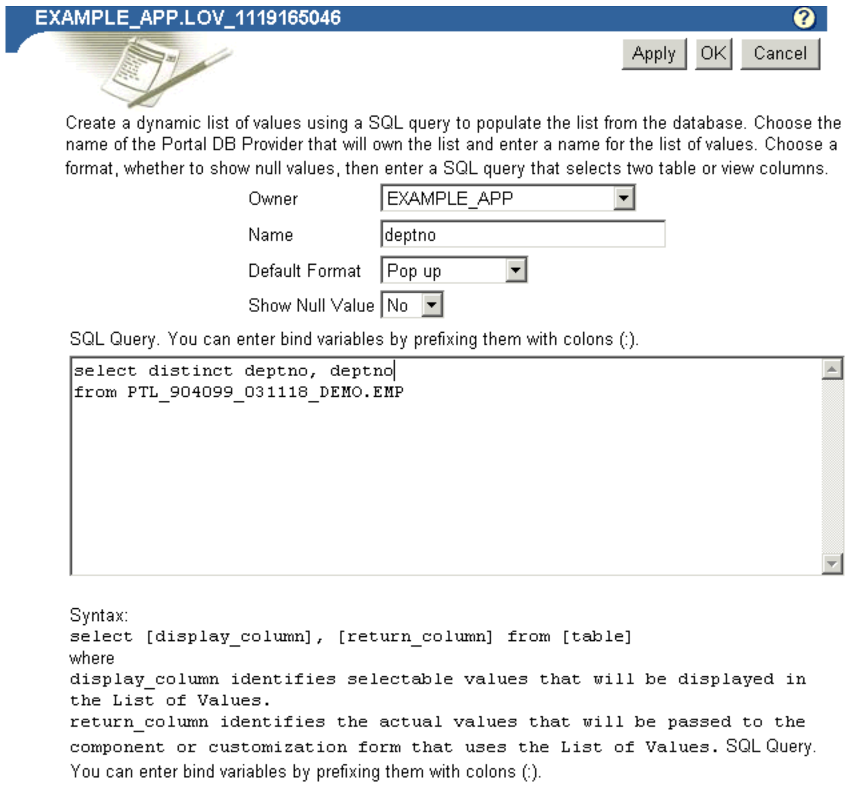
Build Lists of Values that presents to the end user with a list of entry field choices in a Portal DB Provider or component. Lists of Values can be based on a static list or generated from a dynamic SQL query

Dynamic List of Values
Create a dynamic List of Values using a SQL query that defines the list.

Static List of Values
Create a static List of Values by entering the values in a static list.

4. On the resulting page, select the Portal DB Provider that will own this list from the **Owner** drop-down list (Figure A–68).

Figure A–68 Defining List of Values Options



EXAMPLE_APP.LOV_1119165046

Create a dynamic list of values using a SQL query to populate the list from the database. Choose the name of the Portal DB Provider that will own the list and enter a name for the list of values. Choose a format, whether to show null values, then enter a SQL query that selects two table or view columns.

Owner	EXAMPLE_APP
Name	deptno
Default Format	Pop up
Show Null Value	No

SQL Query. You can enter bind variables by prefixing them with colons (:).

```
select distinct deptno, deptno
from PTL_904099_031118_DEMO.EMP
```

Syntax:
`select [display_column], [return_column] from [table]`
where
`display_column identifies selectable values that will be displayed in the List of Values.`
`return_column identifies the actual values that will be passed to the component or customization form that uses the List of Values.` SQL Query. You can enter bind variables by prefixing them with colons (:).

5. In the **Name** field, enter a name for this list of values (Figure A–68).

   This name appears on the selection list when you insert a list of values.

6. From the **Default Format** drop-down list, choose a default format for the list of values (Figure A–68).

   Choose from:

   - **Check box**—Users mark a check box to indicate their selection.
   - **Combo box**—Users select from a drop-down list or manually enter their selection.
   - **Pop up**—Users select from a secondary window, which automatically populates the relevant field with the selected value.

- **Radio group—**Users mark a circle (radio button) with their selection.
- **Multiple Select—**Users select one or more values from a list.

Note that portlet developers who add the list of values to their portlet can override this default and display the list in a different format.

7. Specify whether to show null values by selecting either **Yes**, **No**, or **%** from the **Show Null Value** drop-down list (Figure A–68).
   - When you select **Yes**, null values display as `Null` in the list of values.
   - When you select **No**, null values display as blank spaces in the list of values.
   - When you select **%**, blank spaces are also displayed. The difference between **%** and **No** is that, with %, when you place the list of values on a form, the null value (the wildcard %) is available for the set up of dynamic transactions in the form’s code.
8. Enter a SQL SELECT statement in the **SQL Query** text box (Figure A–68).

   Select two values from two table or view columns (though you can select each value from the same column):
   - The first column specifies the values displayed in the list of values.
   - The second column specifies the actual values that are passed to the portlet.

   For example, the query `select ename, empno from scott.emp` creates a list of values that displays employee names from the ENAME column. It passes the employee’s associated ID number (EMPNO) to the portlet when a user chooses a name from the list of values.

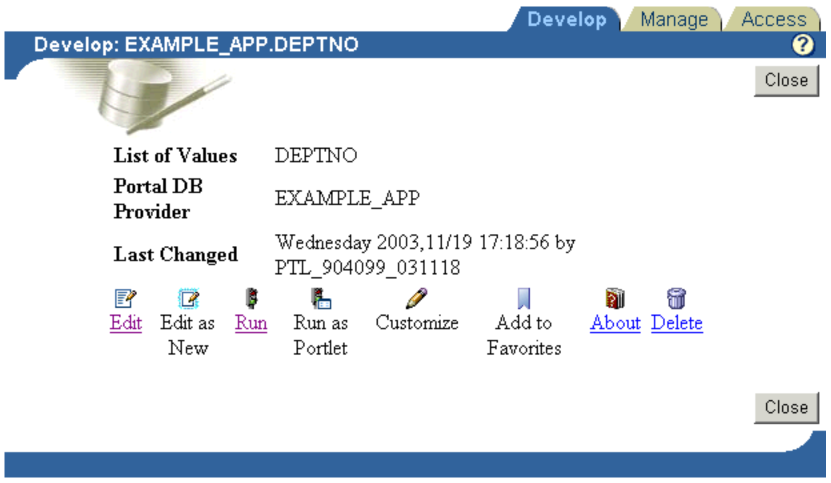
   Do not end your query with a semicolon.

   Note: You can enter bind variables by prefixing them with colons (:), but only if the list of values is to be used in a form portlet. Lists of values with bind variables will not work in other kinds of OracleAS Portal portlets.

   In the SQL query, you can use constants for the OracleAS Portal schema and the application schema. For more information, see "Referencing the OracleAS Portal Schema".
9. Click **OK** to save your changes.

   Clicking **OK** takes you to a summary page where you can manage the list of values (Figure A–69). For example, from this page, you can run the list to see the different ways it can display.

Figure A–69 List of Values Summary Page



List of Values	DEPTNO
Portal DB Provider	EXAMPLE_APP
Last Changed	Wednesday 2003,11/19 17:18:56 by PTL_904099_031118

Once you complete a list of values, it becomes available. It displays automatically on selection lists wherever you can add a list of values. For example, both the form and report customization forms you construct with Portlet Builder wizards allow for the addition of lists of values. As you go through these wizards, you'll note that the list you created here now appears as a selection option.

A.2 Editing a Portlet Builder Component

After you build a portlet using a wizard, you can use options in the Navigator to edit it. You can edit the portlet and overwrite the current version with your changes, or you can edit the portlet as new, and create a new version of it (consequently leaving the old, original version unchanged).

Where the wizards step you through the creation of a portlet, the editor provides tabs that contain the options that were available in the wizard. Each tab corresponds to a step in the wizard that created the portlet. Entry fields on each tab contain the values that were specified during creation of the portlet, or by the user who last edited the portlet.

A portlet is locked while you edit it, preventing other users from making changes to it. The portlet remains locked until you click **OK** in the Edit page. For more information, see "Managing Locks on Portlets".

Note: To edit a portlet, you must have at least the Edit privilege on the portlet or the provider that owns it.

To edit a portlet:

1. At the top of the Portal Builder page, click the **Navigator** link.
2. Click the **Providers** tab to bring it forward.
3. Click the **Locally Built Providers** link.

   The **Name** column displays the names of all the providers on which you have privileges.
4. Click the name of the OracleAS Portal database provider that contains the portlet that you want to edit.

5. Click the name of the portlet.
6. On the resulting page, the **Develop** tab should be exposed. Click the **Edit** link at the bottom of the tab.

   To leave the original version of a portlet unchanged, click the **Edit as New** link instead of **Edit**.
7. Click one or more tabs to bring them forward and edit their associated options.

   The tabs contain the same options that are available in the wizards. For more information about these options, see the previous sections that cover building portlets:

   - Building Forms Declaratively
   - Building Reports Declaratively
   - Building Charts Declaratively
   - Building Lists of Values Declaratively

   As you switch from tab to tab, OracleAS Portal keeps track of any changes you make. Click **OK** only after you have made all the changes on all the tabs that you want to make. If you decide you don't want to save your edits, click **Cancel**.

   Note that it is important that you click either **OK** or **Cancel**. Simply closing the window will cause the component to remain locked. Special measures must be taken to unlock inappropriately locked portlets. For more information, see "Managing Locks on Portlets".

A.3 Managing Portlets

For each locally-built portlet, OracleAS Portal provides a central location from which you can perform management tasks: the management page. This section describes how to locate any locally built portlet's management page and how to perform various management tasks. It includes the following subsections:

- Navigating to the Component Management Page
- Renaming a Portlet
- Deleting a Portlet
- Copying a Portlet
- Generating the PL/SQL Package for a Portlet

Note: To rename, delete, or copy a portlet, you must have at least the Edit privilege on the provider that owns the portlet.

To generate a portlet, you must have at least the View Source privilege on the portlet or on the provider that owns it.

A.3.1 Navigating to the Component Management Page

To navigate to the component management page:

1. Click the **Navigator** link at the top of the Portal Builder page.
2. Click the **Providers** tab to bring it forward.
3. On the Providers tab, click the **Locally Built Providers** link.

4. Click the name of the Database Provider that owns the portlet you will manage, then click the **Manage** link next to the component you will manage.

   Note the three tabs on this page:

   - **Develop**—Run, test, revise, delete and manage the various versions of a component.
   - **Manage**—Export, copy, rename (both internal and display names), generate, and monitor the performance of the component.
   - **Access**—Define security for the component within the OracleAS Portal framework. This includes publication of the component as a portlet, privilege inheritance from the provider, and cache invalidation.

A.3.2 Renaming a Portlet

To rename a portlet:

1. Navigate to the component management page for the portlet you will rename.

   For information on how to navigate to this page, see "Navigating to the Component Management Page".
2. Click the **Manage** tab to bring it forward.
3. Click the **Rename** icon.
4. Rename the internal and/or the display name for the portlet:
   a. For the internal name, in the **New Component Name** field enter a new name for the portlet.
   b. For the display name, in the **New Component Display Name** field enter a new display name for the portlet.
5. Click **OK**.

   The portlet is renamed with the name you provided.

A.3.3 Deleting a Portlet

To delete a portlet from the database:

1. Navigate to the component management page for the portlet you will delete.

   For information on how to navigate to this page, see "Navigating to the Component Management Page".
2. Click the **Develop** tab to bring it forward.
3. Click the **Delete** icon.

   The Delete Portlet page displays the versions of the portlet that you are authorized to delete.
4. Select the check box next to each version of the portlet that you want to delete.
5. Click **Yes**.

A.3.4 Copying a Portlet

To copy a portlet:

1. Navigate to the component management page for the portlet you will copy.

For information on how to navigate to this page, see "Navigating to the Component Management Page".

2. Click the **Manage** tab to bring it forward.
3. Click the **Copy** icon.

   The Copy Portlet page displays.
4. From the **New Owner** list, choose the provider that will own the new copy of the portlet.

   You can choose the same provider.
5. Provide new internal and display names for the portlet:
   a. For the internal name, in the **New Component Name** field enter a new name for the portlet.
   b. For the display name, in the **New Component Display Name** field enter a new display name for the portlet.
6. Click **OK**.

A.3.5 Generating the PL/SQL Package for a Portlet

Use this feature to make sure that the portlet code compiles without errors and performs within a reasonable time. This is particularly useful if you've written more complex code than is currently supported in the portlet builder wizards. For example, you could create a package; base a form or report on the package; then, using this feature, verify that the form or report will compile.

To generate the PL/SQL package for a portlet:

1. Navigate to the component management page for the portlet you will work with.

   For information on how to navigate to this page, see "Navigating to the Component Management Page".
2. Click the **Manage** tab to bring it forward.
3. Click the **Generate** icon.

   OracleAS Portal generates the PL/SQL package using the most recent OracleAS Portal routines.

A.3.6 Viewing Source Code

OracleAS Portal stores portlets in the database as PL/SQL packaged procedures. You can view the package spec and body for the portlet as well as its call interface.

The portlet call interface displays the arguments that can be set during run time. For example, if you created a report and selected **HTML** as a **Default Format** in the Display Options step of the Report build wizard, the call interface displays `HTML` as the default value for the `_format_out` argument.

When you run the package containing the portlet in PL/SQL or by calling it from a URL, you can edit the call interface to accept different arguments. You could change the `_format_out` argument to another report output format, such as ASCII.

This section describes how to view the package spec, body, and call interface for a portlet. It includes the following subsections:

- Viewing the Package Spec and Body for a Portlet

- Viewing the Call Interface for a Portlet

Note: To view portlet source code, you must have at least the View Source privilege on the portlet or the provider that owns it.

A.3.6.1 Viewing the Package Spec and Body for a Portlet

To view the package spec and body for a portlet:

1. Navigate to the component management page for the portlet you will work with.

   For information on how to navigate to this page, see "Navigating to the Component Management Page".

2. Click the **Develop** tab to bring it forward.

3. Click the **Package Spec** link next to **PL/SQL Source** to view the portlet's spec; click the **Package Body** link next to **PL/SQL Source** to view the body.

A.3.6.2 Viewing the Call Interface for a Portlet

To view the call interface for a portlet:

1. Navigate to the component management page for the portlet you will work with.

   For information on how to navigate to this page, see "Navigating to the Component Management Page".

2. Click the **Develop** tab to bring it forward.

3. Click the **Show** link next to **Call Interface**.

A.3.7 Managing Locks on Portlets

A portlet lock prevents other developers from overwriting your changes while you are editing a portlet. No other developer can edit the portlet when it is locked. Portlets are locked automatically when they are being edited. You can determine who is doing the editing by taking the steps outlined in this section.

Note: To view a portlet lock or to unlock a portlet, you must have at least the Edit privilege on the portlet or the provider that owns it.

To view the name of a user who has locked a portlet:

1. Navigate to the component management page for the portlet you will work with.

   For information on how to navigate to this page, see "Navigating to the Component Management Page".

2. Click the **Manage** tab to bring it forward.

3. Click the **Show Locks on this Component** link.

   Locked portlets show a **Status** of `EDIT` on the resulting page. In addition to status, this page also lists the user responsible for the lock, the length of time the version has been locked, and other information. It also provides an **Unlock** link you can click to unlock the portlet.

   The **Unlock** link is useful if a developer has edited a component, then closed the browser window without clicking **OK** or **Cancel**. In such case, the component remains locked. Use the **Unlock** link to unlock the component.

A.4 Managing Versions

OracleAS Portal enables you to store multiple versions of the same portlet in the database. Use the **Edit as New** link on the **Develop** tab (see Section A.3.1) to create a new version of the portlet that is based on the current production version. When you save your edits, you will be saving them to a new version, rather than overwriting any existing versions. The next time you edit the portlet, you can open the most current version or an earlier version of the portlet. This section describes how to access various versions of an OracleAS Portal component.

To access various versions of an OracleAS Portal component:

1. Navigate to the component management page for the portlet you will work with.

   For information on how to navigate to this page, see "Navigating to the Component Management Page".

2. Click the **Develop** tab to bring it forward.

3. To access the production version:

   - Click the **Edit** link to make changes to the existing production version.
   - Click the **Edit as New** link to make changes to a new version that is based on the production version.

     The production version is listed next to **Production Version Status** on the **Develop** tab. This field lists the production version, and indicates whether this version's code package is valid, that is, that it will run.

4. To access an archived version of the component:

   a. Next to **Archive Version(s)**, click the ***x*** **(Archive)** version you want to work with.

      The variable value (*x*) stands for the archive number.

   b. On the resulting page, click **Yes** to make the selected version the production version.

      Note that the version number listed next to **Production Version Status** changes to your selection.

   c. On the **Develop** tab, click the **Edit** or the **Edit as New** link to edit the newly selected production version of the component.

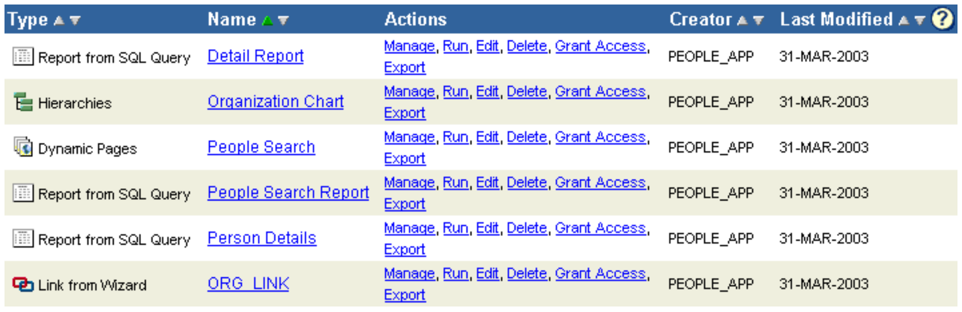
You can delete portlet versions from the database at any time. Follow the steps outlined in "Deleting a Portlet", then check the box next to the version(s) you want to delete, and click **Yes**. If you delete the production version and leave an archive version intact, you may want to select the archive version on the **Develop** tab and make it the production version.

The most recent version of a portlet is called the *production version*. The production version can be valid or invalid, depending on whether the database package containing the portlet will run without errors. If a portlet has a version status of **(PRODUCTION with INVALID package)**, you must either generate the portlet (see Section A.3.5) or edit the portlet (see Section A.2) to fix the errors before it will run.

A.5 Managing Portlet Security

The **Actions** column of the Navigator (Figure A–70) displays the actions you may perform on a portlet. The actions listed here depend on the access privileges you have been granted on the portlet.

Figure A–70 Portlet Actions



Type	Name	Actions	Creator	Last Modified
Report from SQL Query	Detail Report	Manage, Run, Edit, Delete, Grant Access, Export	PEOPLE_APP	31-MAR-2003
Hierarchies	Organization Chart	Manage, Run, Edit, Delete, Grant Access, Export	PEOPLE_APP	31-MAR-2003
Dynamic Pages	People Search	Manage, Run, Edit, Delete, Grant Access, Export	PEOPLE_APP	31-MAR-2003
Report from SQL Query	People Search Report	Manage, Run, Edit, Delete, Grant Access, Export	PEOPLE_APP	31-MAR-2003
Report from SQL Query	Person Details	Manage, Run, Edit, Delete, Grant Access, Export	PEOPLE_APP	31-MAR-2003
Link from Wizard	ORG_LINK	Manage, Run, Edit, Delete, Grant Access, Export	PEOPLE_APP	31-MAR-2003

By default, you have the *Manage* privilege on any portlet you create. The Manage privilege provides the highest level of access. This means you can run, edit, delete, or export the portlet. You can also grant access privileges to another user on any portlet that you create.

To build a portlet within OracleAS Portal, you must have at least the *Edit* privilege on the provider that will own the finished portlet. By default, portlets inherit whatever privileges are assigned to the provider that owns them. For example, after a component is created in the MY_APP provider, all developers with at least the *Execute* privilege on MY_APP can run the component after it is published as a portlet. Portlet owners can override these privileges and set access on a user-by-user level, rather than at the provider level.

Table A–34 lists the actions one can perform on a portlet and indicates which privileges enable a user to perform those actions.

Table A–34 Portlet Actions Associated with Portlet Privilege Levels

Action	Manage	Edit	View Source	Customize	Execute
Grant portlet privileges to other users	X				
Manage portlet locks	X	X			
Edit any portlet version	X	X			
Delete the portlet from the database	X	X			
Rename the portlet	X	X			
Export the portlet to another database	X	X			
Copy the portlet	X	X			
Generate the portlet PL/SQL package	X	X	X		
Monitor portlet usage	X	X	X		
View portlet's call interface, package spec, and body	X	X	X		
Customize the portlet	X	X	X	X	
Run the portlet	X	X	X	X	X
Add the portlet to the Favorites list	X	X	X	X	X

A.5.1 Granting Portlet Access Privileges

Note: To grant access privileges on a portlet to other users or groups, you must have at least the Manage privilege on the portlet or the provider that owns it.

You can define access to your portlet at the provider level and at the portlet level. This section describes how to inherit provider access privileges or override them by setting privileges at the portlet level. It contains the following subsections:

- Inheriting Portlet Access Privileges from a Provider
- Granting Access Privileges to Individual Users

A.5.1.1 Inheriting Portlet Access Privileges from a Provider

By default, access to your portlet is inherited from the provider that owns it. For example, all users who have the *Execute* privilege on the owning provider are automatically able to execute your portlet. Users with the *Edit provider* privilege can edit all the portlets owned by the provider, and so forth. You can set provider access through the **Grants** tab (for more information, see "Granting and Revoking Privileges on Database Objects"). On the Grants tab, you can grant different privileges to different groups of users. Once privileges on the provider are established, you can specify that any portlets created under the provider must inherit the same set of privileges. This section describes how to specify that portlets should inherit privileges granted to their host providers.

Note: Before you can publish Portlet Builder components as portlets, the host provider must be exposed to OracleAS Portal as a provider. For information on how to enable this setting, see "Exposing a Provider".

To inherit portlet access privileges from a provider:

1. Click the **Navigator** link at the top of the Portal Builder page.
2. Click the **Providers** tab to bring it forward.
3. Click the **Locally Built Providers** link.
4. Click the name of the Database Provider that contains the portlet on which you want to grant access privileges.
5. Click the **Manage** link next to the relevant portlet.

   The Manage page displays.
6. Click the **Access** tab.
7. Check the **Inherit Privileges from Provider** check box.

   Portlet access will be defined by the portlet's host provider.

A.5.1.2 Granting Access Privileges to Individual Users

You can override the default access privileges (that is, those granted through the provider) by setting them at the portlet level. This section describes how to set access privileges at the portlet level, allowing you to more specifically target privileges to individual users.

To grant access privileges to individual users:

1. Click the **Navigator** link at the top of the Portal Builder page.
2. Click the **Providers** tab to bring it forward.
3. Click the **Locally Built Providers** link.
4. Click the name of the Database Provider that contains the portlet on which you want to grant access privileges.
5. Click the **Manage** link next to the relevant portlet.

   The Manage page displays.
6. Click the **Access** tab.
7. Clear the **Inherit Privileges from Provider** check box, then click **Apply**.

   The Access page displays two new sections: **Grant Access** and **Change Access**.
8. In the **Grantee** field, enter the name of the user or group to whom you want to grant privileges.

   If you are not sure of the name of the user or group click the **Browse Users** or **Browse Groups** icon and select from the list provided.
9. From the privilege list, choose an access privilege to grant to the user or group.

   See Table A–34 for information on the actions allowed with different access privileges.
10. Click **Add**.

    The user or group you entered in the **Grantee** field displays along with the access privilege you granted in the **Change Access** section at the bottom of the page.

    Note: OracleAS Portal uses the Oracle Internet Directory for identity management, serving as the repository for users and groups. In Oracle Internet Directory, groups are uniquely identified by their distinguished name (DN). Each group has a unique DN, though many groups can share a common name, in the same way that two people can share a common name, yet have completely different lineage (that is, John Smith and John Doe). When working within the portal, groups created from within that portal are displayed simply with their common names. However, when the portal references a group from some other location in the Oracle Internet Directory—such as a group from some other portal associated with the same Identity Management Infrastructure—the DN of the group is displayed to distinguished it from the portal's locally defined groups.

11. (Optional) To modify a user or group's access privilege, choose a new privilege next to the user or group in the **Change Access** section.
12. (Optional) To remove all privileges, click the **Delete** icon next to the user or group in the **Change Access** section.
13. Click **Apply**.
14. Click **Close** to exit the Manage page and return to the Portal Navigator.

A.6 Performing Test Runs on a Portlet

After you create a component under a Portal DB Provider, you will likely want to test whether it runs to your satisfaction and make edits to it if it does not. All of this functionality is available to you through the **Develop** tab on the Manage page. From this tab, you can run a component as a full page, as a portlet, and through the component's customization form.

You also have the option of running the component as a portlet through the portlet customization form. But this option is not available on the Develop tab. To do this, you must first publish the component as a portlet, place the portlet on a page, then click the **Customize** link in the portlet header.

This section discusses how to run a component using these various methods. It includes the following subsections:

- Running a Component as a Full Page
- Running a Component as a Portlet
- Running a Component through the Customization Form
- Running the Component as a Portlet through the Portlet Customization Form

Note: To run a portlet, you must have at least the Execute privilege on the portlet or the provider that owns it. You must also ensure that the **Publish as Portlet** check box is selected on the **Access** tab of the Manage page.

You can run a portlet only if there is a valid portlet version. See "Managing Versions" for more information.

A.6.1 Running a Component as a Full Page

To run a component as a full page:

1. Click the **Navigator** link at the top of the Portal Builder page.
2. Click the **Providers** tab to bring it forward.
3. Click the **Locally Built Providers** link.
4. Click the name of the Database Provider that owns the portlet.
5. Click the **Manage** link next to the component you want to run.

   The **Develop** tab of the Manage page displays.
6. Click **Run**.

   The component displays in a separate browser window on a full Web page.

A.6.2 Running a Component as a Portlet

To run a component as a portlet:

1. Click the **Navigator** link at the top of the Portal Builder page.
2. Click the **Providers** tab to bring it forward.
3. Click the **Locally Built Providers** link.
4. Click the name of the Database Provider that owns the portlet.
5. Click the **Manage** link next to the component you want to run.

The **Develop** tab of the Manage page displays.

6. Click **Run as Portlet**.

   The portlet displays in a separate browser window in a smaller-than-full-page format.

A.6.3 Running a Component through the Customization Form

To run a component through the customization form:

1. Click the **Navigator** link at the top of the Portal Builder page.
2. Click the **Providers** tab to bring it forward.
3. Click the **Locally Built Providers** link.
4. Click the name of the Database Provider that owns the portlet.
5. Click the **Manage** link next to the component you want to run.

   The **Develop** tab of the Manage page displays.
6. Click **Customize**.

   The customization form displays.
7. Customize your settings and click the **Run** button.

   The component displays in a separate browser window on a full Web page.

A.6.4 Running the Component as a Portlet through the Portlet Customization Form

To run a component as a portlet through the portlet customization form:

1. Publish the component as a portlet:
   a. Click the **Navigator** link at the top of the Portal Builder page.
   b. Click the **Providers** tab to bring it forward.
   c. Click the **Locally Built Providers** link.
   d. Click the **Database Provider** that owns the component you will work with.
   e. Click the **Manage** link next to the portlet you will work with.
   f. On the Manage page, click the **Access** tab to bring it forward.
   g. Under the **Portal Access** heading, verify that **Publish as Portlet** is selected.

   Note: The **Publish As Portlet** check box is available only when the provider has been configured appropriately. For more information, see "Exposing a Provider".

   h. Click **Close** to save your change and return to the Portal Navigator.
2. Add the portlet to a page:
   a. Go to the page where you will place the portlet.
   b. Click the **Edit** link at the top of the page.
   c. Go to a portlet region on the page.
   d. Click the **Add Portlet** icon.

   e. Enter the portlet's display name in the Portlet Repository's **Search** field, and click **Go**.

      If you do not remember the exact name, you can drill to the portlet's location within the Portlet Repository. Typically, you'll find newly-created portlets under the **Portlet Staging Area** node. Click the node, then click the portlet's host provider name.

   f. Click the portlet to add it to the **Selected Portlets** list.

   g. Click **OK** to add the portlet and return the page where you have placed it.

3. Click the **Customize** link in the portlet header.

   If this link does not display, edit the host region's properties to include it (in Page Edit mode, click the **Edit Region** icon at the top of the region).

4. Customize your settings and click **OK**.

   The portlet displays with your customizations.

A.6.5 Running in Batch Mode

OracleAS Portal developers can add a Batch button to a customization form that enables end users to run the component in batch mode. Batch mode is asynchronous, allowing users to run a transaction in the background, freeing their systems for other tasks.You can place a Batch button on the customization forms for charts and reports. Batch processing is useful if the component is based on a large amount of data, or if you anticipate that the portlet will display many rows of data.

When you execute a job in batch mode, OracleAS Portal displays a page indicating that the job was submitted to the batch queue, and provides a number for identifying the job.

This section provides information on how to add the Batch button to an existing component, and the parameters that should be pre-set by the database administrator in the `init.ora` file to enable batch processing. It includes the following subsections:

- Setting init.ora Parameters for Batch Jobs
- Adding a Batch Button to an Existing Component

A.6.5.1 Setting init.ora Parameters for Batch Jobs

To enable Oracle Portal end users to execute jobs in batch mode, the database administrator should review the parameters in the `init.ora` file for the Oracle database where Oracle Portal is installed. If these parameters are not set correctly, even though batch jobs may be sent to the batch queue, they may not run.

Table A–35 provides some suggested settings:

Table A–35 Suggested Batch-Handling Settings for the init.ora File

`init.ora` **Parameter and Setting**	**Specifies**
job_queue_processes=2	Two background processes.
job_queue_intervals=60	The processes wake up every 60 seconds.
job_queue_keep_connections=TRUE	Sleep, don't disconnect.

A.6.5.2 Adding a Batch Button to an Existing Component

To add a Batch button to an existing component:

1. Navigate to the component management page for the portlet you will work with.

   For information on how to navigate to this page, see "Navigating to the Component Management Page".
2. If necessary, click the **Develop** tab to bring it forward.
3. Click the **Edit** link at the bottom of the tab.
4. Click the **Customization Form Display Options** tab to bring it forward.

   This is the third tab from the right.
5. Scroll to the **Button Options** section at the bottom of the tab.
6. Select the check box next to **Batch**.
7. Optionally, set the **Batch** button's display options:
   a. Enter a display name for the **Batch** button.

      The default is *Batch*.

   b. Specify the location for the button.

      Choose from **Top**, **Top and Bottom**, and **Bottom**. Top is the default

   c. Specify button alignment.

      Choose from **Center**, **Left**, or **Right**. Center is the default.
8. Click **OK** to save your changes and return to the Manage page
9. Click **Close** to return to the Portal Navigator.

A.7 Referencing the OracleAS Portal Schema

To make the import and export of applications more robust, wherever you can use PL/SQL code, you can use constants in place of the names of the OracleAS Portal Schema, the application schema, and the application name. Table A–36 lists the components that can use these constants and the places where they can be used within the Portlet Builder.

Table A–36 Constants for Applications and Schemas for Portlet Builder Components

Component	Constants	Places to Use in the Portlet Builder
Reports	#APP_SCHEMA# #PORTAL_SCHEMA# #APP_NAME#	▪ SQL query ▪ Table Name (#APP_SCHEMA#, #PORTAL_SCHEMA#) ▪ Advanced PL/SQL section ▪ LOV Name (#APP_NAME#)
Charts	#APP_SCHEMA# #PORTAL_SCHEMA# #APP_NAME#	▪ SQL query ▪ Table Name (#APP_SCHEMA#, #PORTAL_SCHEMA#) ▪ LOV Name (#APP_NAME#)
Calendars	#APP_SCHEMA# #PORTAL_SCHEMA# #APP_NAME#	▪ SQL query ▪ Advanced PL/SQL section ▪ LOV Name (#APP_NAME#)

Table A–36 (Cont.) Constants for Applications and Schemas for Portlet Builder

Component	Constants	Places to Use in the Portlet Builder
Forms	#APP_SCHEMA# #PORTAL_SCHEMA# #APP_NAME#	■ Table Name (#APP_SCHEMA#, #PORTAL_SCHEMA#) ■ Advanced PL/SQL section ■ LOV Name (#APP_NAME#)
Dynamic Page	#APP_SCHEMA# #PORTAL_SCHEMA# #APP_NAME#	■ SQL query ■ Advanced PL/SQL section ■ LOV Name (#APP_NAME#)
XML	#APP_SCHEMA# #PORTAL_SCHEMA# #APP_NAME#	■ SQL query ■ Advanced PL/SQL section ■ LOV Name (#APP_NAME#)
Hierarchy	#APP_SCHEMA# #PORTAL_SCHEMA# #APP_NAME#	■ Table Name (#APP_SCHEMA#, #PORTAL_SCHEMA#) ■ Advanced PL/SQL section ■ LOV Name (#APP_NAME#)
Frame Driver	#APP_SCHEMA# #PORTAL_SCHEMA# #APP_NAME#	■ SQL query ■ Advanced PL/SQL section ■ LOV Name (#APP_NAME#)
Links	#APP_SCHEMA# #PORTAL_SCHEMA#	Component Name
LOV	#APP_SCHEMA# #PORTAL_SCHEMA#	SQL query

These constants should be replaced at runtime, that is, passed as a parameter to the component when the component is run. When you edit a component in the Portlet Builder, any constants you have used should still appear "as is," that is, as #APP_SCHEMA# or whatever constant you used.

Table A–37 provides some examples of how you can use these constants.

Table A–37 Examples of the Use of Constants

Location in Portlet Builder	Examples

SQL Query	`Select * from #app_schema#.emp;`
	`Select * from #portal_schema#.emp;`
	`Select #app_name# from #portal_schema#.emp;`

Advanced PL/SQL Section	`Htp.p('#APP_NAME#');`
	`Htp.p('#APP_SCHEMA#');`
	`Htp.p('#PORTAL_SCHEMA#');`
	`Select * from #app_schema#.emp;`
	`Select * from #portal_schema#.emp;`
	`Select #app_name# from #portal_schema#.emp;`

LOV Name	`#APP_NAME#.dept_lov`
Table Name	`#APP_SCHEMA#.EMP` `#PORTAL_SCHEMA#.DEPT`

A.8 Coding Additional Functionality

The data for OracleAS Portal components is fetched from a database through the use of a SQL query. When you build charts and reports, you can write your own SQL query or use a build wizard to generate it for you. When you build a calendar or frame driver, you must write your own SQL query.

Charts, calendars, and hierarchies all require OracleAS Portal-specific syntax in the SQL statement that creates these portlets. For example, you must identify in the chart SQL statement a table or view column that supplies labels for the chart. You also identify a column that contains numeric data to determine the size of chart bars. The wizards for building charts and other types of components guide you through the specification of any special syntax.

You can also specify in the SQL query hypertext links that jump from column values that are displayed in the component to other components or Web pages. For example, you can link employee names on a department chart to individual reports containing information about each employee on the chart.

This section explores some of the ways you can build additional functionality into your locally built components. It includes the following subsections:

- Using Bind Variables
- Writing Event Handlers for Items on Forms
- Using PL/SQL to Get and Set Values in a Form
- Using PL/SQL to Get or Set Cookies in a Form or Report
- Defining Values through Page Parameters

A.8.1 Using Bind Variables

You can add bind variables to the SQL query that enables the locally built component to accept user input from a customization form. Each bind variable corresponds to a column in the table or view on which the component is based. They each create an entry field in the customization form for the portlet. The user can then choose which data to display in the component.

Note: You can also set up bind variables declaratively, through wizards. For an example of how to do this as well as for additional information on how to map the variable to a page parameter in OracleAS Portal, see "Defining Values through Page Parameters".

A bind variable appears in a SQL query as an alphanumeric string preceded by a colon (:var1, :var2, :var3, and so on). For example, the following SQL query creates entry fields for the `SALARY` and `DEPT` columns of the `SCOTT.EMP` table:

```
select ename, sal, dept
from scott.emp
where sal = :salary and deptno like :dept
```

Entering this SQL query creates a customization form with two text entry fields. The first enables the user to select a salary. The second field selects a department number. OracleAS Portal uses the values that the user typed in the customization form entry fields to create the output.

You do not need to know SQL to specify bind variables. You can identify columns that will accept parameters in the **Column** field in the **Customization Form Display Options** step of several portlet build wizards.

A.8.2 Writing Event Handlers for Items on Forms

You can code JavaScript and PL/SQL event handlers to customize the layout and operation of items on forms. An in-depth knowledge of JavaScript and PL/SQL is required to successfully include event handlers in your forms.

- You can use JavaScript event handlers to customize the behavior of form items, such as buttons, check boxes, lists, text boxes, and the like. Your browser must support the JavaScript version you use.
- You can use PL/SQL event handlers to customize the behavior of buttons.

If you code both JavaScript and PL/SQL event handlers for the same button, be careful to avoid potential conflicts between the execution of the two scripts.

Refer to your JavaScript or PL/SQL documentation for descriptions of the events that are available to you in OracleAS Portal.

This section describes how to add event handlers through the portlet build wizards. It includes the following subsections:

- Writing a JavaScript Event Handler for an Item on a Form
- Writing a PL/SQL Event Handler for a Button on a Form

A.8.2.1 Writing a JavaScript Event Handler for an Item on a Form

To write a JavaScript event handler for an item on a form:

1. Edit the form that includes the item for which you want to write an event handler:
   a. Go to the Manage page for the relevant form component.

      For information on navigating to this page, see "Navigating to the Component Management Page".
   b. If necessary, click the **Develop** tab to bring it forward.
   c. Click the **Edit** link toward the bottom of the tab.
2. Click the **Formatting and Validation Options** tab to bring it forward.
3. In the left frame, click the name of the item for which you will write a JavaScript event handler.
4. In the right frame, scroll to the **JavaScript Event Handlers** section.
5. From the **JavaScript Event Handlers** list, choose the required event, and enter the JavaScript to be executed when the event occurs.

Example A–1 Converting a Value to Uppercase

To convert a value in a text box or text area input field to uppercase whenever the field loses focus, choose the onBlur event and enter the following JavaScript:

```
this.value = this.value.toUpperCase();
```

Example A–2 Confirming Submission of a Form

To add a confirmation when the end user clicks a button before submitting a form, choose the onClick event and type the following JavaScript:

```
return confirm('Are you sure you want to submit the form?');
```

This displays a prompt dialog where the end user can click OK or Cancel; clicking OK submits the form.

Example A–3 Checking the Data Type of a Value

To verify that a value entered by the end user is a number only, choose the onChange event and type the following JavaScript:

```
if (isNaN(this.value)) {
alert('Please enter a valid number.');
}
```

A.8.2.2 Writing a PL/SQL Event Handler for a Button on a Form

To write a PL/SQL event handler for a button on a form:

1. Edit the form that includes the item for which you want to write an event handler:
   a. Go to the Manage page for the relevant form component.

      For information on navigating to this page, see "Navigating to the Component Management Page".
   b. If necessary, click the **Develop** tab to bring it forward.
   c. Click the **Edit** link toward the bottom of the tab.
2. Click the **Formatting and Validation Options** tab to bring it forward.
3. In the left frame, click the name of the button for which you will write a PL/SQL event handler.
4. In the right frame, scroll to the **PL/SQL Button Event Handler** section.
5. Define either:
   - A pre-defined event: Choose an event for the button from the **PL/SQL Button Event Handler** list.
   - A custom event: Choose **Custom** from the **PL/SQL Button Event Handler** list, and type the associated PL/SQL code.

A.8.3 Using PL/SQL to Get and Set Values in a Form

You can access and modify form field values using the methods of the form's session storage object contained in the variable `p_session`. To get a value, you must use type-specific get methods on `p_session`. This `p_session` has the following get functions for the data types NUMBER, VARCHAR2, and DATE:

- `get_value_as_NUMBER`
- `get_value_as_VARCHAR2`
- `get_value_as_DATE`

You must provide the block name and attribute name as arguments to these functions. For a single-block form, the block name is `DEFAULT`. For a master-detail form, the block name is `MASTER_BLOCK` or `DETAIL_BLOCK`

For the detail block, you must also provide the row index. The attribute name is the column name prefixed by A_.

To get the value of `DEPTNO` in a single block form, you would code:

```
declare
   my_deptno number;
begin
   my_deptno := p_session.get_value_as_NUMBER(
      p_block_name => 'DEFAULT',
      p_attribute_name => 'A_DEPTNO');
end;
```

For example, if you're doing this in the third record of a detail block, use:

```
my_deptno := p_session.get_value_as_NUMBER(
   p_block_name => 'DETAIL_BLOCK',
   p_attribute_name => 'A_DEPTNO',
   p_index => 3);
```

To set a field value, use `p_session.set_value`. This `p_session.set_value` function takes the same arguments as the `get_value` functions, with the addition of `p_value` for specifying the value (`p_value` is overloaded for any data type). Here, `set_value` is a procedure, and does not return a value. For example, to set the value of `DEPTNO`, use:

```
p_session.set_value(
   p_block_name => 'DEFAULT',
   p_attribute_name => 'A_DEPTNO',
   p_value => '20');
```

You can use the session storage object (`p_session`) in any button event handler and in the following places in the **Advanced PL/SQL** step of the Forms wizard (see also Table A–6):

- Before page
- After page
- Before processing
- After processing

Note that the Before form and After form handlers do not have access to the session storage object.

If you define custom code for a button mapped to an Insert, Update, or Delete event, you must call the procedure `doInsert`, `doUpdate`, or `doDelete` at an appropriate place in your code to perform the button's default function. For example, if you programmatically set a field value in an Insert button, call `doInsert` after `p_session.set_value` to perform the insert into the table.

A.8.4 Using PL/SQL to Get or Set Cookies in a Form or Report

When you use PL/SQL in OracleAS Portal, you have access to a variety of packages that are part of the PL/SQL Gateway. The gateway is delivered as part of OracleAS Portal. One of these is the package `owa_cookie`.

This package contains data types, procedures, and functions that enable you to send HTTP cookies to and get them from the client's browser. HTTP cookies are opaque strings sent to the browser to maintain state between HTTP calls. State can be maintained throughout the client's session—longer if an expiration date is included.

Maintaining a state means that the client can be identified throughout a session, allowing for the execution of transactions. So, for example, shopping applications can store information about currently-selected items; for-fee services can send back registration information, freeing the client from retyping a user ID on the next connection; and sites can store per-user preferences on the client, an in turn, have those clients supply those preferences every time it reconnects to that site.

The `owa_cookie` package contains subprograms and data types that you can use to set and get cookie values. Table A–38 lists some `owa_cookie` subprograms and data types, and describes how each is used.

Table A–38 owa_cookie Subprograms and Data Types

Subprogram or Data Type	Used to
`owa_cookie.cookie` data type	Contain cookie name-value pairs
`owa_cookie.get` function	Get the value of the specified cookie
`owa_cookie.get_all` procedure	Get all cookie name-value pairs
`owa_cookie.remove` procedure	Remove the specified cookie
`owa_cookie.send` procedure	Generate a "Set-Cookie" line in the HTTP header

Sessions are used by the Web Request Broker to maintain persistent states within gateways through multiple accesses over a period of time. Since the PL/SQL Gateway is unique in connecting to the database and all the states are maintained within the database, the concept of *sessions* does not apply to the PL/SQL Gateway. Instead, cookies can be used to maintain persistent state variables from the client browser.

A.8.5 Defining Values through Page Parameters

For many components, you can set values for columns using page parameters. For example:

- Indicate that a column is customizable in the component's Customization Form Display Options page.
- Enable parameters for the page group that will host the component.
- Add a page parameter to the page that will host the component.
- Map the page parameter to the portlet parameter.

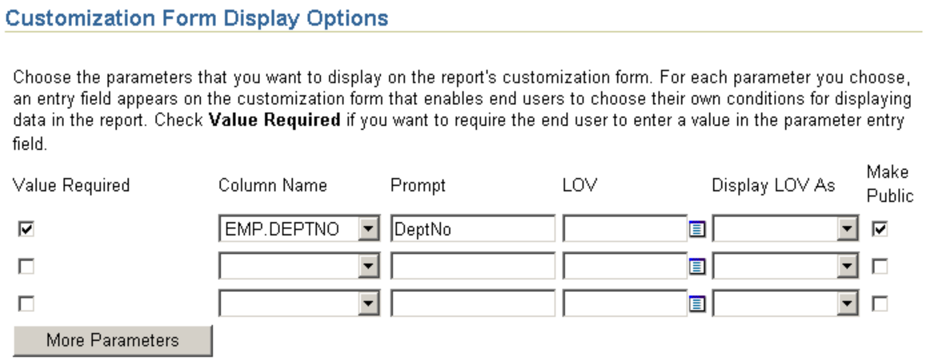
This section provides a brief overview of this process. We'll add a customizable value—a parameter or bind variable—to a report and map that parameter to the page that will host the portlet component.

Note: This procedure explains how to set a parameter for a report portlet that displays directly on a page. If you include the report portlet as a link (click **Edit Region**; on the **Attributes** tab, select the **Display Name Link** attribute; click the **Actions** icon, then **Edit Portlet Instance** link, and check **Link That Displays Item In New Browser Window**), the parameter attached has no effect when end users click the link to display the report: the result shown is `no rows displayed`.

To set up a parameter and use it on a portal page:

1. Create a report as described in "Building Reports Declaratively".
2. When you reach the step covering **Customization Form Display Options**, check the **Value Required** check box, select a column, assign a prompt to it, and make it public (Figure A–71).

Figure A–71 Report Customization Form Display Options


Customization Form Display Options

Choose the parameters that you want to display on the report's customization form. For each parameter you choose, an entry field appears on the customization form that enables end users to choose their own conditions for displaying data in the report. Check **Value Required** if you want to require the end user to enter a value in the parameter entry field.

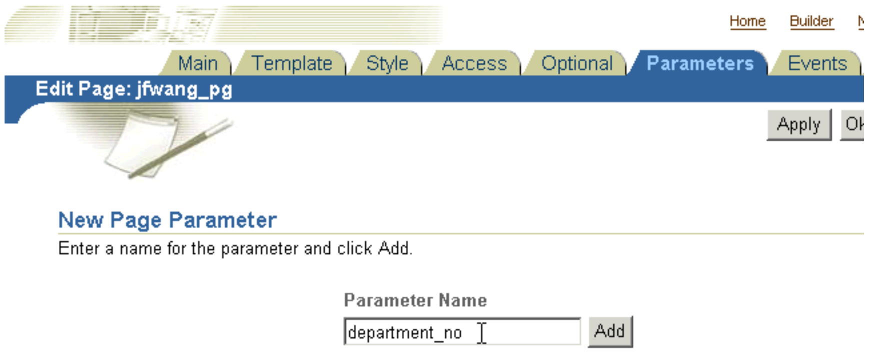
Value Required	Column Name	Prompt	LOV	Display LOV As	Make Public
✓	EMP.DEPTNO	DeptNo			✓

3. Complete the creation of your report.
4. Go to the Page Groups portlet (typically, on the **Build** tab of the Portal Builder.)
5. In the **Page Group** field, select the page group that will host the report and click the **Edit** button to edit the page group's properties.
6. Click the **Configure** tab to bring it forward, and scroll down to the **Parameters and Events** section of the tab.
7. Click the **Edit** link.
8. Select the **Enable Parameters and Events** check box.
9. Click **OK**, then click **Close**.
10. In the Page Groups portlet, click the **Browse** icon next to the **Name** field under **Edit a Page**.
11. In the resulting secondary window, drill to the page that will host the portlet, and click the **Return Object** link next to it.
12. Click the **Edit** button next to the now-populated **Name** field.
13. Add the new report portlet to the page:
    a. Click the **Add Portlet** icon over a region.
    b. In the Portlet Repository, click the **Portlet Staging Area** node.
    c. Click the name of the provider that owns the portlet.
    d. Click the portlet to add it to the **Selected Portlets** list.
    e. Click **OK** to return to the host page.

    The portlet will display with the following error message: `Error: Required field not set yet - emp.deptno (WWV-14900).` Once you define the page parameter and wire it to the portlet parameter you can provide a value, and this message will no longer display.
14. In the page toolbar at the top of the page, click the **Page: Properties** link.
15. On the resulting page, click the **Parameters** tab to bring it forward.

    This tab displays only when parameters and events are enabled for the page's parent page group.
16. In the **Parameter Name** field, enter a name for the page parameter you will wire to the parameter (bind variable) you created for your component (Figure A–72).

Figure A–72 The Parameter Name Field on the Parameters tab



Edit Page: jfwang_pg

New Page Parameter

Enter a name for the parameter and click Add.

Parameter Name: department_no [Add]

17. Click **Add**.

18. Optionally, go to the **Page Parameter Properties** section and configure the parameter (Figure A–73):

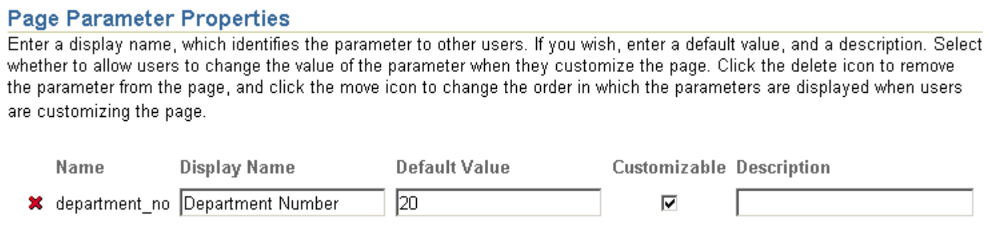
    a. Enter a display name, which identifies the parameter to other users.

    b. Enter a default value.

    c. Select whether to allow users to change the value of the parameter when they customize the page.

    d. Enter a description of the parameter.

Figure A–73 Defining Page Parameter Properties



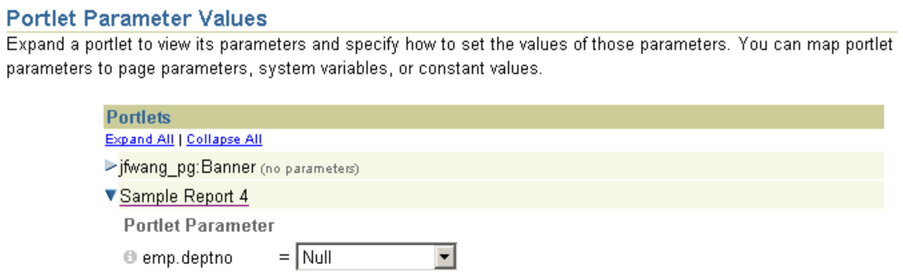
Page Parameter Properties

Enter a display name, which identifies the parameter to other users. If you wish, enter a default value, and a description. Select whether to allow users to change the value of the parameter when they customize the page. Click the delete icon to remove the parameter from the page, and click the move icon to change the order in which the parameters are displayed when users are customizing the page.

Name	Display Name	Default Value	Customizable	Description
✖ department_no	Department Number	20	✓	

19. Go to the **Portlet Parameter Values** section, and expand the node next to the portlet you added to the page in Step 13 (Figure A–74).

Figure A–74 Expanding a Portlet Node under Portlet Parameter Values



Portlet Parameter Values

Expand a portlet to view its parameters and specify how to set the values of those parameters. You can map portlet parameters to page parameters, system variables, or constant values.

Portlets

Expand All | Collapse All

▶ jfwang_pg:Banner (no parameters)

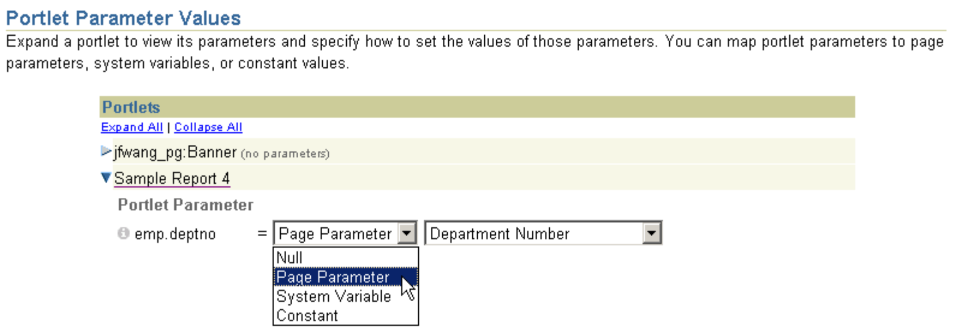
▼ Sample Report 4

Portlet Parameter

emp.deptno = Null

20. From the drop-down list next to your portlet parameter, select **Page Parameter** (Figure A–75).

Figure A–75 Selecting the Type of Mapping to Use with a Portlet Parameter



Portlet Parameter Values

Expand a portlet to view its parameters and specify how to set the values of those parameters. You can map portlet parameters to page parameters, system variables, or constant values.

Portlets

Expand All | Collapse All

jfwang_pg:Banner (no parameters)

Sample Report 4

Portlet Parameter

emp.deptno = Page Parameter Department Number

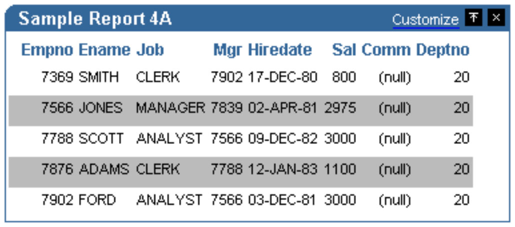
21. By default, the next drop-down list should display the relevant page parameter display name. If it doesn't, select the relevant name from the list (in our example, the relevant display name is **Department Number**).

    This maps the parameter you defined for your portlet (on the Customization Form Display Options page) to the parameter you have defined for the page (on the Parameters tab).

22. Click **OK**.

23. If you provided a default value for the page parameter in Step 18b, the report portlet will display with results relevant to the parameter, for example, with results for the specified department number (Figure A–76).

Figure A–76 Report Results for Department 20

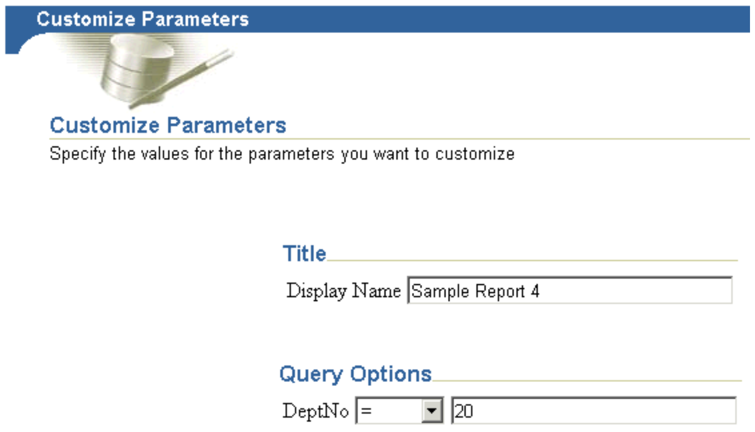


Sample Report 4A

Empno	Ename	Job	Mgr	Hiredate	Sal	Comm	Deptno
7369	SMITH	CLERK	7902	17-DEC-80	800	(null)	20
7566	JONES	MANAGER	7839	02-APR-81	2975	(null)	20
7788	SCOTT	ANALYST	7566	09-DEC-82	3000	(null)	20
7876	ADAMS	CLERK	7788	12-JAN-83	1100	(null)	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	(null)	20

If you did not provide a default value, you can provide one by clicking the **Customize** link in the portlet header (if the parameter was made customizable in Step 18b) and providing a value under Query Options (Figure A–77).

Figure A–77 Customizing Page and Portlet Parameter Values



Customize Parameters

Customize Parameters

Specify the values for the parameters you want to customize

Title

Display Name: Sample Report 4

Query Options

DeptNo = 20

A.9 Using Shared Components to Create a Look and Feel

Each portlet build wizard contains options that enable you to define a look and feel for the portlet's content. For example, you can choose the row colors that display in a report; the font type, color, and size for chart labels; and the height, width, and order of entry fields in a form. In addition to its built-in look-and-feel capabilities, OracleAS Portal offers advanced features through the Shared Components provider for the creation of a more customized look and feel.

Through the Shared Components provider, you can build JavaScripts for use in your wizard-built forms; define custom colors, fonts, and images; and apply User Interface Templates to pages and portlets.

The Shared Components provider is located in the Portal Navigator on the Providers tab under Locally Built Providers.

OracleAS Portal includes out-of-the-box a default set of shared components. These are *system* type components. The components you develop yourself under the Shared Components provider are *user* type components. You can edit, export, and delete a user-type shared component, but not a system-type. To edit a system type, you must first copy it, then edit the copy.

This section explores the capabilities of the Shared Components provider and describes how to put them to use. It includes the following subsections:

- Granting Access to Shared Components
- Using JavaScript to Create Field- and Form-Level Validation
- Creating Color Definitions
- Creating Image Definitions
- Creating Font Definitions
- Using User Interface Templates

A.9.1 Granting Access to Shared Components

Access privileges to the Shared Components provider define the actions you can perform on shared components. Table A–39 lists and describes the access privileges that are relevant to the Shared Components provider:

Table A–39 Shared Components Provider Access Privileges

Access Privilege	Enables You to
Manage	■ Grant shared component access privileges to other users or groups. ■ Create a new shared component. ■ Copy a System type shared component to create a new User type. ■ Edit any User type shared component. ■ Delete any User type shared component. ■ Export any User type shared component to another schema or database.
Create	■ Create a new shared component. ■ Copy a System type shared component to create a new User type.

By default, all users who are members of the DBA or PORTAL_DEVELOPERS groups have Manage shared component access privileges.

Access privileges are granted for all shared components. Access privileges cannot be granted on a shared component type, such as all JavaScripts, nor on an individual shared component, such as a particular JavaScript. Access must be granted to all shared components or none of them.

To grant access privileges to the Shared Components provider to a user or group:

1. In the Navigator, click the **Providers** tab to bring it forward.
2. At the root level of the Providers tab, click the **Locally Built Providers** link.
3. In the **Actions** column for the Shared Components provider, click the **Grant Access** link.
4. In the **Grantee** field, enter the name of the user or group that you want to allow to access the shared components.

   Optionally, click the **Browse Users** or **Browse Groups** icon and select from the list provided.
5. Choose the level of access to grant to the user or group from the list of available privileges.

   See Table A–39 for a list of relevant privileges.
6. Click **Add**.

   The user or group you specified now appears in the **Change Access** section at the bottom of the page.
7. (Optional) To modify an access privilege, choose a new privilege next to the user or group in the **Change Access** section.
8. (Optional) To remove a privileges, click the **Delete** icon next to the user or group in the **Change Access** section.
9. Click **OK**.

A.9.2 Using JavaScript to Create Field- and Form-Level Validation

OracleAS Portal provides tools for you to create JavaScripts that perform field- and form-level validation on entry fields in forms. Field-level validation is performed when the end user causes the onBlur condition to occur after entering a value in an entry field, for example, when tabbing to another entry field. Form-level validation occurs after the user enters a value in an entry field and submits all values on the page, for example, when clicking an OK button.

This section provides a few guidelines for using JavaScript to build field- or form-level validation in a form and describes how to create and add JavaScript to your form. It contains the following subsections:

- Guidelines for Writing Field- or Form-Level Validation JavaScript
- Creating JavaScript under the Shared Components Provider
- Adding JavaScript to a Form

A.9.2.1 Guidelines for Writing Field- or Form-Level Validation JavaScript

Follow these guidelines when writing a field- or form-level validation JavaScript:

Note: You must have at least the Create shared component access privilege to create a JavaScript.

You must have the Manage shared component access privilege to edit a JavaScript.

- All validation routines should be written as functions and return either `TRUE` or `FALSE` values.
- The routine should display an alert message to users if the element (entry field) being validated contains an invalid value.
- The routine should bring focus (position the cursor) to the entry field where the user entered the incorrect value flagged by the JavaScript.

Example A–4 demonstrates the use of JavaScript to perform field-level validation.

Example A–4 Example JavaScript

```
1->  function isNumber(theElement)
     {
2->     if (isNaN(Math.abs(theElement.value)))
        {
3->        alert("Value must be a number.");
4->        theElement.focus();
           return false;
        }
        return true;
     }
```

The JavaScript presented in Example A–4 performs field-level validation tasks in the following sequence:

1. Identifies the name of the function and the entry field being validated.
2. Checks whether the absolute value of the entry field is a number. A relevant JavaScript function that signifies that a value is not a number is: `isNaN`
3. If the value in the entry field is not a number, the user is alerted with the message, "Value must be a number."
4. The routine brings focus to the entry field.

A.9.2.2 Creating JavaScript under the Shared Components Provider

To create JavaScript under the Shared Components Provider:

1. In the Navigator, click the **Providers** tab to bring it forward.
2. At the root level of the Providers tab, click the **Locally Built Providers** link.
3. Click the link for the **Shared Components** provider.
4. Click the **Javascript** link next to **Create New...**
5. In the **JavaScript Name** field, enter a descriptive name for the JavaScript.

   For example, enter `NotNull` for a JavaScript that ensures there are no null values in an entry field.
6. In the **Language** field, enter the language in which the JavaScript will be written.

   For example, enter `JavaScript1.1` or `JavaScript1.2`.

7. Click **Next**.
8. Enter or copy your JavaScript into the field provided.
9. Click **Finish**.

To edit the JavaScript you just created, drill to the relevant JavaScript (in the Portal Navigator: Providers tab: Locally Built Providers link: Shared Components: JavaScripts: your JavaScript), and click the **Edit** link next to the relevant JavaScript.

A.9.2.3 Adding JavaScript to a Form

Once you have created JavaScript under the Shared Components provider, it is automatically added to selection list that is available in the Build Form wizard. For example, when you create a form, you can select a JavaScript that you created once you reach the Formatting and Validation Options page. The list of available JavaScripts is located on this page under the **Validation Options** section (for more information, see "Building Forms Declaratively").

A.9.3 Creating Color Definitions

Creating a color definition is an opportunity for you to provide a meaningful name to a color you plan to use in your portal. A color definition is an association between a color name and its hexadecimal value. For example, you might use a standard red as your corporate color. Using your ability to define colors, you could select that red and give it a meaningful name within OracleAS Portal, such as `standard_red`, `company_red`, or even `<`*`your company name`*`>_red`.

You associate a color value in the form #XXXXXX, where X is a value in the range 0-9 or A-F, with any name you choose. The color names you define are used in fonts, page backgrounds, and other elements of OracleAS Portal portlets.

Note: You must have at least the Create shared component access privilege to create a color definition.

You must have at least the Manage shared component access privilege to edit a color definition.

To create a color definition:

1. In the Navigator, click the **Providers** tab to bring it forward.
2. At the root level of the Providers tab, click the **Locally Built Providers** link.
3. Click the link for the **Shared Components** provider.
4. Click the **Color** link next to **Create New...**
5. In the **Color Name** field, enter the name you want to give the color.

   You can identify a color by any name you choose; for example, My_Blue_Color.
6. In the **Color Value** field, enter the hexadecimal value for the color, for example, #FF0000 for a shade of red.

   Hexadecimal values must be prefaced by the # character. You can click a color in the palette to automatically enter its hexadecimal value in the **Color Value** field.
7. (Optional) To preview a color value, click **Preview**.
8. When you are satisfied with your color definition, click **Create**.

The page updates with a link, which you can click to edit the color definition. If you do not want to edit the color definition at this time, click **Close**.

A.9.4 Creating Image Definitions

Creating an image definition is an opportunity for you to provide a meaningful name and type to an image you plan to use in your portal. An image definition is an association between an image name and the name of the file containing the image. You can specify any name you choose.

Note: You must have at least the Create shared component access privilege to create an image definition.

You must have at least the Manage shared component access privilege to edit an image definition.

To create an image definition:

1. In the Navigator, click the **Providers** tab to bring it forward.
2. At the root level of the Providers tab, click the **Locally Built Providers** link.
3. Click the link for the **Shared Components** provider.
4. Click the **Image** link next to **Create New...**
5. In the **Image Name** field, enter the name you want to give the image.

   You can identify an image by any name you choose; for example, SiteLogo.
6. In the **Image Filename** field, enter the name and extension of the file containing the image.

   For example, enter `logo.gif`. The image must be located in a directory mapped to the OracleAS Portal virtual directory `/images/`.

   Note: The `/images/` virtual directory path is set in the Oracle HTTP Listener `plsql.conf` file.

   ---
7. From the **Image Type** list, choose an image type, for example Icon 24x24.

   The type you choose will display next to the image in the Type column of the Portal Navigator.
8. Click **Create**.

   The page updates with a link, which you can click to edit the image definition. If you do not want to edit the image definition at this time, click **Close**.

A.9.5 Creating Font Definitions

Creating a font definition is an opportunity for you to provide a meaningful name to a font you plan to use in your portal. A font definition is an association between the name of a font face and any descriptive name you choose to give it. For example, your company may have identified a font to be used in all public documents. Using your ability to create a font definition, you could identify that font in OracleAS Portal with a custom name, such as `<`*`your company name`*`>_font`. The fonts you define are used for text that appears in OracleAS Portal portlets.

Note: You must have at least the Create shared component access privilege to create a font definition.

You must have at least the Manage shared component access privilege to edit a font definition.

To create a font definition:

1. In the Navigator, click the **Providers** tab to bring it forward.
2. At the root level of the Providers tab, click the **Locally Built Providers** link.
3. Click the link for the **Shared Components** provider.
4. Click the **Font** link next to **Create New...**
5. In the **Font Name** field, enter the name you want to give the font.

   You can identify a font by any name you choose; for example, web_banner_font.
6. In the **Font Value** field, enter the name of a font, for example Arial.

   You can specify the name of any font that is supported by a Web browser. If you specify a font that is not supported, the Web browser will use its own default font.

   You can specify alternative fonts by separating them with commas in the **Font Value** field; for example, Times New Roman, Times. In this example, if the user's Web browser does not support the Times New Roman font, it will use the Times font instead.
7. Click **Create**.

   The page updates with a link, which you can click to edit the font definition. If you do not want to edit the font definition at this time, click **Close**.

A.9.6 Using User Interface Templates

Use User Interface (UI) templates to provide a header and footer to a Web page or portlet. UI templates can be applied to either pages or OracleAS Portal portlets. By applying a template, you can automatically specify a page title, a title background, links to home and help pages, and background colors and images.

UI templates are good for standardizing the overall look and feel of many pages, or for standardizing groups of portlets in an OracleAS Portal database provider. For example, you can design a UI template for a provider that includes the company logo in the heading, the name of the company in the title, and a common background image. By ensuring every portlet in the provider uses the same UI template, you impose a standard appearance.

You can also use unstructured UI templates to pass values through JavaScript to a page that uses the template.

This section provides information about the two types of user interface template—structured and unstructured—and describes generally how to use them to create a look and feel. It also includes information about configuring a page group to allow for the use of UI templates and applying a UI template to a page. It contains the following subsections:

- Building a Structured User Interface Template
- Building an Unstructured User Interface Template
- Configuring a Page Group to Allow Use of UI Templates

- Applying a UI Template to a Page

A.9.6.1 Building a Structured User Interface Template

Structured user interface templates are applied only to portlets. They are created with a wizard. In the wizard, you specify images, text, and layout elements that are applied to every portlet that uses the template.

You cannot use your own HTML code to extend the template beyond the pre-identified attributes. To achieve greater flexibility, or to build a user interface template that you can apply to pages as well as portlets, you may want to build an unstructured user interface template (see "Building an Unstructured User Interface Template").

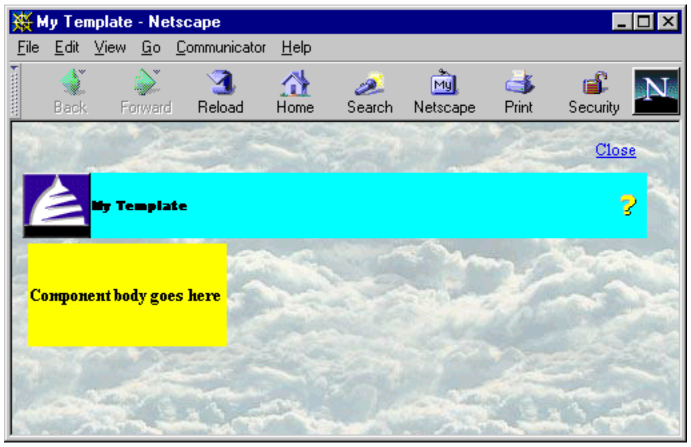
To build a structured user interface template:

1. In the Navigator, click the **Providers** tab to bring it forward.
2. At the root level of the Providers tab, click the **Locally Built Providers** link.
3. Click the link for the **Shared Components** provider.
4. Click the **User Interface Template** link next to **Create New...**
5. Click the **Structured UI Template** link.
6. In the **Template Name** field, enter a name for the template.

   Make the name as descriptive as possible. This is the name users will see when applying a UI template to a portlet during the build process. For example, if the template will be applied to all portlets created for a scheduling provider, you could name it Schedule_Template.
7. Select other options to refine the look and feel of the template.

   For example, you can choose an image that will appear in the upper left corner of the template and a background image that will display behind the portlet, as shown in Figure A–78.

Figure A–78 Structured UI Template with Cloud Image


My Template - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security
Close
My Template
Component body goes here

   If you have a question about an option, click the help icon. Leave an option blank if you do not want to include it in your template.
8. (Optional) Click **Preview** to open a new browser window that displays the UI template.

You can reselect options then click **Preview** again to see how the changes affect the look of the template.

9. When you are satisfied with your template, click **Create**.

   The page updates with a link, which you can click to edit the template. If you do not want to edit the template at this time, click **Close**.

A.9.6.2 Building an Unstructured User Interface Template

Unstructured user interface templates are based on HTML code that you supply. Because you are writing your own HTML code, you can create a more elaborate and sophisticated unstructured UI template than you can a structured UI template.

To create an unstructured user interface template, you first write HTML code to create a Web page. You can also copy this code into OracleAS Portal from another source, such as a Web page editor. Once entered into OracleAS Portal, you edit the HTML code to add substitution tags. When the HTML code executes, the substitution tags embed portlets, titles, and other elements into the Web page. For example, you can add a #BODY# tag that adds a portlet such as a chart or report to the original Web page background. For a list of all the substitution tags you can include in your HTML code, see Table A–40.

In an unstructured template, you can use <ORACLE></ORACLE> tags to include SQL statements or PL/SQL blocks. You can also include HEAD elements such as custom JavaScript, cascading style sheet references, and META tags. You can use special substitution tags for integrating page metadata to embed PL/SQL scripting.

You can also apply unstructured UI templates to pages. User interface templates control the decoration displayed **around** the page content. To control the look and feel of the **actual** page content, you must use page templates. For more information about using UI and page templates, refer to the *Oracle Application Server Portal User's Guide*, available on Portal Center (`http://www.oracle.com/technology/products/ias/portal/index.html`).

To build an unstructured user interface template:

1. In the Navigator, click the **Providers** tab to bring it forward.
2. At the root level of the Providers tab, click the **Locally Built Providers** link.
3. Click the link for the **Shared Components** provider.
4. Click the **User Interface Template** link next to **Create New...**
5. Click the **Unstructured UI Template** link.
6. In the **Template Name** field, enter a name for the template.

   Because this is the name users will see when applying a UI template to a page or portlet during the build process, you should make the name as descriptive as possible. For example, if the template will be applied to all portlets created for a calendar, you could name it Calendar_Template.

7. In the **Template Definition** field, enter or paste the HTML code you want to use as the basis for your unstructured user interface template.

   The HTML code you supply should create a Web page.

8. Embed substitution tags in the HTML code in the location where you want the items associated with the tags to appear in the finished template.

Table A–40 lists and describes the tags you can use. For example, you may want to embed a #BODY# substitution tag in the code. #BODY# adds the main body of the page, such as page content or an OracleAS Portal portlet to the Web page when the HTML code executes. If the HTML source code divides a page into two frames, you can embed the #BODY# tag in different places in the code, causing the portlet to display in the left frame or the right frame.

Table A–40 Unstructured UI Template Substitution Tags

Tag	Value set by
#BODY#	The portlet itself or the page's portlets.
#IMAGE_PREFIX#	The OracleAS Portal images directory as specified in the plsql.conf configuration file.
#USER#	The user name of the user who is currently logged on.
#USER.FULLNAME#	The full name of the user who is currently logged on. The full name includes the user's first, middle, and last names.
#VERSION#	The version of this installation of OracleAS Portal.
#HELPSCRIPT#	The JavaScript function used to open a window to display the online help. If you include #HELPLINK# in your template, you should also include this tag.
#DIRECTION#	The direction of character layout (left-to-right or right-to-left). This value is set based on the language.
#ALIGN_LEFT#	Align text to the left.
#ALIGN_RIGHT#	Align text to the right
#PAGE.STYLE#	The HTML LINK element that references the page's cascading style sheet. This tag is allowed only in the document HEAD.
#PAGE.STYLE.URL#	The URL of the page's cascading style sheet.
#PAGE.BASE#	The HTML base element for the base URL of the document. This tag is allowed only in the document HEAD.
#PAGE.BASE.URL#	The base URL.
#PAGE.BGIMAGE#	The HTML source for the whole page background image.
#PAGE.BGCOLOR#	The HTML source for the whole page background color.
#PAGE.SUBPAGELINK#	The URL for a sub-page link.
#PORTAL.HOME#	The HTML image hyperlink to the portal home page.
#PORTAL.HOME.URL#	The URL of the portal home page.
#PORTAL.HOME.IMAGE#	The image used for the portal home page link.
#PORTAL.HOME.LABEL#	The text used for the portal home page link.
#PORTAL.NAVIGATOR#	The HTML image hyperlink to the Navigator.
#PORTAL.NAVIGATOR.URL#	The URL of the Navigator.
#PORTAL.NAVIGATOR.IMAGE#	The image used for the Navigator link.
#PORTAL.NAVIGATOR.LABEL#	The text used for the Navigator link.

Table A–40 (Cont.) Unstructured UI Template Substitution Tags

Tag	Value set by
#PORTAL.HELP#	The HTML image hyperlink to the online help.
#PORTAL.HELP.URL#	The URL of the online help.
#PORTAL.HELP.IMAGE#	The image used for the online help link.
#PORTAL.HELP.LABEL#	The text used for the online help link.
#PORTAL.LOGOUT#	The HTML text hyperlink to the logout URL.
#PORTAL.LOGOUT.URL#	The logout URL.
#PORTAL.LOGOUT.LABEL#	The text used for the logout link.
#PORTAL.ACCOUNTINFO#	The HTML text hyperlink to the account information dialog.
#PORTAL.ACCOUNTINFO.URL#	The URL of the account information page.
#PORTAL.ACCOUNTINFO.LABEL#	The text used for the account information link.
#PORTAL.COMMUNITY#	The name of the OracleAS Portal Community Web site.
#PORTAL.COMMUNITY.URL#	The URL of the OracleAS Portal Community Web site.
#PORTAL.COMMUNITY.IMAGE#	The image of the OracleAS Portal Community Web site.
#PORTAL.COMMUNITY.LABEL#	The label of the OracleAS Portal Community Web site.
#PAGE.CUSTOMIZEPAGE#	The HTML text hyperlink to the customize page.
#PAGE.CUSTOMIZEPAGE.URL#	The URL of the customize page.
#PAGE.CUSTOMIZEPAGE.LABEL#	The text used for the customize page link.
#PAGE.EDITPAGE#	The HTML text to allow page editing.
#PAGE.EDITPAGE.URL#	The HTML text to allow page URL editing.
#PAGE.EDITPAGE.LABEL#	The HTML text to allow page label editing.
#PAGE.REFRESH#	The HTML text hyperlink used to refresh the page.
#PAGE.REFRESH.URL#	The refresh URL.
#PAGE.REFRESH.LABEL#	The text used for the refresh link.

9. (Optional) Click **Preview** to open a new browser window that displays the UI template.

   You can update the code then click **Preview** again to see how the changes affect the look of the template.

10. When you are satisfied with your template, click **Create**.

    The page updates with a link, which you can click to edit the template. If you do not want to edit the template at this time, click **Close**.

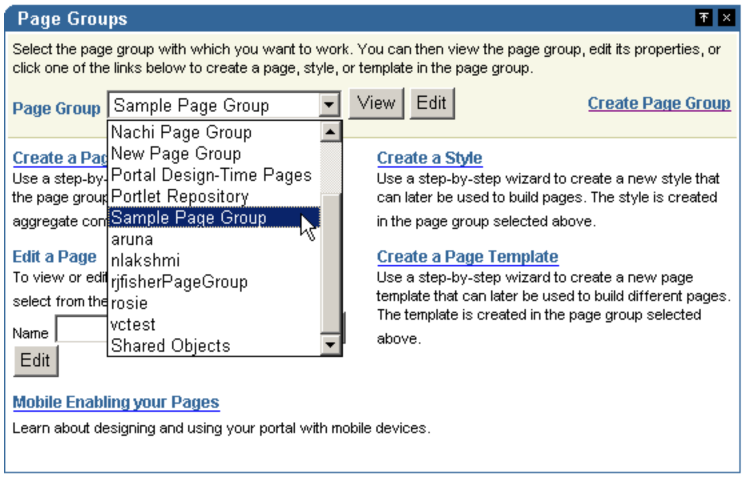
A.9.6.3 Configuring a Page Group to Allow Use of UI Templates

Before you can apply a UI template to a page, you must configure the host page group to allow for their use. To configure a page group to allow for UI templates:

1. Go to the Page Groups portlet in the Portal Builder, and select the relevant page group from the **Page Group** drop-down list (Figure A–79).

In a typical installation, you will find the Page Groups portlet on the **Build** tab of the Portal Builder.

Figure A–79 Selecting a Page Group in the Page Groups Portlet



Page Groups

Select the page group with which you want to work. You can then view the page group, edit its properties, or click one of the links below to create a page, style, or template in the page group.

Page Group: Sample Page Group [View] [Edit] Create Page Group

Create a Pag
Use a step-by-
the page group
aggregate con

Edit a Page
To view or edi
select from the
Name
[Edit]

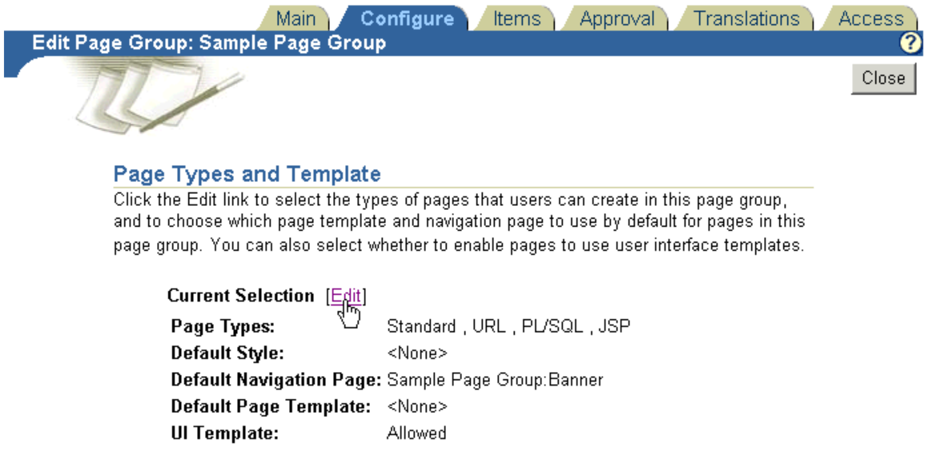
Create a Style
Use a step-by-step wizard to create a new style that can later be used to build pages. The style is created in the page group selected above.

Create a Page Template
Use a step-by-step wizard to create a new page template that can later be used to build different pages. The template is created in the page group selected above.

Mobile Enabling your Pages
Learn about designing and using your portal with mobile devices.

2. Click the **Edit** button next to the selected page group.
3. On the Page Group Properties page, click the **Configure** tab to bring it forward.
4. On the **Configure** tab, click the **Edit** link under **Page Types and Template** (Figure A–80).

Figure A–80 The Edit Link under Page Types and Template



Edit Page Group: Sample Page Group

[Close]

Page Types and Template

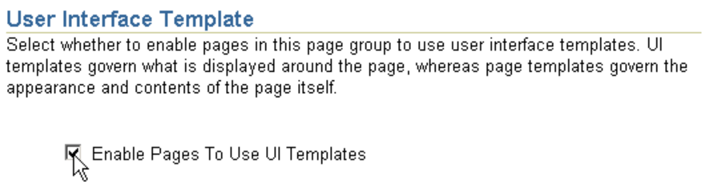
Click the Edit link to select the types of pages that users can create in this page group, and to choose which page template and navigation page to use by default for pages in this page group. You can also select whether to enable pages to use user interface templates.

Current Selection [Edit]

Page Types:	Standard , URL , PL/SQL , JSP
Default Style:	<None>
Default Navigation Page:	Sample Page Group:Banner
Default Page Template:	<None>
UI Template:	Allowed

5. On the resulting Page Defaults page, scroll down to the **User Interface Template** section, and enable the **Enable Pages To Use UI Templates** check box (Figure A–81).

Figure A–81 The User Interface Template Section of the Page Defaults Page



User Interface Template

Select whether to enable pages in this page group to use user interface templates. UI templates govern what is displayed around the page, whereas page templates govern the appearance and contents of the page itself.

☑ Enable Pages To Use UI Templates

6. Click **OK** to save your change and exit the Page Defaults page.
7. Click **Close** to exit Page Group Properties.

See Section A.9.6 for information on creating a UI Template. See Section A.9.6.4 for information on applying a UI template to a page.

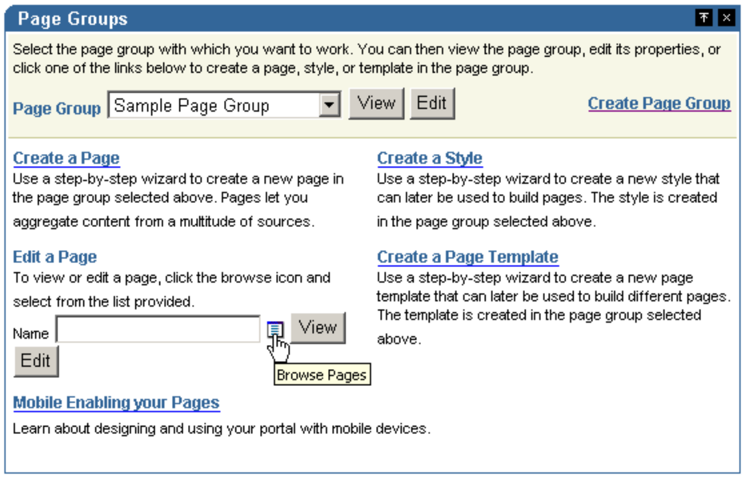
A.9.6.4 Applying a UI Template to a Page

Once your page group is configured to allow the use of UI templates (Section A.9.6.3), you can apply the ones you have created to pages in that page group. To apply a UI template:

1. Go to the Page Groups portlet, and click the **Browse Pages** icon next to the **Name** field under **Edit a Page** (Figure A–82).

   In a typical installation, you will find the Page Groups portlet on the **Build** tab of the Portal Builder.

Figure A–82 The Browse Pages Icon under Edit a Page


Page Groups

Select the page group with which you want to work. You can then view the page group, edit its properties, or click one of the links below to create a page, style, or template in the page group.

Page Group: Sample Page Group — View — Edit — Create Page Group

Create a Page
Use a step-by-step wizard to create a new page in the page group selected above. Pages let you aggregate content from a multitude of sources.

Create a Style
Use a step-by-step wizard to create a new style that can later be used to build pages. The style is created in the page group selected above.

Edit a Page
To view or edit a page, click the browse icon and select from the list provided.

Name — View — Edit

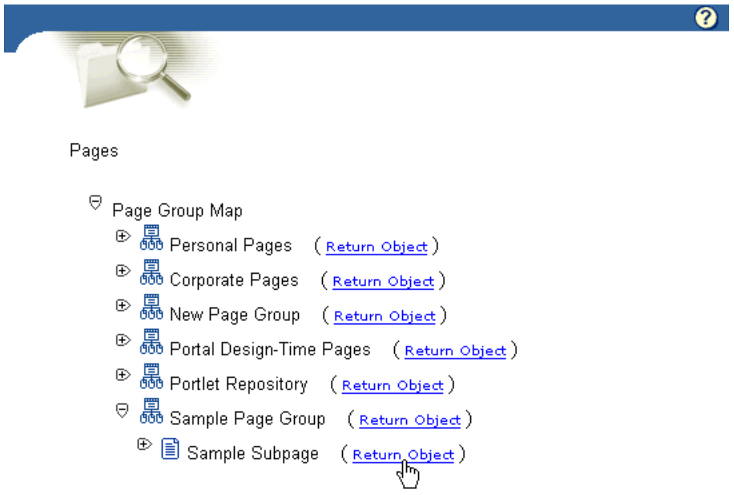
Browse Pages

Create a Page Template
Use a step-by-step wizard to create a new page template that can later be used to build different pages. The template is created in the page group selected above.

Mobile Enabling your Pages
Learn about designing and using your portal with mobile devices.

2. Drill down to the page to which you will apply a UI template.
3. Click the **Return Object** link next to the relevant page (Figure A–83).

Figure A–83 The Return Object Link next to a Page

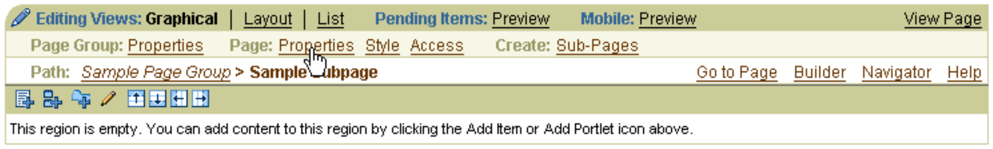

Pages

Page Group Map
- Personal Pages (Return Object)
- Corporate Pages (Return Object)
- New Page Group (Return Object)
- Portal Design-Time Pages (Return Object)
- Portlet Repository (Return Object)
- Sample Page Group (Return Object)
  - Sample Subpage (Return Object)

4. In the Page Groups portlet, click the **Edit** button next to the **Name** field.
5. On the resulting page, click the **Page: Properties** link in the toolbar at the top of the page (Figure A–84).

   Be sure to click the **Page: Properties** link, rather than **Page Group: Properties** link.

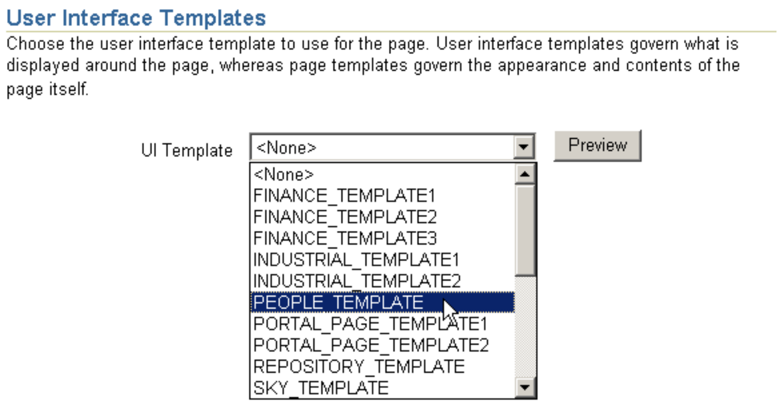
Figure A–84 The Page: Properties Link in the Page Toolbar


Editing Views: Graphical | Layout | List
Pending Items: Preview
Mobile: Preview
View Page
Page Group: Properties
Page: Properties Style Access
Create: Sub-Pages
Path: Sample Page Group > Sample Subpage
Go to Page Builder Navigator Help

This region is empty. You can add content to this region by clicking the Add Item or Add Portlet icon above.

6. On the Page Properties page, click the **Optional** tab to bring it forward.
7. Go to the **User Interface Templates** section on the Optional tab, and select a UI template from the **UI Template** drop-down list (Figure A–85).

Figure A–85 Selecting a UI Template on the Optional Tab



User Interface Templates

Choose the user interface template to use for the page. User interface templates govern what is displayed around the page, whereas page templates govern the appearance and contents of the page itself.

8. Click **OK** to save your change and return to the page.

A.10 Example: Building Charts and Reports

This example assumes that you have access to an OracleAS Portal database provider in the SCOTT schema called myCompany_DB_Provider. If you have the appropriate privileges, you can create this provider yourself (for information about how to do this, see "Creating a Provider for Locally Built Portlets"). If you do not have the appropriate privileges to create OracleAS Portal database providers, ask your portal administrator to create the provider for you.

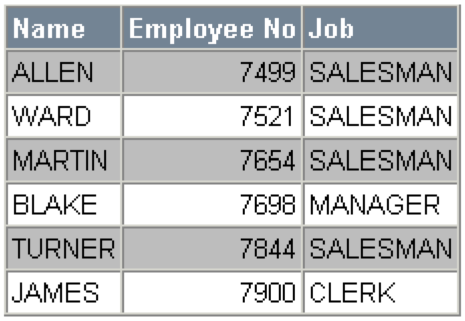
This example includes the following exercises:

- Exercise: Building the Team Details Report
- Exercise: Building the Average Salaries Chart
- Exercise: Building the Team Bonuses Report

A.10.1 Exercise: Building the Team Details Report

The Team Details report, shown in Figure A–86, displays a list of employees in the Sales department (department 30).

Figure A–86 Team Details Report



Name	Employee No	Job
ALLEN	7499	SALESMAN
WARD	7521	SALESMAN
MARTIN	7654	SALESMAN
BLAKE	7698	MANAGER
TURNER	7844	SALESMAN
JAMES	7900	CLERK

To build the Team Details report:

1. In the Navigator, click the **Providers** tab to bring it forward.
2. At the root level of the Providers tab, click the **Locally Built Providers** link.
3. Click the link for the provider **myCompany_DB_Provider**.

   If you do not see this provider, you can create it. For more information, see "Creating a Provider for Locally Built Portlets".
4. Click the **Report** link next to **Create New...**
5. Click the **Reports From Query Wizard** link.
6. In the **Name** field, enter `<YourName>_team_details`.
7. In the **Display Name** field, enter `Team Details`.
8. In the **Description** field, enter `team details exercise`.
9. In the **Portal DB Provider** list, **MYCOMPANY_DB_PROVIDER** should already be selected.
10. Click **Next**.
11. In the **Tables and Views** field, enter `SCOTT.EMP` if necessary.
12. Click **Add**.
13. Click **Next**.
14. From the **Columns** list, select:
    - EMP.ENAME
    - EMP.EMPNO
    - EMP.JOB

    Click the right arrow button after each selection to move it from **Columns** to **Selected Columns**.
15. If necessary, use the up and down arrows to the right of **Selected Columns** to arrange the columns in the order specified above.
16. Click **Next**.
17. From the **Column Name** list, choose **EMP.DEPTNO**.
18. From the **Condition** list, choose **=**.
19. In the **Value** field, enter `30`.
20. Click **Next**.

21. Select **Tabular**.
22. Click **Next**.
23. Next to each column, enter the **Column Heading Text** as indicated in Table A–41:

Table A–41 Team Details Report Column Heading Text

Column	Column Heading Text
ENAME	Name
EMPNO	Employee No
JOB	Job

This will add descriptive labels above columns that appear in your report.

24. Click **Next** twice, or until you see the **Display Options** step of the wizard.
25. In both the **Full Page Options** and **Portlet Options** sections, select the values shown in Table A–42:

Table A–42 Team Details Report Full Page and Portlet Display Options

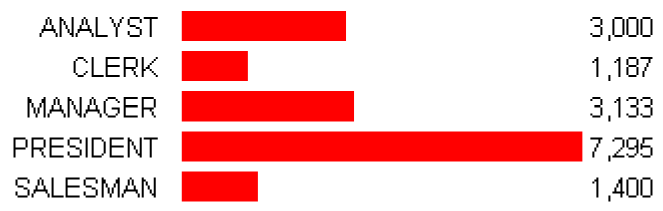
Option	Value
Heading Font Face	Arial
Heading Font Color	White
Heading Size	10pt
Row Text Font Face	Arial
Row Text Font Color	Black
Row Text Size	10pt
Heading Background Color	Slate Gray
Border	Thin Border

26. Click **Finish**.
27. Click **Run as Portlet** to see what your report will look like.
28. Close the browser window where the report is displayed, and click **Close** to close the component manager and return to the Portal Navigator.

A.10.2 Exercise: Building the Average Salaries Chart

The Average Salaries chart, shown in Figure A–87, displays the average salary for each job title.

Figure A–87 Average Salaries Chart



To build the Average Salaries chart:

1. In the Navigator, click the **Providers** tab to bring it forward.
2. At the root level of the Providers tab, click the **Locally Built Providers** link.
3. Click the link for the provider **myCompany_DB_Provider**.
4. Click the **Chart** link next to **Create New...**
5. Click the **Charts From SQL Query** link.
6. In the **Name** field, enter `<YourName>_team_average_salary`.
7. In the **Display Name** field, enter `Average Salaries`.
8. In the **Description** field, enter `average salaries exercise`.
9. In the **Portal DB Provider** list, **MYCOMPANY_DB_PROVIDER** should already be selected.
10. Click **Next**.
11. In the SQL Query field, enter the following code:

    ```
    select
       null     the_link,
       job      the_name,
       avg(sal) the_data
    from emp
    group by job
    ```

    This SQL query will work only if you have SELECT privileges on the EMP table in the SCOTT schema.
12. Click **Next** twice, or until you see the **Display Options** step of the wizard.
13. In both the **Full Page Options** and **Portlet Options** sections, select the values shown in Table A–43:

Table A–43 Average Salaries Chart Full Page and Portlet Display Options

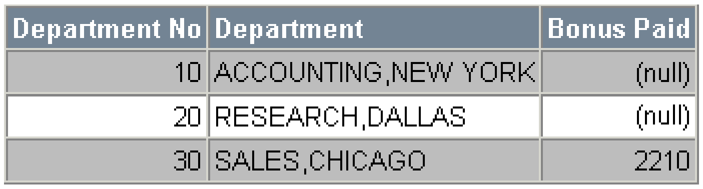
Option	Value
Type Face	Arial
Font Color	Black
Font Size	10 pt
Chart Type	Horizontal
Bar Image	Red Bar (red.gif)

14. Click **Finish**.
15. Click **Run as Portlet** to see what your chart will look like.
16. Close the browser window where the chart is displayed, and click **Close** to close the component manager and return to the Portal Navigator.

A.10.3 Exercise: Building the Team Bonuses Report

The Team Bonuses report, shown in Figure A–88, displays the average bonus paid to employees in each department.

Figure A–88 Team Bonuses Report



Department No	Department	Bonus Paid
10	ACCOUNTING,NEW YORK	(null)
20	RESEARCH,DALLAS	(null)
30	SALES,CHICAGO	2210

To build the Team Bonuses report:

1. In the Navigator, click the **Providers** tab to bring it forward.
2. At the root level of the Providers tab, click the **Locally Built Providers** link.
3. Click the link for the **myCompany_DB_Provider** provider.
4. Click the **Report** link next to **Create New...**
5. Click the **Reports From SQL Query** link.
6. In the **Name** field, enter `<YourName>_team_bonuses`.
7. In the **Display Name** field, enter `Team Bonuses`.
8. In the **Description** field, enter `team bonuses example`.
9. In the **Portal DB Provider** list, **MYCOMPANY_DB_PROVIDER** should already be selected.
10. Click **Next**.
11. In the SQL Query field, enter the following code:

    ```
    select dept.deptno, dept.dname||','||dept.loc, sum(emp.comm)
    from dept, emp
    where dept.deptno = emp.deptno
    group by dept.deptno, dept.dname||','||dept.loc
    ```

    The above query summarizes data from selected columns contained in the DEPT and EMP tables. Although we used a SQL query to build the report, we just as easily could have built it using the Reports Query Wizard.

    This SQL query will work only if you have SELECT privileges on the DEPT and EMP tables in the SCOTT schema.
12. Click **Next**.
13. Select **Tabular**.
14. Click **Next**.
15. Next to each column, enter the **Column Heading Text** as indicated in Table A–44:

Table A–44 Team Bonuses Report Column Heading Text

Column	Column Heading Text
DEPTNO	Department No
DEPT.DNAME||','||DEPT.LOC	Department
SUM(EMP.COMM)	Bonus Paid

    This will add descriptive labels above the columns that appear in your report.

16. Click **Next** twice, or until you see the **Display Options** step of the wizard.

17. For both the **Full Page Options** and **Portlet Options** sections, select the values shown in Table A–45:

Table A–45 Team Bonuses Report Full Page and Portlet Display Options

Option	Value
Heading Font Face	Arial
Heading Font Color	White
Heading Size	10pt
Row Text Font Face	Arial
Row Text Font Color	Black
Row Text Size	10pt
Heading Background Color	Slate Gray
Border	Thin Border

18. Click **Finish**.
19. Click **Run as Portlet** to see what your report will look like.
20. Close the browser window where the report is displayed, and click **Close** to close the component manager and return to the Portal Navigator.

B Troubleshooting OracleAS Portal

This appendix describes common problems that you might encounter when using OracleAS Portal and explains how to solve them. It contains the following topics:

- Problems and Solutions
- Diagnosing OmniPortlet Problems
- Diagnosing Web Clipping Problems
- Need More Help?

B.1 Problems and Solutions

This section describes common problems and solutions. It contains the following topics:

- Java Portlet Wizard Not Available
- Portlet Code Does Not Compile
- Application Server Connection Test Fails
- Provider Test Page Shows Error
- Provider Registration Fails
- Portlet Does Not Display on Page
- After Initial Successful Display, Portlet Does Not Display on Page
- Other Portlet Problems
- Provider Group Not Created
- URL Portlet Does Not Work

B.1.1 Java Portlet Wizard Not Available

In JDeveloper, when you try to open the Java Portlet Wizard by choosing **File > New > Web tier** , the **Portlets** menu selection is not present.

Problem

The Java Portlet Wizard is not installed.

Solution

Install the Java Portlet Wizard, as described in the *Installing the Oracle JDeveloper 10g Portal Add-in* document on Portal Studio:

http://portalstudio.oracle.com/pls/ops/docs/folder/community/pdk/utilities/jdev/jdev.addin.install.guide.html

On Portal Studio (`http://portalstudio.oracle.com`), you can find this document in the Integration/Utilities section.

B.1.2 Portlet Code Does Not Compile

When you try to compile the code for your portlet, a compilation error occurs.

Problem 1

The Portlet Development library is not selected.

Solution 1

Make sure that Portlet Development library is selected for the project, as follows: edit your project's properties and select the **Profiles > Development > Libraries** entry in the pane on the right. Make sure that the Portlet Development library is listed under **Selected Libraries**.

Problem 2

There is a syntax error in the portlet's Java code.

Solution 2

Check the portlet's Java code syntax.

B.1.3 Application Server Connection Test Fails

In JDeveloper, the Application Server Connection Test fails with the message `Connection refused: connect`.

Problem 1

OC4J is not running.

Solution 1

Make sure you OC4J is up and running by trying to access it on its default port, `8888`: type the following URL in your browser: `http://`*`yourhostyourdomain`*`:8888`

Problem 2

The connection information is incorrect.

Solution 2

Verify the connection information that you provided in the Connection Setup wizard.

B.1.4 Provider Test Page Shows Error

When accessing the provider test page with your browser, an error is shown.

Problem 1

The `provider.xml` syntax is incorrect.

Solution 1

Correct the `provider.xml` syntax. Refer to the *PDK-Java XML Provider Definition Tag Reference* document on Portal Studio:

http://portalstudio.oracle.com/pls/ops/docs/folder/community/pdk/jpdk/v2/xml_tag_reference_v2.html

Problem 2

JAR files are missing from the deployment environment.

Solution 2

Locate the missing JAR files and add them to the `classpath`.

B.1.5 Provider Registration Fails

When you register your provider with OracleAS Portal, the provider registration fails with the message:

```
WWC-43176  The provider URL specified may be wrong or the
provider is not running.
```

Problem 1

The provider is down.

Solution 1

Make sure that you can access the provider test page from your browser: enter the provider registration URL in your browser's address bar.

Problem 2

The provider is not accessible from your OracleAS Portal middle tier.

Solution 2

Try to access the provider node from your OracleAS Portal middle tier by pinging it:

```
$> ping providerhost
```

Problem 3

The provider node host name is not recognized.

Solution 3

Replace the provider node host name with the provider node IP address in the provider registration URL.

Problem 4

Provider registration fails due to related circumstances.

Solution 4

Use `logcfg.sql` to gather more diagnostic information.

B.1.6 Portlet Does Not Display on Page

In OracleAS Portal, when you access a portal page, the portlet does not display on the page.

Problem 1

The provider is not running.

Solution 1

Make sure that the provider is up and running by accessing the provider test page from your browser: enter the provider registration URL in your browser's address bar.

Problem 2

The security manager for the provider is preventing the portlet from displaying.

Solution 2

In the provider definition file, `provider.xml`, delete or comment out the security manager, if any.

B.1.7 After Initial Successful Display, Portlet Does Not Display on Page

In OracleAS Portal, when you access the portal page, the portlet initially displays on the page, but returns error messages in subsequent display attempts.

Problem

The provider session information is incorrect.

Solution

Check whether or not the provider uses sessions. If it does, edit the provider registration information to make sure that you registered it accordingly, as follows:

```
Login Frequency: Once per User Sessions
```

B.1.8 Other Portlet Problems

In addition to specific issues listed in the preceding sections, the following information may help you to troubleshoot other portlet problems.

Problem 1

In OracleAS Portal, when accessing the portal page, the portlet does not display or displays an error message.

Solution 1

Check the application log file, located in `J2EE_HOME/application-deployments/`*`web_application_name`*: look for errors.

Problem 2

Pertinent information that may help solve the problem is not being recorded in the log file.

Solution 2

Increase the provider's log level to produce more detailed logging information by adding the following entry to the `orion-web.xml` file:

```
<orion-web-app deployment-version="9.0.4.0.0"
jsp-cache-directory="./persistence" temporary-directory="./temp"
servlet-webdir="/servlet/">
<env-entry-mapping
name="oracle/portal/log/logLevel">6</env-entry-mapping>
</orion-web-app>.
```

You may have to restart your OC4J to ensure that the changes made to the configuration file take effect.

B.1.9 Provider Group Not Created

When you create a new provider group, the provider group is not created.

Problem 1

There is a problem in `mod_osso`.

Solution 1

Check for a `mod_osso` midtier registration problem.

Problem 2

There is a problem in the `jpdk` application `classpath`.

Solution 2

Check the `jpdk` application log for errors.

B.1.10 URL Portlet Does Not Work

Attempting to display your portlet through URL generates the following error message: `500 INTERNAL SERVER ERROR`

Problem

The `httpProxyHost` is not defined correctly.

Solution

In your `provider.xml` file, check the default proxy information. For example:

```
<proxyInfo class=""oracle.portal.provider.v1.http.ProxyInformation"">
@    <httpProxyHost>www-proxy.us.oracle.com</httpProxyHost>
   <httpProxyPort>80</httpProxyPort>
</proxyInfo>
```

Change the `httpProxyHost` and `httpProxyPort` proxy settings or remove these settings if you do not use proxy. Restart the OC4J instance to test if this resolves the problem.

B.2 Diagnosing OmniPortlet Problems

This section provides information to help you troubleshoot problems you may encounter while using OmniPortlet.

To view errors that occur during the execution of OmniPortlet:

- Open the application log file:
  `$OC4J_HOME/j2ee/home/application-deployments/portalTools/application.log`
  or
  `$IAS_HOME/j2ee/`*`OC4J_instance`*`/application-deployments/portalTools/`*`OC4J_instance`*`_default_island_1/application.log`
- Display the HTML source (for example, in Microsoft Internet Explorer brower, choose **View > Source**), and locate the errors embedded in the output HTML as comments.

To alter the logging level of the OmniPortlet Provider:

- Open the `web.xml` file and modify the `context-param` value of `oracle.portal.log.LogLevel` to the possible values ranging from `1` to `7` (where `7` means `debug`). The `web.xml` file is located at:
  ```
  $OC4J_HOME/j2ee/home/applications/portalTools/
    omniPortlet/WEB-INF/web.xml
  ```
  or
  ```
  $IAS_HOME/j2ee/OC4J_instance/applications/portalTools/
    omniPortlet/WEB-INF/web.xml
  ```

The OmniPortlet errors that you are most likely to encounter, and possible solutions, are:

- Chart Not Rendered on UNIX
- Unable to Access HTTPS Site
- OmniPortlet Cannot Access the Specified URL
- Portlet Content Is Not Refreshed

B.2.1 Chart Not Rendered on UNIX

On UNIX, when you select the Layout Style of Chart, the chart is not displayed.

Problem

The `DISPLAY` environment variable is not set up correctly.

Solution

Specify a valid `DISPLAY` setting in either of the following ways:

- Set the `DISPLAY` environment variable environment variable to an X server that you can access: `setenv DISPLAY localhost:0.0java -jar oc4j.jar`
- Or, with JDK 1.4 or above, specify a "headless" display when starting your JVM: `java` **`-Djava.awt.headless=true`** `-jar oc4j.jar`

B.2.2 Unable to Access HTTPS Site

When you access an HTTPS URL, the following error displays:

```
java.lang.NoClassDefFoundError:
at oracle.security.ssl.OracleSSLCipherSuite.isSSLLibDomestic
when accessing HTTPS site with certificate
```

Problem

The required SSL library is not in the library path.

Solution

See the section "Library For HTTPS Access in Standalone OC4J Installation" (under "OmniPortlet Provider Test Page") in the *OracleAS Portal Developer Kit (PDK) Configuring the OmniPortlet Provider* document, which is part of the PDK zip file on OTN at
`http://www.oracle.com/technology/products/ias/portal/pdk.html`
(click the link **Download the PDK Content**).

B.2.3 OmniPortlet Cannot Access the Specified URL

Your OmniPortlet displays errors or does not not display the correct content.

Problem 1

The URL is not active.

Solution 1

Type the URL directly in your browser Address field to test that it is a valid URL.

Problem 2

If a proxy server is required to reach the site, the proxy settings are not valid. The following messages may display:

```
Failed to open specified URL.
Cannot open the URL specified because of connection timeout.
```

Solution 2

Check that your proxy settings are valid: in the OmniPortlet Provider Test Page, click the Edit link for the HTTP Proxy Setting to specify the proxy server settings.

Problem 3

The message `OmniPortlet timed out` displays in your OmniPortlet.

Solution 3

1. If a proxy server is required to reach the site, see Solution 3.
2. If the URL request take a long time to process (for example, if it executes a long running query), try increasing the timeout value (in seconds) in the OmniPortlet `provider.xml` file in the `$IAS_HOME/j2ee/OC4J_instance/applications/portalTools/omniPortlet/WEB-INF/providers/omniPortlet` directory. If you do this, you also need to do the following:
   - Bounce your middle-tier.
   - Increase the provider registration Timeout value of the Portal instances with which this provider is registered.

Problem 4

If HTTP authentication is required, the user name and password are missing or not valid. The following message displays:

```
Authorization failed when connecting to the URL specified.
Provide correct user name and password to connect.
```

Solution 4

To configure the Secured Data (Web Clipping) Repository Setting, click the Edit link in the OmniPortlet Provider Test Page, and configure your settings. Then, in the Source tab, click Edit Connection, and enter a valid username and password.

Problem 5

If opening a URL of an HTTPS site with a certificate, the certificate is identified as not valid with the following error message in the portlet:

```
SSL handshake failed for HTTPS connection to the specified URL.
The certificate file needs to be augmented.
```

Solution 5
See the section "Adding Certificates for Trusted Sites" in the "Securing OracleAS Portal" chapter in the *Oracle Application Server Portal Configuration Guide*.

Problem 6
If the proxy server requires authentication, the user name and password are missing or not valid. The following message displays:

```
Invalid or missing user proxy login information.
```

Solution 6
Check that your proxy server user name and password are valid: see Section B.3.2.2, "Proxy Authentication".

B.2.4 Portlet Content Is Not Refreshed
After changing portlet properties in the edit defaults page, your portlet content is not refreshed.

Problem 1
Web cache invalidation is not configured properly.

Solution 1
See the section "Web Cache" in the *OracleAS Portal Developer Kit (PDK) Configuring the OmniPortlet Provider* documentdocument, which is part of the PDK zip file on OTN at `http://www.oracle.com/technology/products/ias/portal/pdk.html` (click the link **Download the PDK Content**).

Problem 2
You have customized the portlet.

Solution 2
Select the Reset to Defaults option on the Customize page in order for the Edit Defaults changes to appear.

B.2.5 Edit Defaults Changes are Not Reflected in the Customized Portlet
When you customize the portlet at runtime using the Customize link, the new property values are not reflected in the customized version of the portlet.

Problem
When you customize the portlet, a complete copy of the personalization object file is created. Since all properties are duplicated, subsequently modifying the portlet through Edit Defaults will not be reflected in the customized version of the portlet.

Solution
To ensure the latest changes are made to the portlet, you must click **Customize** again (after the modifications through the Edit Defaults wizard are made), then select the **Reset to Defaults** option.

B.3 Diagnosing Web Clipping Problems

This section provides information to help you troubleshoot problems you may encounter while using the Web Clipping provider or Web Clipping Studio:

- Checking the Status of the Provider with the Test Page
- Solving Problems with Connections
- Setting Logging Levels

B.3.1 Checking the Status of the Provider with the Test Page

You can use the Web Clipping Provider Test Page to determine whether or not the provider is functioning properly. To access the Test Page, click **Web Clipping Provider** from the Portal Tools Application Welcome Page, which is located at:

```
http://Hostname:Port/portalTools
```

The Provider Test Page: Web Clipping is displayed. It provides the following information:

- Portlet information: Information about the Web Clipping portlet (The Web Clipping provider contains only one portlet.)
- Provider initialization parameters and values.
- Provider status, with links to pages to edit the configuration.

For more information about using the Test Page, see the "Administering Web Clipping" appendix in the *Oracle Application Server Portal Configuration Guide*.

B.3.2 Solving Problems with Connections

If you encounter difficulties making or maintaining connections to the site to be clipped or a site that was clipped, note the following:

- If a proxy server is needed to connect to HTTP servers outside of a firewall, make sure that the proxy servers are configured correctly. See Section B.3.2.1 for more information.
- If the proxy servers are configured for proxy authentication, you will receive HTTP error code 407 when you attempt to clip a page outside the firewall unless you have manually configured proxy authentication. See Section B.3.2.2 for information about manual configuration.
- If a reverse proxy is used, make sure that the reverse proxy server is configured correctly. See the "Performing Advanced Configuration" chapter of the *Oracle Application Server Portal Configuration Guide* for information about configuring a reverse proxy.
- If you are attempting to add a clip to a Web Clipping portlet, and experience difficulty making or maintaining connections, the cause may be that the configuration includes a load balancer, but the configuration was not set correctly:
  - If multiple OC4J instances are set up behind a load balancer, the Web Clipping Repository and HTTP proxy must be configured to be identical on all OC4J instances before you join the OC4J instances to the Load Balancer.

    Web clippings have definitions that must be stored persistently in the Web Clipping Repository hosted by an Oracle Database server. In a multiple middle-tier environment, all instances of OC4J must be configured to store definitions in the same repository.

  In addition, all instances of OC4J must have identical configurations for the HTTP proxy.

  – The Load Balancer must be session-enabled. If it is not, the first request connects, but subsequent requests, which may be routed to a different instance, fail.

  For more information about configuring with a load balancer, see the "Performing Advanced Configuration" chapter of the *Oracle Application Server Portal Configuration Guide*.

- To be sure a URL is correct, test the URL that you want to clip in a browser before you attempt to clip it. Also test the URL from the provider middle tier to be sure that it is accessible from there.
- If you cannot clip a page, make sure that the page is not overpopulated with IFrames. View the page in a browser, looking at the page source. If it contains IFrames, start with the URL pointed to by the IFrame "src" attribute.
- If images in a clipping are not retrieved when the rest of the clipping is retrieved, check your browser proxy settings. Because images are treated as links (using the "src" attribute of the IMG tag), images from clipped sites are served directly from the original sites. If the images required that the proxy setting be enabled during creation of the clipping or Show mode, disabling the browser proxy setting disables viewing of the images in a clipping. Enable the browser proxy setting.
- If you cannot connect, check the error log. (See Section B.3.3 for information about the error log.) Check to see if the log contains a message about logon to the database being denied. If this is the case, the PORTAL schema password in the infrastructure database may have been modified manually and no longer matches the password stored in Oracle Internet Directory. Refer to the *Oracle Internet Directory Administrator's Guide* for more information about setting the password.

B.3.2.1 Configuring Proxy Servers

If a proxy server is needed to connect to HTTP servers outside of a firewall, make sure that the proxy servers are configured correctly.

To configure the proxy servers, go to the Web Clipping Provider Test Page, as described in Section B.3.1. In the Web Clipping Provider Test Page, click **Edit** in the **Actions** column of the **HTTP Proxy** row. In the Edit Provider page, specify the **HTTP Proxy Host** and the **HTTP Proxy Port** for the HTTP Proxy.

For access to servers that are inside the firewall, you can specify a list of domain names that do not require going through the firewall by selecting **No Proxy for Domains beginning with** and entering the URL. You do not need to restart OC4J for the new settings to take effect.

For more information about configuring proxy servers, see the "Administering Web Clipping" appendix of the *Oracle Application Server Portal Configuration Guide*.

B.3.2.2 Proxy Authentication

Web Clipping supports global proxy authentication, and per-user authentication. You can specify the realm of the proxy server and whether all users will automatically log in using a user name and password you provide, each user will log in using an individual user name and password, or all users will log in using a specified user name and password. For more information about configuring proxy authentication, see the "Administering Web Clipping" appendix of the *Oracle Application Server Portal Configuration Guide*.

B.3.3 Setting Logging Levels

By default, the logging level of Web Clipping is set to level 3, which provides information about configuration, severe errors, and warnings. This is a reasonable level for day-to-day operation. To view information that is useful for debugging, set the logging level to 7.

To set the logging level, edit the `web.xml` file and specify the level for the `oracle.portal.log.LogLevel` parameter. The `web.xml` is located at:

```
OracleHome/j2ee/OC4J_Portal/applications/portalTools/webClipping/WEB-INF
```

For example, to set the level to display debugging information, set the value of the parameter `oracle.portal.log.LogLevel` to 7, as shown in the following example:

```
<context-param>
        <param-name>oracle.portal.log.LogLevel</param-name>
        <param-value>7</param-value>
</context-param>
```

Errors that occur when accessing the Test Page or during the execution of the Web Clipping portlet are written to one of the following files:

```
OC4J_HOME/j2ee/home/application-deployments/portalTools/application.log
IAS_HOME/j2ee/OC4J_instance/application-deployments/portalTools/OC4J_instance_
default_island_1/application.log
```

B.4 Need More Help?

You can find more solutions on Oracle *MetaLink*, `http://metalink.oracle.com`. If you do not find a solution for your problem, log a service request.

See Also:

- *Oracle Application Server Release Notes*, available on the Oracle Technology Network: `http://www.oracle.com/technology/documentation/ias.html`

Glossary

About mode

An optional **portlet show mode** that displays information about the portlet's copyright, version, and author.

access control list

See **ACL**.

ACL

Access Control List. A list of groups and users authorized for specific access to an object.

advanced search

A search engine that enables users to:

- Find content that contains any or all terms in the search string.
- Search selected **page group**s, or search across all page groups.
- Restrict the search to a particular **page**, **category**, **perspective**, **item type**, or **attribute**.

If **Oracle Text** is installed and enabled, advanced search can also be used to perform near, soundex, and fuzzy searches.

See also **search portlet**. Contrast with **basic search** and **custom search**.

API

Application Programming Interface. A set of exposed data structures and functions that an application can use to invoke services on a **portlet**, **page**, or **page group**.

Note that OracleAS Portal APIs are exposed through the **PDK** available on **Portal Center** (`http://www.oracle.com/technology/products/ias/portal/`).

application

Obsolete terminology. See **database provider**.

Application Programming Interface

See **API**.

Application Service Provider

See **ASP**.

approval notification

A message in the Notification portlet indicating that the creation or update of an **item** requires an approval. Approval notifications are sent to the list of approvers identified in an **approval process**. An approver may respond to the notification by approving or rejecting the item in question.

approval process

A series of one or more steps in which a newly created or updated **item** must be approved before it can be published. Each step in an approval process must have one or more approvers, where each approver is either a user or a **group**. Routing to the approvers can be in serial (one at a time) or in parallel (all at once), and each step can be defined to require a response (either an approval or a rejection) by any one member or by all members. Once the required number of responses is received during a step, the process continues to the next step. The process ends when the item is rejected, or the final step is reached and the document is approved.

ASP

Application Service Provider. Provides remote hosting of applications, maintaining and operating the hardware, software, and other resources required to run the applications. A good example is Oracle Portal Online (`http://portal.oracle.com`), a hosted subscription service that provides the features of OracleAS Portal to smaller businesses and organizations who want to build **portal**s but do not have the resources in-house to build and manage them.

attribute

Stores information (or metadata) about an **item** or **page**: for example, Create Date, Expire Date, or Author. **page group administrator**s can create custom attributes to extend the functionality of **item type**s and **page type**s. For example, a base attribute on a file is Display Name; a custom attribute might be a check box to indicate whether the file is confidential. Custom attributes are useful for assigning unique, searchable identifiers to items.

authenticated user

User who is logged on to OracleAS Portal. By default, authenticated users can access and, based on privileges granted to the user, act on certain OracleAS Portal objects, such as **page**s.

Contrast with **public user**s, who can access only public content.

authorization

The evaluation of security constraints to send a message or make a request. Authorization uses specific criteria—authentication and restriction—to determine whether the request should be permitted.

authorized user

See **authenticated user**.

banner

See **region banner**. See also **navigation page**.

base attribute

See **attribute**.

base item type

See **item type**.

base page type

See **page type**.

basic search

Enables users to find content that contains a specific search string.

See also **search portlet**. Contrast with **advanced search** and **custom search**.

basic search box item type

A **navigation item type** that a user can add to a **page** to allow other users to search. You can specify whether users of the search box can search all **page group**s or only in selected page groups.

batch job

Running an OracleAS Portal **portlet** in the background using the OracleAS Portal batch job facility. An end user can run a portlet in batch mode by selecting options on the portlet's **customization form**. Batch processing is useful if the portlet is based on a large amount of data, if the portlet displays many rows of data, or if the job may take a long time to run.

bind variable

Variable in a SQL statement that must be replaced with a valid value or address of a value in order for the statement to execute successfully. Portlet developers typically use bind variables (for example, dept) to display a **parameter entry field** in an OracleAS Portal **portlet**'s **customization form**. The entry field enables end users to choose the data that the portlet will display.

bookmark

See **favorite**.

breadcrumbs

See **page path item type**.

Builder page

See **Portal Builder page**.

bulk load

See **zip file item type**.

caching

The act of storing frequently accessed information, typically Web pages or **portlet**s in OracleAS Portal, in a location where it can be accessed quickly, to avoid frequent content generation. For example, **Oracle Application Server Web Cache** stores dynamically-generated portlets in its memory, then serves them to the **PPE** when there is a request for the specified portlet. This storage reduces the total time spent handling the request by avoiding connections to the back-end database and other Web sites.

See also **expiry-based caching**, **invalidation-based caching**, **system level caching**, and **validation-based caching**.

calendar

An OracleAS Portal **portlet** that displays the results of a SQL **query** in calendar format.

call interface

Displays the arguments that were selected when an OracleAS Portal **portlet** was originally created or last edited.

category

A predefined **attribute** used to group or classify **page**s, **item**s, and **portlet**s in a **page group**. A category helps users answer the question "What is this item or page?" For example, in a travel page group, you might have categories for maps, snapshots, and hotel reviews. Only one category can be assigned to a particular item or page.

Contrast with **perspective**.

chart

An OracleAS Portal **portlet** that displays the results of a SQL **query** as a chart, such as a bar chart, pie chart, or line chart. Charts are based on at least two table or view columns: one that identifies the bars on the chart and another that calculates the size of the bars on the chart.

check-out/check-in

See **document control**.

child object

An object which is part of a hierarchy. For example, sub-pages, sub-categories, and sub-perspectives are child objects of a **page**, **category**, and **perspective** respectively.

See also **manifest**.

CHTML

Compact HTML. A subset of **HTML** recommendations, designed for small devices.

classification

Categories and perspectives are used to classify the content of a **page** so that it is easy for users to locate that content during a search.

See also **category** and **perspective**.

cluster

A database **object** used to store tables that are related to one another and that are often joined together in the same area on a disk.

community

See **Portal Community**.

compact HTML

See **CHTML**.

component

Obsolete terminology. See **portlet**.

content area

Obsolete terminology. See **page group**.

content contributor

User who has the appropriate privileges to add **item**s to a **page**. Appropriate page privileges include Manage Content and Manage Items With Approval. Appropriate item privileges include Manage, Edit, and View.

content item type

A means of identifying the actual content of an **item** that is being uploaded to a **page**, such as a document, text, or an image.

Built-in content item types are:

- **file item type** and **simple file item type**
- **text item type** and **simple text item type**
- **URL item type** and **simple URL item type**
- **image item type** and **simple image item type**
- **image map item type**
- **PL/SQL item type** and **simple PL/SQL item type**
- **page link item type** and **simple page link item type**
- **zip file item type**

See also **item type**. Contrast with **navigation item type**.

CSS

Cascading Style Sheet.

current version

The version of an **item** that is displayed on the **page**. The current version is not necessarily the most recent version of the item.

See also **versioning**.

custom attribute

See **attribute**.

custom item type

See **item type**.

customization form

Page that prompts end users for values to pass to an OracleAS Portal **portlet**. End users can view the customization form for a portlet, if one has been created, by clicking the portlet's **Customize** link.

custom page type

See **page type**.

custom provider

A type of **provider** that enables you to create and maintain **portlet**s that access customer-specific content or applications. You can build custom portlets in OracleAS Portal either declaratively or programmatically.

custom search

Enables users to define a variety of searches against information stored in the OracleAS Portal schema of the **Oracle Application Server Metadata Repository**. By editing the default customizations of the Custom Search portlet, you can define unique search submission forms and results pages that meet the specific search requirements, or configure portlets that execute and return results based on predefined search criteria.

See also **search portlet**. Contrast with **advanced search** and **basic search**.

DAD

Database Access Descriptor. A set of values that specify how an application connects to an Oracle database to fulfill an HTTP request. The information in the

DAD includes the user name (which also specifies the schema and the privileges), password, connect-string, error log file, standard error message, and Globalization Support parameters such as NLS language, NLS date format, NLS date language, and NLS currency.

Database Access Descriptor

See **DAD**.

database administrator

See **DBA**.

database object

See **object**.

database provider

A type of **provider** that is written as a PL/SQL **stored procedure** and is used to create **portlet**s that reside in the database. One example of a database provider is a provider built using the wizards in OracleAS Portal to provide form, report, and chart portlets.

Contrast with **Web provider**.

data-driven portlet

You can use data-driven **portlet**s to display, create, or update data stored in the objects of an Oracle database. OracleAS Portal provides build wizards for creating data-driven portlets. The build wizards produce PL/SQL procedures that are stored in the database.

data portlet

An OracleAS Portal **portlet** that displays data in a spreadsheet format.

DAV

See **WebDAV**.

DBA

Database Administrator. User belonging to the DBA **group**. By default, members in the DBA group have access to all OracleAS Portal product pages, and have the Manage privilege for all **page**s, **page group**s, **database provider**s, and administration.

default subscriber

The base **subscriber** that is installed along with the install of OracleAS Portal.

de-militarized zone

See **DMZ**.

developer

Builds portlets for others to include on their pages. Relies heavily on the APIs to extend the capabilities of OracleAS Portal; may frequently consult the **Portal Knowledge Exchange** or the forums for advice or inspiration.

Developer Services

See **Portal Developer Services**.

DIP

Directory Integration Platform. The provisioning platform provided by **OID** to synchronize different directories and directory enabled applications.

direct access URL

A feature that enables a user to directly access or bookmark an OracleAS Portal **object** (for example, a **page**, **category**, **perspective**, or document) using a **URL** that contains the object type and name. Direct access URLs are typically shorter and more traditional in appearance since the URL does not contain parameters, for example:

```
http://mymachine.mycompany.com:5000/pls/portal/url/page/toplevelsite
```

Contrast with **dynamic URL**. See also **durable link**.

Directory Information Tree

See **DIT**.

Directory Integration Platform

See **DIP**.

display name

An **object**'s external name used throughout OracleAS Portal, for example, in the **Navigator**, on **page**s, and in the page editor. When the object is published as a **portlet**, the display name is used as the title of the portlet in the **Portlet Repository**.

Distinguished Name

See **DN**.

DIT

Directory Information Tree. A hierarchical tree-like structure in Oracle Internet Directory (**OID**) consisting of the **DN**s of the entries.

DN

Distinguished Name. The unique name of a directory entry in Oracle Internet Directory (**OID**). It includes all the individual names of the parent entries back to the root. The Distinguished Name tells you exactly where the entry resides in the directory's hierarchy. This hierarchy is represented by a directory information tree (**DIT**).

DMZ

De-militarized Zone. Computer host or small network inserted as a "neutral zone" between a company's private network and the outside public network. It prevents outside users from getting direct access to a server that has company data. A DMZ is an optional and more secure approach to a **firewall** and effectively acts as a **proxy server** as well. (The term comes from the geographic buffer zone that was set up between North Korea and South Korea following the UN police action in the early 1950s.)

document control

Allows users to check-out an item so that other users cannot edit that item, thus preventing users from overwriting each others changes. When the user has finished editing the item, he or she checks the item back in, making it available again for other users to edit.

DR

Disaster Recovery.

durable link

A link to an item that uses the item's GUID (globally unique ID) to uniquely identify it. An item's GUID does not change so its durable link will not break when the item is edited, renamed, moved, or imported to a different portal instance.

Contrast with **direct access URL**.

dynamic page

An OracleAS Portal **portlet** that displays dynamic content on a page. The dynamic page build wizard enables you to specify one or many PL/SQL blocks within HTML code to create a page. This code executes every time an end user requests the page.

dynamic URL

A URL that contains a query string (one or more parameters and the characters ? and &).

Contrast with **direct access URL**

edge side includes

See **ESI**.

Edit mode

Page editing: Edit mode enables an authenticated user with appropriate privileges to set **page** properties and to add, modify, or delete **portlet**s and **item**s on the page. To switch to Edit mode, the user clicks an **Edit** link on the page. There are three Edit mode views: **graphical view**, **layout view**, and **list view**.

See also **Mobile Preview mode** and **Pending Items Preview mode**.

Portlets: An optional portlet **show mode** that enables personalization of the portlet on a per user, per instance basis.

Contrast with **Edit Defaults mode**.

Edit Defaults mode

An optional **portlet show mode** that enables administrators to set the defaults of a portlet for all users.

Contrast with **Edit mode**.

Enterprise Manager

Oracle Enterprise Manager 10*g* A component of the **Oracle Application Server** that enables administrators to manage Oracle Application Server services through a single environment. For example, an administrator can use Enterprise Manager to monitor the services that make up an OracleAS Portal instance, including **HTTP** services, **mod_plsql** services, the **Oracle Application Server Single Sign-On Server**, the **PPE**, the Oracle database, and **provider**s.

enterprise portal

Enterprise **portal**s are common, integrated starting points that provide personalized access to relevant enterprise information sources. Enterprise portals enable site visitors to customize their view of the resources available on the public Internet.

ESI

Edge Side Includes. A markup language to enable partial page caching (**PPC**) of HTML fragments.

event

See **page event**.

Event servlet

The Event servlet implements the functionality of OracleAS Portal to allow for dynamic page navigation when accessing an event enabled **portlet**. The Event servlet runs in the same container as the **PPE**.

expiration period

Number of days after which, or an exact date on which, an **item** expires. After an item expires, it is viewable only by the item's or **page**'s owner and the **page group administrator** in **Edit mode**. Expired items are removed from the database during a **system purge** of all expired items.

expiry-based caching

A **caching** method that uses a retention period to specify how long the item is valid in the cache, before a refresh is required. Pages that use expiry-based caching may also be cached in the user's browser.

See also **invalidation-based caching** and **validation-based caching**.

expiry notification

A message automatically sent to a user or **group** indicating that an **item** on the **page** is about to expire. The notification is set up by the **page group administrator**.

explicit object

Export object category. An object which is explicitly selected, from the Navigator or Bulk Actions for export.

See also **manifest** and **referenced object**.

export

A method of creating a set of files (**transport set**) that contains **page groups**, **pages**, **portlets**, and other content from a single OracleAS Portal instance. You can then **import** this set of files into another Oracle Application Server instance.

eXtensible Markup Language

See **XML**.

external application

Application external to OracleAS Portal that is typically launched from the External Applications **portlet**. As each external application is configured by a **portal administrator**, users simply supply their user name and password information. The **Oracle Application Server Single Sign-On Server** will present these credentials for future authentication challenges.

external object

Export object category. An object which is an external dependency of an **explicit object**. External objects ensure that the explicit objects perform on the target portal.

See also **manifest**.

favorite

A hyperlink in the Favorites **portlet** that provides quick access to a frequently visited **URL**, either inside or outside your company firewall. An **authenticated user** can customize the Favorites portlet with his or her own preferred set of frequently accessed URLs.

favorite group

A collection of **favorites** (and favorite groups) that are usually logically related.

Federated Portal Adapter

See **FPA**.

file item type

One of the default **item types** that a **content contributor** can add to a **page**. When a user adds a file item to a page, the file is uploaded into the OracleAS Portal schema of the **Oracle Application Server Metadata Repository** and displayed as a hyperlink on the page. When a user clicks the **display name** link, the files may be downloaded to the user's computer or displayed in the user's Web browser, depending on the file type and the configuration of the browser.

firewall

A machine that acts as an intermediary to protect a set of computers or networks from outside attack. It regulates access to computers on a local area network from outside, and regulates access to outside computers from within the local area network. A firewall can work either by acting as a **proxy server** that forwards requests so that the requests behave as though they were issued by the firewall machine, or by examining requests and attempting to eliminate suspect calls.

folder

Obsolete terminology. See **page**.

form

An OracleAS Portal **portlet** that provides a transactional interface to one or more database tables, views, or procedures. For example, you can use tools in OracleAS Portal to build a form for entering new employee information into your Human Resources database.

See also **master-detail form**.

FPA

Federated Portal Adapter. The Federated Portal Adapter is a module in the portal instance (written in both Java and PL/SQL) that receives **SOAP** messages for a **Web provider**, parses the SOAP, and then dispatches the messages to a **database provider** as PL/SQL procedure calls. In effect, the Federated Portal Adapter makes a database provider behave exactly the same way as a Web provider, allowing users to distribute their database providers across database servers. All remote providers can be treated as Web providers, hiding their implementation (database or Web) from the user. The most common use is to share database providers (including page groups) owned by one portal instance among other portal instances.

frame driver

An OracleAS Portal **portlet** consisting of a Web page divided into two frames. A driving frame contains a SQL **query** that drives the contents of the second (target) frame.

Full Screen mode

An optional **portlet show mode** that provides more content than can be shown in the portlet when it is sharing a page with other portlets.

function

PL/SQL subprogram that performs a specified sequence of actions and then returns a value. Functions are usually small, very specific blocks of code written to perform a specific task within the scope of a larger application.

In a **page**, end users execute functions by clicking the title of a PL/SQL or custom item.

gist

An **Oracle Text** summary consisting of the document paragraphs which best represent the overall subject matter. You can use such summaries to skim the main content of the text or assess your interest in the text's subject matter.

global privilege

A **privilege** that grants a certain level of access to a user or **group** on all **object**s of a particular type. For example, you could grant a Web Designer group Manage privileges on all **style**s.

grantee

User who is given privileges on an **object** by another user.

graphical view

Page editing view that renders **page** content in-place on the page. Graphical view enables you to view pages and **item**s as they appear on the finished page as you edit.

Contrast with **layout view** and **list view**.

group

Collection of OracleAS Portal users who typically share a common need or interest; for example, Human Resources, Accounting, and so on. Groups make it easy to grant access to an **object** (such as a **page** or **portlet**) to several users at once, You can also use groups to implement user roles by assigning role-related privileges to a group, then adding users in that role.**OID** tracks the membership of OracleAS Portal groups.

group owner

User who has the privilege to add or delete members from a **group**, or to delete the group itself. Groups can have more than one owner.

HA

High Availability.

Handheld Device Markup Language

See **HDML**.

HDML

Handheld Device Markup Language. A simple language to define hypertext-like markup content and applications for handheld devices with small display.

Help mode

An optional **portlet show mode** that displays usage information about the functionality of the portlet.

hierarchy

An OracleAS Portal **portlet** that displays data from a self-referencing table or view. At least two columns in the table must share a recursive relationship. A hierarchy can contain up to three levels and display data such as employees in an organization chart or the hierarchical relationship between menus in a Web site.

home page

The **page**, defined within OracleAS Portal, that typically displays when logging on or when a user selects the Home **smart link item type**. The **portal administrator** chooses this page for **public user**s; **authenticated user**s may choose their own. If the portal administrator enables **mobile page** design, he or she can specify a separate mobile home page to display when the portal is accessed from a mobile device.

hosted site

See **stripe**.

HTML

Hyper Text Markup Language. A format for encoding hypertext documents that may contain text, graphics, and references to programs and other hypertext documents.

HTTP

Hyper Text Transfer Protocol. The underlying format, or protocol, used across the Web to format and transmit messages and determine what actions **Web server**s and browsers should take in response to various commands. HTTP is the protocol typically used between **Oracle Application Server** and its clients.

Hyper Text Markup Language

See **HTML**.

Hyper Text Transfer Protocol

See **HTTP**.

IDE

Integrated Development Environment. A visual tool containing editors, debuggers, screen painters, object browsers, and the like.

ILS

Item Level Security. Mechanism that controls granular access to **item**s on a given **page**. ILS authorizes item managers to grant explicit item access to users and **group**s that take precedence over page-level privileges.

image item type

One of the default **item type**s that a **content contributor** can add to a **page**. You can add images in JPEG, GIF, or PNG formats.

image map item type

One of the default **item type**s that a **content contributor** can add to a **page**. An image map is a single image with hotspots that, when clicked, link to other **URL**s. For example, you can create an image map of the world each continent is hyperlinked to more information about the continent.

import

A method of transporting content and **object**s (for example, **page groups**, **page**s, and **portlet**s) into an OracleAS Portal instance. For example, you can import a page, its associated **style**, and its contents from one instance of OracleAS Portal to another.

index

Optional structure associated with a table used to locate rows of the table quickly, and (optionally) to guarantee that every row is unique.

Integrated Development Environment

See **IDE**.

internal image name

The name used to identify an image that has been uploaded to the portal. Uploaded images can be reused within the portal by referencing their internal names.

internal provider

A type of **provider** that makes **page group object**s (**page**s, **navigation page**s, etc.) available to instances of OracleAS Portal.

invalidation-based caching

A **caching** method used by **Oracle Application Server Web Cache**, where an item remains in the cache until some event occurs that requires it to be refreshed. For example, a user may update an item, requiring the cache to be updated. In response to the event, the portal or a **provider** sends an invalidation message to Oracle Application Server Web Cache. The next time there is a request for the invalidated item, it is refreshed in the cache.

Any data saved in Oracle Application Server Web Cache is considered valid until it is invalidated or it expires. When the information cached in Oracle Application Server Web Cache becomes inaccurate, it must be invalidated. The **page metadata** saved in Oracle Application Server Web Cache is invalidated, for example, if the page designer changes the page structure or when the user's privileges change. A **portlet** may become invalidated when the end user customizes it.

See also **expiry-based caching** and **validation-based caching**.

item

An individual piece of content (text, hyperlink, image, etc.) that resides on a **page** in an item **region**. Users with an appropriate privilege level can add items to a page. Item content and metadata are stored in the OracleAS Portal schema of the **Oracle Application Server Metadata Repository**. Items are rendered on the page according to the layout, **style**, and **attribute** display defined for the item region.

See also **item type**.

item ID

Local database reference to the content of an **item**. An item ID value is used in custom **item type**s to pass items to PL/SQL procedures. The function uses the item ID to access the content of the item.

item level security

See **ILS**.

item type

Defines the contents of an **item** and the **attribute**s that are stored (metadata) about an item. Item types are categorized as **content item type**s and **navigation item type**s.

Custom item types can be created by **page group administrator**s to extend the functionality provided by item types and store additional attribute information about items.

item versioning

See **versioning**.

J2EE

Java 2 Platform, Enterprise Edition. This platform enables application developers to develop, deploy, and manage multitier, server-centric, enterprise level applications. The J2EE platform offers a multitiered distributed application model, integrated XML-based data interchange, a unified security model, and flexible transaction control. You can build your own J2EE **portlet**s and expose them through **Web provider**s.

See also **OC4J**.

J2SE

Java 2 Platform, Standard Edition. This platform enables application developers to develop, deploy, and manage Java applets and applications on a desktop client platform such as a personal computer or workstation. J2SE not only defines **API** standards, but also specifies the deployment of enterprise applications, thus enabling application server administrators to perform the deployment regardless of the vendor of the J2SE server.

See also **OC4J**.

Java 2 Platform, Enterprise Edition

See **J2EE**.

Java 2 Platform, Standard Edition

See **J2SE**.

JavaScript

Scripting language developed by Netscape that enables generation of **portlet**s that introduce dynamic behavior in otherwise static **HTML**. OracleAS Portal enables you to use JavaScript to create routines that validate entry fields in **forms** and **customization forms**. You can also create JavaScript event handlers for entry fields and buttons on forms.

Java Specification Request

See **JSR 168**.

JavaServer Page

See **JSP**.

JSP

JavaServer Page. An extension to **servlet** functionality that provides a simple programmatic interface to Web pages. JSPs are **HTML** pages with special tags and embedded Java code that is executed on the Web or application server. JSPs provide dynamic functionality to HTML pages. They are actually compiled into servlets when first requested and run in the servlet container.

See also **JSP tags**.

JSR 168

Java Specification Request (JSR) 168. Defines a set of APIs for building standards-based portlets using Java. Portlets built to this specification can be rendered to a portal locally or deployed to a WSRP container for rendering portlets remotely. For more information, see `http://jcp.org/en/jsr/detail?id=168`. JSR 168 is not supported in OracleAS Portal 10.1.2.

JSP tags

Tags that can be embedded in **JSPs** to enclose Java code. These tags use the `<jsp:` syntax and enclose action elements in the JSP with `begin` and `end` tags similar to **XML** elements.

keyword

An **attribute** used to provide additional information about a **page** or **item** so that users can easily locate the page or item during a search.

layout view

Page editing view that enables you to add, arrange, and remove **regions** on the **page**. You can also hide, show, delete, or move content in this view.

Contrast with **graphical view** and **list view**.

LBR

Load-balancing router. A very fast network device that distributes Web requests to a large number of servers. It provides portal users with a single published address, without them having to send each request to a specific **middle tier** server.

LDAP

Lightweight Directory Access Protocol. A standard for representing and accessing user and group profile information.

level

Used to provide structure to **mobile page**s and as a way to limit the amount of content displayed on the smaller screens of mobile devices. Users drill down into the levels on a mobile page to view more content.

library

Collection of one or more PL/SQL or Java program units. Libraries can be referenced by several applications simultaneously.

Lightweight Directory Access Protocol

See **LDAP**.

Link mode

An optional **portlet show mode** that enables portlets to render themselves on mobile devices, such as cellular telephones.

list view

Page editing view that displays a listing of all **page** content and provides options that enable you to perform actions (delete, move, copy, etc.) on multiple **object**s.

Contrast with **graphical view** and **layout view**.

list of objects item type

A **navigation item type** that a user can add to a **page**. This item type is a list of objects (for example, pages and **perspective**s) that users can choose to display as a drop-down list or as links (with or without associated images).

list of values

See **LOV**.

load-balancing router

See **LBR**.

local provider group

The collection of **provider**s that are defined within an instance of OracleAS Portal. Provider groups make it easier to share providers defined or registered within one instance of OracleAS Portal with other OracleAS Portal instances.

See also **provider group**. Contrast with **remote provider group**.

lock

Setting automatically applied to an OracleAS Portal **portlet** when it is being edited. The setting prevents other users from editing the portlet.

In **WebDAV** the action of preventing other users from editing a file. Locking a file in a WebDAV client checks out the corresponding **item** in the portal itself.

login/logout link item type

A **navigation item type** that a user can place on a **page** to enable other users to log in or log out of the portal.

LOV

List of values. An OracleAS Portal **portlet** that enables developers to add selectable values to entry fields in **form**s. A single list of values can be displayed in different formats, such as combo boxes, radio buttons, or check boxes.

manifest

The list of objects in a **transport set** and their dependents, also provides a granular level of control over the import mode.

master-detail form

An OracleAS Portal **portlet** that displays a master table row and multiple detail rows within a single HTML page. Values in the master row determine which detail rows are displayed for querying, updating, inserting, and deleting.

See also **form**.

menu

An OracleAS Portal **portlet** that displays a Web page containing options that end users can click to navigate to other menus, other OracleAS Portal portlets, or **URL**s.

middle tier

Part of the OracleAS Portal architecture that handles **HTTP** user requests by forwarding them to the appropriate Portal database or **provider**, assembles Portal **page**s, and manages **caching** of Portal content.

mobile page

A **page type** that enables page creators to produce **page**s specifically for mobile devices, for example, cellular phones.

Mobile Preview mode

A preview mode that enables you to preview how your page will look on a mobile device.

mobile XML

See **Oracle Application Server Wireless XML**.

mod_oc4j

The **Oracle HTTP Server** module that manages the communication between the Oracle HTTP Server and **OC4J**.

mod_plsql

The **Oracle HTTP Server** module that handles the database connections made from the Oracle HTTP Server. It enables PL/SQL database procedures to generate **HTTP** responses containing formatted data and **HTML** code that can display in a Web browser.

navigation item type

Used to provide navigation and to access or execute portal-specific functions.

Built-in navigation item types include:

- **smart link item type**
- **smart text item type**
- **login/logout link item type**
- **basic search box item type**
- **list of objects item type**
- **object map link item type**
- **page path item type**

- **page function item type**

See also **item type**. Contrast with **content item type**.

navigation page

A special purpose **page** within a **page group** that is typically embedded on other pages or **page template**s to implement standard user interface effects such as navigation bars and banners. Often contains **navigation item type**s for navigation within the portal.

Navigator

A feature for locating **object**s and interacting with OracleAS Portal. Provides access to **object**s to which the user has privileges, such as **page group**s, **provider**s, and database objects.

non-default subscriber

A **subscriber** that has a **stripe** on a hosted OracleAS Portal provided by an **ASP**.

object

1. OracleAS Portal object: A structure such as a **page group**, **portlet**, **page**, or **style**.
2. Database object: An Oracle database structure such as a table, procedure, or trigger. These objects can be created using OracleAS Portal wizards or Oracle database commands.

object map link item type

A **navigation item type** that a user can add to a page to display a map of **object**s that are available in the portal.

OC4J

Oracle Application Server Containers for J2EE. The **J2EE** server component of **Oracle Application Server** written entirely in Java that executes on the standard Java Development Kit (JDK) Virtual machine (Java VM). It includes a **JSP** Translator, a Java **servlet** container, and an Enterprise JavaBeans (JB) container.

OID

Oracle Internet Directory. The repository for storing OracleAS Portal user credentials and **group** memberships. By default, the **Oracle Application Server Single Sign-On Server** authenticates user credentials against Oracle Internet Directory information about dispersed users and network resources. Oracle Internet

Directory combines LDAP version 3 with the high performance, scalability, robustness, and availability of the Oracle database.

OmniPortlet

A Web provider that provides portlets that can display spreadsheet, XML, and Web Service data as tabular, chart, news, bullet, and form layouts.

OPCA

OracleAS Portal Configuration Assistant. A Java-based configuration tool for installing and configuring OracleAS Portal. In a typical **Oracle Application Server** installation, the OPCA is executed by the Oracle Universal Installer (OUI) in the post-installation phase. The OPCA can also be invoked in the standalone mode.

OPPI

(No longer in use) OPPI referred to "OracleAS Portal Partner Initiative" which no longer exists. This Initiative was folded into the standard Oracle Partner Network (OPN). Partners can declare a focus on Oracle Application Server and an interest in OracleAS Portal when they sign up for OPN, but there is no separate program just for these partners.

Oracle Application Server

Oracle's integrated application server:

- Standards compliant (**J2EE**, Web Services, and **XML**)
- Delivers a comprehensive set of capabilities, including portal, **caching**, wireless, integration, and personalization
- Provides a single, unified platform for Java and J2EE, Web Services, XML, SQL, and PL/SQL

Oracle Application Server Containers for J2EE

See **OC4J**.

Oracle Application Server Metadata Repository

An Oracle database that contains schemas and business logic used by application server components (including OracleAS Portal) and other pieces of the infrastructure.

OracleAS Portal uses a schema within the Oracle Application Server Metadata Repository to store and manage the content and metadata associated with the portal instance.

Oracle Application Server Portal

A component of **Oracle Application Server** used for the development, deployment, administration, and configuration of enterprise class **portal**s. OracleAS Portal incorporates a portal building framework with self-service publishing features to enable you to create and manage information accessed within your portal.

Oracle Application Server Portal Developer Kit

See **PDK**.

Oracle Application Server Single Sign-On Server

A component of **Oracle Application Server** that enables users to log in to all features of the Oracle Application Server product suite, as well as to other Web applications, using a single user name and password. OracleAS Portal is integrated with Oracle Application Server Single Sign-On Server as a **partner application** and delegates authentication to it.

Oracle Application Server Web Cache

A component of **Oracle Application Server** that improves the performance, scalability, and availability of frequently used Web sites. By storing frequently accessed URLs in memory, Oracle Application Server Web Cache eliminates the need to repeatedly process requests for those URLs on the Web server. Oracle Application Server Web Cache uses **invalidation-based caching** and is integrated with OracleAS Portal for improved performance.

See also **portal cache**.

Oracle Application Server Wireless

A component of **Oracle Application Server** used to deliver information and applications to mobile devices. Using Oracle Application Server Wireless, you can create custom portal sites that use different kinds of content, including Web pages, custom Java applications, and **XML**-based applications. Oracle Application Server Wireless sites make this diverse information accessible to mobile devices without you having to rewrite the content for each target device platform.

Oracle Application Server Wireless XML

Device independent markup language used for communication between OracleAS Portal and **Oracle Application Server Wireless**.

Oracle Application Server

See **Oracle Application Server**.

OracleAS Portal

See **Oracle Application Server Portal**.

OracleAS Portal Configuration Assistant

See **OPCA**.

OracleAS Portal Partner Initiative

See **OPPI**.

OracleAS Portal Verification Service

(Previously known as Portal Studio). A major component of Portal Center (`http://www.oracle.com/technology/products/ias/portal`) that is specifically tuned to the needs of the portal developer. This site provides developers a way to test and display remote portlets exposed as Web or WSRP providers without having to install their own copy of OracleAS Portal. Developer's portlets reside on their servers and must be accessible over the internet. You can access OracleAS Portal Verification Service directly at `http://portalstandards.oracle.com`.

Oracle Enterprise Manager

See **Enterprise Manager**.

Oracle HTTP Server

The **Web server** component of **Oracle Application Server**, built on Apache Web server technology and used to service **HTTP** requests. It is the part of the **middle tier** that handles requests between the Web and OracleAS Portal. Extensions to the Oracle HTTP Server support Java **servlet**s, **JSP**s, Perl, PL/SQL, and CGI applications.

Oracle Internet Directory

See **OID**.

Oracle Technology Network

See **OTN**.

Oracle Text

A feature of Oracle9*i* and later that provides advanced search and retrieval services on content stored in an Oracle repository. It is fully integrated into OracleAS Portal to provide users with the ability to perform a full text search and retrieval of content managed within the OracleAS Portal schema of the **Oracle Application Server**

Metadata Repository. It also provides automatic grouping and classification of results by **gist** and **theme**.

Oracle Ultra Search

An **Oracle Text**-based application that supports crawling, indexing, and federated searching of multiple, heterogeneous repositories including databases, file systems, **Web server**s, and e-mailing list archives.

Contrast with **search portlet**.

OTN

Oracle Technology Network. The online Oracle technical community that provides a variety of technical resources for building Oracle-based applications. You can access OTN at `http://www.oracle.com/technology/`.

Overwrite mode

See **Replace on Import mode**.

package

A database **object** consisting of a PL/SQL specification and a body. The specification includes the data types and subprograms that can be referenced by other program units. The body includes the actual implementation of the package.

page

An OracleAS Portal **object** that contains **portlet**s and **item**s. Each time you display a page, it is dynamically assembled and formatted according to the portlets and layout chosen for that page.

See also **page type**.

page designer

A page designer (also known as page manager) is a user with the Manage privilege on a page. A user with this privilege can perform any action on the page and can create sub-pages under the page. The page designer is often responsible for designing the layout (or region configuration) of the page and assigning privileges on the page to other users (for example, to determine who can add content to the page).

The scope of a page designer's control over a page may be limited if the page is based on a template.

page event

A user action defined by a portlet developer. User actions include clicking a link, button, or other control on a Web page. A page designer can specify that an event forces the reloading of the current **page** or the loading of another page, and passes **parameter**s to the newly loaded page.

page function item type

A **navigation item type** that a user can add to a **page**. A page function is a procedure call that a user can add to a custom **page type**. If there are no page functions associated with the current page, this item type does not display.

page group

An OracleAS Portal **object** that groups and sets properties of related portal objects, such as **page**s, **style**s, **navigation page**s, and **perspective**s. Page groups typically contain a hierarchy of pages and sub-pages for organizing content.

page group administrator

User who has full privileges over an entire **page group**. Page group administrators set up and maintain the page group; designate page owners; and create a taxonomy. Page group administrators can also view and manage all the **page**s in the page group.

page group map

Displays the hierarchical organization of all **page group**s in a portal and enables users to access individual **page**s within the page group. The page group map is tailored for each user; only the pages the user is authorized to view and/or edit are displayed.

page group quota

See **quota**.

page link item type

One of the default **item type**s that a **content contributor** can add to a **page**. A page link provides a route using a hyperlink to another page within the **portal**. When the user clicks the **display name** link, the page referenced by the item is displayed in the user's browser.

page manager

See **page designer**.

page metadata

Stored information or attributes about a **page**, which is used by OracleAS Portal to set its layout and cache.

page path item type

A **navigation item type** that a user can add to a **page**. A page path is a chain of page reference names. Page paths, often called breadcrumbs, describe the complete directory path.

page template

An OracleAS Portal **object** that establishes a common look and feel and common content for every **page** that uses the template. Users creating the page may select a page template to define the layout of the page and add default content. A page template includes all the features of a page, therefore may contain **item**s, **navigation page**s, and **portlet**s.

Contrast with **user interface (UI) template**.

page toolbar

In page **Edit mode**, the links at the top of the page that enable you to edit various aspects of the **page**, switch editing views (for example, from **graphical view** to **layout view**), edit **page group** properties, and so on.

page type

Defines the content of a **page** and the information that is stored about a page. Base page types included with OracleAS Portal are: **standard page**, **mobile page**, **PL/SQL page type**, **JSP**, and **URL page**. Custom page types are page types created by **page group administrator**s to extend the functionality provided by base page types and store additional information about pages.

Parallel Page Engine

See **PPE**.

parameter

A value passed between **page**s and **portlet**s, or between portlets.

A page parameter is a page level parameter (created by a page designer) whose values can be mapped to portlet parameters.

A portlet parameter is declared by a provider. Page designers map page parameters to portlet parameters. When the **PPE** requests a portlet from a provider, only the portlet parameters that the portlet declared and mapped to page parameters are sent.

parameter entry field

Field on a **customization form** that enables end users to enter values that will be passed to an OracleAS Portal **portlet**.

partial page caching

See **PPC**.

partner application

An application that has delegated its authentication to the **Oracle Application Server Single Sign-On Server**. If registered with the SSO Server, users can log in to multiple **partner application**s using a single log in page. In a given session, once users have been authenticated by the SSO Server, they won't need to log in again to access additional partner applications.

path aliasing

See **direct access URL**s.

PDK

OracleAS Portal Developer Kit. The development framework used to build and integrate Web content and applications with OracleAS Portal. It includes toolkits, samples, and technical articles that help make portal development simple. You can take existing Java **servlet**s, **JSP**s, **URL**-accessible content and Web Services and turn them into **portlet**s. It is typically used by external developers and vendors to create portlets and services. The PDK is regularly updated on **Portal Center** (`http://www.oracle.com/technology/products/ias/portal/index.html`) to provide developers with the latest tools and techniques.

See also **PDK-Java**, **PDK-PL/SQL**, and **PDK-URL Services**.

PDK-Java

A toolkit for implementing **portlet**s in Java and adding portal features. Used to declaratively turn your existing Java **servlet**s, **JSP**s, and Web services into portlets.

See also **PDK**. Contrast with **PDK-PL/SQL**.

PDK-PL/SQL

A set of articles, samples, and services that enable PL/SQL programmers to easily create portlets and extend them by using PL/SQL **API**s.

See also **PDK**. Contrast with **PDK-Java**.

PDK-URL Services

A utility for declaratively turning secured and public Web content into **portlet**s. These services are capable of dynamically passing parameters to target **URL**s, clipping and reformatting content, and providing SSO for applications requiring form-based or basic authentication. These services also allow developers to take any application written in any language and easily create integrated portlets. PDK-URL Services takes the URL of an application, parses the content, and uses the **PDK-Java** framework to create a portlet.

See also **PDK**.

Pending Approvals Monitor

A portlet that enables you to list pending approvals in the page groups that you administer. You can list the pending approvals by approver, date, page group, or submitter.

Pending Items Preview mode

A preview mode that enables you to view items that are awaiting approval. This mode can be used by content contributors to preview items they have added before they are approved, and by approvers to preview items before they approve or reject them.

See also **Edit mode**.

personal page

An area within OracleAS Portal where **authenticated user**s can store personal content and share it with other users. The **portal administrator** can choose to create a personal page for a user when creating a user account.

perspective

A cross-category grouping of **item**s. Perspectives help users answer the question, "Who will be interested in this item?" For example, you can add links to diverse vacation spots around the world and assign perspectives like *Vacations for Nordic Enthusiasts*, *Archeology Expeditions*, and *Extreme Vacations for Adventurers* to items about vacation types. Users publishing content using an item type that includes a perspective attribute may specify none, one, or many values.

Contrast with **category**.

PL/SQL item type

One of the default **item type**s that a **content contributor** can add to a **page**. A PL/SQL item contains a block of PL/SQL code. When a user clicks the item, the block is executed. The result displays in the user's browser. PL/SQL items can also be displayed directly on the page.

PL/SQL function

See **function**.

PL/SQL page type

One of the **page type**s supported by OracleAS Portal. PL/SQL pages contain PL/SQL code that generates **HTML** when the page is rendered.

poll

A set of questions used to find out information from users.

Contrast with **survey** and **test**.

portal

A common interface (that is, a Web page) that provides a personalized, single point of interaction with Web-based applications and information relevant to individual users or class of users. Portals built using OracleAS Portal are made up of **page**s managed within **page group**s, containing **portlet**s and **item**s.

portal administrator

User with the highest level of privileges in OracleAS Portal. Portal administrators can view and modify anything in OracleAS Portal, even **page**s and **database provider**s marked private. (The only exception is **group**s: although portal administrators can modify the PORTAL_ADMINISTRATORS and PORTAL_PUBLISHERS groups, they cannot modify any other group unless they have been named **group owner**.)

Portal Builder page

A predefined **page** that contains development and administrative **portlet**s used to build and manage OracleAS Portal **object**s and services.

portal cache

The portal cache stores cache entries for objects that use **validation-based caching**. It also acts as a backup to the memory-based **Oracle Application Server Web Cache** when objects use both validation-based caching and **invalidation-based caching**.

Portal Catalog

(No longer in use). Portal Catalog referred to the application that was formerly on Portal Center which contained information on partners who offered OracleAS Portal related products and services. This information was migrated to the Oracle PartnerNetwork Solutions Catalog (`http://solutions.oracle.com/`). Users should now look in the Solutions Catalog for this information.

Portal Center

(`http://www.oracle.com/technology/products/ias/portal/index.html`)
A Web site where you can find out everything you want to know about OracleAS Portal. Updated frequently, it always has the latest product information, and is home to the **Portal Developer Services** and **Portal Catalog**. This Web site contains all information about the product (including documentation, demonstrations, etc.), and provides access to OracleAS Portal expertise.

Portal Community

A network of people dedicated to creating and exchanging information about OracleAS Portal. This community includes anyone who uses OracleAS Portal; it can leverage the **Portal Knowledge Exchange** for sharing Portal-related information and take advantage of the **Portal Developer Services**.

Portal DB provider

See **database provider**.

Portal Developer Kit

See **PDK**.

Portal Developer Services

The network of people dedicated to creating and exchanging OracleAS Portal expertise. This program provides online testing tools through **Portal Center** (`http://www.oracle.com/technology/products/ias/portal/index.html`), as well as other venues, and interaction with the product team and other Portal developers, using newsletters, surveys, and the **Portal Knowledge Exchange**.

See also **Portal Community**.

Portal Knowledge Exchange

A self-service Web site where subscribers to the **Portal Developer Services** can share white papers, techniques, and portlets with others in the community.

Contributions can also be rated so that the most valuable contributions can be easily located.

portal page

See **page**.

portal repository

See **Oracle Application Server Metadata Repository**.

portal session

A period of interaction between a browser and OracleAS Portal, from the initial access to log off, closure of the browser window, or expiration of the session after a period of inactivity.

Portal smart link item type

A **navigation item type** that is self-configuring. For example, if you add a Home smart link to a **navigation page**, when a user clicks the Home link, he or she is automatically taken to his or her **home page**.

Portal smart text item type

A **navigation item type** that is self-configuring. For example, if you add a Current Date smart text item to a **navigation page**, the current date is automatically pulled from the server and need not be specified (through complex coding) by you.

Portal Studio

See **OracleAS Portal Verification Service**.

portlet

A reusable, pluggable Web component that typically displays portions of Web content. Portlets are the fundamental building blocks of an OracleAS Portal **page**. Using the wizard-based portlet builder, you can easily create your own **data-driven portlet**s. OracleAS Portal also provides several ways to build portlets programmatically and to integrate any kind of Web content. Portlets may be implemented using various technologies, such as Java, **JSP**s, Java **servlet**s, PL/SQL, Perl, ASP, etc. The **PDK** covers the standard-based portlet development options that OracleAS Portal provides.

portlet provider

See **provider**.

portlet publisher

User who can publish OracleAS Portal **objects** (**pages** or otherwise) as **portlets** so that they can be included on pages.

portlet record

A programmatic structure that contains detailed information about a **portlet**, such as its implementation style and **show mode** (PL/SQL).

Portlet Repository

A special **page group** that contains the **portlets** available from the local **providers** and any registered remote providers. When you register a provider, the provider and its portlets are added to the Portlet Repository.

PPC

Partial Page Caching. A feature that enables **Oracle Application Server Web Cache** to independently cache and manage fragments of HTML documents. A template page is configured with Edge Side Includes (**ESI**) markup tags that tell Oracle Application Server Web Cache to fetch and include the HTML fragments. The fragments themselves are HTML files containing discrete text or other objects.

PPE

Parallel Page Engine. A multi-threaded **servlet** engine that runs in the **OC4J** container and services **page** requests. The PPE reads **page metadata**, calls **providers** for **portlet** content, accepts provider responses, and assembles the requested page in the specified page layout. The Parallel Page Engine is part of the OracleAS Portal **middle tier**.

pretty URL

See **direct access URL**.

Preview mode

An optional **portlet show mode** that provides users with a preview of the portlet before they add it to a page.

primary key

Column in a database table consisting of unique values that can be used to identify rows in a table.

privilege

In OracleAS Portal, the right to perform an action. Privileges are either global (set through in the User or Group Profile) or specific to particular **objects** (usually set through the object's Access tab). When building applications, access can also be granted to database objects, shared portlets, portlets, and applications.

producer

See **provider**.

profile

Information about an OracleAS Portal user or group, such as password, user ID, and **privileges**.

property sheet

A built-in **attribute** that displays a summary of an **item**'s attributes or a **page**'s properties.

provider

The communication link between OracleAS Portal and a **portlet**. There are two types of providers: **Web providers** and **database providers**. Web providers may reside anywhere on the network and are addressed through **SOAP**. Web providers may be implemented using any Web technology. You can build your own Web providers by using the **PDK-Java** and the **PDK-URL Services**. Database providers reside within an Oracle database and manage portlets while performing data-intensive operations.

Providers act as containers for portlets; each portlet communicates with OracleAS Portal through its provider. Providers also manage the portlets they contain.

provider definition

A declarative, **XML**-based configuration file (`provider.xml`) that describes a **Web provider**, its **portlets**, and the location of the content to be displayed in the portlets. This configuration file also describes the behavior of the provider and its portlets.

provider group

A logical collection of **Web providers** defined by a provider group service. A **portal administrator** can register provider groups for use with their portal. Once registered, a provider group simplifies the process of registering individual **providers** in the group. This enables organizations that create Web providers to publish registration details of their providers and facilitate automatic registration with any OracleAS Portal instance. The only information that must be given to the portal administrator is the name and location of the provider group.

See also **local provider group** and **remote provider group**.

provider record

Record returned by a **database provider** containing specified information about a **portlet**.

proxy server

A proxy server typically sits on a network **firewall** and enables clients behind the firewall to access Web resources. All requests from clients go to the proxy server rather than directly to the destination server. The proxy server forwards the request to the destination server and passes the received information back to the client. The proxy server channels all Web traffic at a site through a single, secure port; this enables an organization to create a secure firewall by preventing Internet access to internal machines, while allowing Web access.

public page

Any **page** in a **page group** that is viewable by **public user**s (users who are not logged onto OracleAS Portal). The page manager or **page group administrator** must explicitly designate a page as public.

public user

User who can access, but is not logged onto, OracleAS Portal. When users first access OracleAS Portal, they do so as public users, whether or not they have the ability to log on. A public user can view any **page** that has been marked as public, but cannot customize or edit any content, or view pages that have any form of access control.

Contrast with **authenticated user**.

purge

See **system purge**.

query

A SQL SELECT statement that specifies which data to retrieve from one or more tables or views in a database.

quota

The amount of space provided in a page group or in the OracleAS Portal schema of the **Oracle Application Server Metadata Repository** to store uploaded documents.

recent object

An OracleAS Portal **object**, such as a **page** or **portlet**, that has recently been displayed or edited. Each **authenticated user** has his or her own Recent Objects portlet that provides links to the last *n* objects accessed.

referenced object

Export object category. An object which is directly or indirectly referenced by an **explicit object**.

See also **manifest**.

reference path

Path that uniquely identifies a **portlet** instance on a **page**. A reference path can be used to target **parameter**s to individual portlets on a given page.

region

A carved-out area on a **standard page** used to define the page layout, define page content (**portlet**s and **item**s), and control the **style** and **attribute**s for content displayed in a region. A standard page can have one or multiple regions. Regions can be created above, below, or beside other regions.

You can create the following types of regions:

- Undefined regions are regions that have not been assigned a particular type.
- Item type regions allow you to add items such as text, images, files, and so forth.
- Portlet type regions allow you to include portlets in a region.
- Sub-Page Links regions allow you to display a list of the current page's sub-pages in a region.
- Tab type regions allow you to include **tab**s in a region.

region banner

A colored, horizontal bar with a title displayed in a **region** of an OracleAS Portal **page**. A banner breaks up the visual flow of a page and groups related **item**s that appear beneath it.

remote database

Database running on a separate machine that can be accessed over the network through a connect string or database link.

remote provider group

The collection of **provider**s that are defined outside of your local instance of OracleAS Portal.

See **provider group**. Contrast with **local provider group**.

Replace on Import mode

Import mode. When this option is selected, if the object exists on the target, then it is replaced. If the object does not exist then it is created. When this option is not selected, if the object exists on the target, it is referenced. If it does not exist on the target, it is created.

report

An OracleAS Portal **portlet** that displays the results of a SQL **query** in a tabular format.

Reuse mode

See **Replace on Import mode**.

rich text editor

A WYSIWIG editor that enables **content contributor**s to easily apply formatting to text items. The rich text editor is only available in Internet Explorer.

root page

The top level of the **page** hierarchy in a **page group**; it contains all other sub-pages in the page group. Also known as the page group's **home page**.

routing method

OracleAS Portal provides three approval routing methods:

- All, Parallel enables OracleAS Portal to send the approval to recipients in the step all at the same time. All of the recipients must respond to the approval before the item approval can move to the next step.
- All, Serial enables OracleAS Portal to send the approval to recipients in the step one at a time in the sequence specified. All of the recipients must respond to the approval before the item approval can move to the next step.

- Any, Parallel enables OracleAS Portal to send the approval to recipients in the step all at the same time. However, only one of the recipients must respond to the approval before the item approval can move to the next step.

See also **approval process**.

row

Set of values in a table; for example, the values representing one employee in the SCOTT.EMP table.

saved search

A saved search enables you to save all the search criteria under a single name. This feature enables you to repeat the search quickly, by choosing the saved search name rather than re-entering the criteria manually. You can save the results of a **basic search**, **advanced search**, or **custom search**. The Saved Searches portlet lists all the saved searches in a page group.

search portlet

Enables users to search for **page**s and content within the OracleAS Portal schema of the **Oracle Application Server Metadata Repository**. Users can also search based on text strings, categories, perspectives, and attributes, and using operators such as CONTAINS, GREATER THAN, LESS THAN, and EQUAL TO.

Contrast with **Oracle Ultra Search**.

self registration

Allows users to create new accounts for themselves through a link in the Login portlet.

sequence

A database **object** used to automatically generate numbers for table rows.

servlet

A Java program that usually runs on a **Web server**, extending the Web server's functionality. **HTTP** servlets take client HTTP requests, generate dynamic content (such as through querying a database), and provide an HTTP response.

session

See **portal session**.

shared object

An OracleAS Portal **object** such as a **personal page**, **navigation page**, **style**, **page template**, **perspective**, **category**, or custom type that can be shared across **page groups**.

shared portlet

An OracleAS Portal **portlet** that is shared between other portlets in an OracleAS Portal **database provider**. Each shared portlet can be displayed on multiple **pages** with the same personalization.

Shared Screen mode

A **portlet show mode** that renders the body of the portlet. Every portlet must have at least a Shared Screen mode.

show mode

The ways by which a **portlet** can be called to display information. These methods include:

- **Shared Screen mode**
- **Edit mode**
- **Edit Defaults mode**
- **Preview mode**
- **Help mode**
- **About mode**
- **Link mode**
- **Full Screen mode**

simple file item type

A simplified version of the **file item type**, this item type includes fewer **attributes** and therefore is quicker and easier to create.

simple image item type

A simplified version of the **image item type**, this item type includes fewer **attributes** and therefore is quicker and easier to create.

Simple Object Access Protocol

See **SOAP**.

simple page link item type

A simplified version of the **page link item type**, this item type includes fewer **attribute**s and therefore is quicker and easier to create.

simple PL/SQL item type

A simplified version of the **PL/SQL item type**, this item type includes fewer **attribute**s and therefore is quicker and easier to create.

simple text item type

A simplified version of the **text item type**, this item type includes fewer **attribute**s and therefore is quicker and easier to create.

simple URL item type

A simplified version of the **URL item type**, this item type includes fewer **attribute**s and therefore is quicker and easier to create.

Single Sign-On

See **Oracle Application Server Single Sign-On Server**.

smart link item type

See **Portal smart link item type**.

smart text item type

See **Portal smart text item type**.

snapshot

A **table** that contains the results of a **query** on one or more tables, called master tables, in a remote database.

snapshot log

A **table** associated with the master table of a **snapshot** tracking changes to the master table.

SOAP

Simple Object Access Protocol. A lightweight, **XML**-based protocol for exchanging information in a decentralized, distributed environment. SOAP supports different styles of information exchange, including: Remote Procedure Call style (RPC) and Message-oriented exchange. RPC style information exchange allows for request-response processing, where an endpoint receives a procedure oriented message and replies with a correlated response message. Message-oriented information exchange supports organizations and applications that need to exchange business or other types of documents where a message is sent but the sender may not expect or wait for an immediate response.

SSL accelerator card

A hardware device that handles traffic much faster than regular SSL software.

See also **LBR**.

SSO

Single Sign-On.

See **Oracle Application Server Single Sign-On Server**.

standard page

A **page type** used to contain and manage **item**s and **portlet**s.

stored procedure

A set of PL/SQL procedures that are stored in a database.

stripe

Secure slice on the **virtual private portal** that is assigned to a particular **subscriber**.

structured UI template

A **shared portlet** that controls the look and feel of OracleAS Portal **portlet**s and runs in standalone mode. Structured UI templates display the same image and text in the same location around every portlet that uses the template.

See **user interface (UI) template**. Contrast with **unstructured UI template**.

style

A set of values and parameters that controls the colors and fonts of **page**s and **region**s within a page. Style settings include font style, size, color, alignment, and background color. Styles can be created for a specific **page group** or as a **shared object** that is used by pages within multiple page groups.

sub-category

A **category** that appears hierarchically below another (parent) category. This provides a way of grouping closely related categories together.

sub-item

An **item** that appears hierarchically below another (parent) item. This provides a way of grouping closely related items together, for example, the spreadsheet that is used by a particular HTML file item could be added as a sub-item of the HTML file item.

sub-page

A **page** that appears hierarchically below another (parent) page. Every page in a page group (except the root page) is a sub-page.

sub-perspective

A **perspective** that appears hierarchically below another (parent) perspective. This provides a way of grouping closely related perspectives together.

subscriber

Company that signs up with an **ASP** and receives a **stripe** on a hosted OracleAS Portal.

subscription notification

A method by which end users can subscribe to a particular **page** or **item** so that they are notified (through the My Notifications portlet) when that page or item is updated. The page designer must include a Subscribe **Portal smart link item type** for users to be able to subscribe to a page, and must display the Subscribe **attribute** for users to be able to subscribe to items. Additionally, the **page group administrator** must enable approvals and notifications for the **page group**.

substitution tag

Special portal tag used within **unstructured UI template**s to dynamically embed titles, headings, and other elements into the template.

survey

A set of questions used to find out information from users. Surveys can redirect users to different sections of the survey depending on their answers to particular questions.

Contrast with **poll** and **test**.

synonym

An additional name assigned to a **table** or **view** that can thereafter be used to refer to it.

system level caching

System level caching places a single copy of an object in the system cache (on the middle tier) for all users. Consequently, all customization options introduced by individual users on their own instance of the object are disabled. Also, access privileges for the object will not be enforced. Examples of content that may be suitable for system level caching include page banners and news portlets.

system purge

Deletes all items in a **page group** from the OracleAS Portal schema of the **Oracle Application Server Metadata Repository** that are marked as deleted or expired. System purges are performed by the **page group administrator** or **portal administrator**.

tab

An area on a **page** used to increase the amount of content that the page can display by effectively doubling (or tripling, quadrupling, and so on) the amount of real estate available. Tabs also allow you to group content that are common to a subject area, organization, specific role, and so forth.

table

The basic storage structure in a relational database.

tablespace

Allocation of space in the database.

template

See **page template** or **user interface (UI) template**.

temporary tablespace

Allocation of space in the database used for the creation of temporary table segments for operations such as sorting table rows.

test

A set of questions used to assess a user's understanding of a particular subject or subjects. You can provide the correct answer for questions and assign each question a score. You can also hand score essay-type answers.

Contrast with **poll** and **survey**.

text item type

One of the default **item types** that a **content contributor** can add to a **page**. When you create a text item, you enter text (up to 32KB) in the Item Wizard. The text block is then stored in the OracleAS Portal schema of the **Oracle Application Server Metadata Repository**.

theme

A snapshot generated by **Oracle Text** that describes a document. Rather than searching for documents that contain specific words or phrases, users can use Oracle Text to search for documents that are about a certain subject, even if that subject is not mentioned explicitly in the document.

title

See **display name**.

translation

A **page group** rendered in another language. When a **page group administrator** creates a translation, **content contributors** can add content in that language. Page group users can also view the translated content by setting their browser language to one of the supported languages.

transport set

A transport set is a collection of OracleAS Portal **objects** for **export** or **import**. It can contain more than one object of a particular type, such as multiple **page groups** and multiple **pages**.

trigger

A database **object** associated with a table. It executes before or after one or more specified events.

Ultra Search

See **Oracle Ultra Search**.

Uniform Resource Locator

See **URL**.

unstructured UI template

A **shared portlet** that is used to insert content and **HTML** code into a specified **page** or control the look and feel of OracleAS Portal **portlets**. Unstructured UI templates are based on HTML code that, when executed, dynamically embeds titles, headings, and other elements that make up a page.

See also **substitution tag** and **user interface (UI) template**. Contrast with **structured UI template**.

URL

Uniform Resource Locator. A compact string representation of the location for a resource that is available through the Internet. It is also the format Web clients use to encode requests to **Oracle Application Server**.

URL item type

One of the default **item type**s that a **content contributor** can add to a **page**. A URL item, when clicked, provides a route to another Web page. When a user clicks the URL item's **display name**, the Web page referenced by the **URL** displays.

URL page

A **page type** that provides a route to another Web page, identified by its **URL**. When a user clicks the page link, the Web page referenced by the link is displayed.

URL portlet

An OracleAS Portal **portlet** that displays the contents of a Web page specified by a **URL**.

user interface (UI) template

A **shared portlet** that controls the look and feel of OracleAS Portal **portlet**s in full page display mode. Selecting a UI template when you are building a portlet automatically selects a title on the page where the portlet is displayed, a title background, links to other Web pages, and background colors and images.

See also **structured UI template** and **unstructured UI template**. Contrast with **page template**.

validation-based caching

A **caching** method that is performed using the **portal cache**. Before an item in the portal cache is used, the **PPE**, or **mod_plsql**, contacts the portal or a **provider** to determine if the cached item is still valid.

Contrast with **expiry-based caching** and **invalidation-based caching**.

versioning

Allows multiple versions of an **item** to simultaneously exist in the OracleAS Portal schema of the **Oracle Application Server Metadata Repository**. This feature is useful for tracking document changes from one version to the next or for reverting to a previous version if necessary.

view

A virtual **table** whose rows do not actually exist in the database, but which is based on a table that is physically stored in the database.

View mode

Runtime view of a **page**.

Contrast with **Edit mode**.

virtual private database

See **VPD**.

virtual private portal

Refers to the OracleAS Portal features for hosting multiple companies or multiple organizations securely within the same portal instance.

VPD

Virtual Private Database. A feature for **ASP**s that want to leverage the Oracle database to host their customers. Essentially, it uses one physical database instance for all customers, but to each customer it looks like they have their own database. Users cannot see any information that is not meant for them and complete customer isolation is achieved. It requires little to no changes in the core application to take effect as most of the work is done at the database level. Implementing VPD basically requires two key steps: adding a context column (for example., company name) to all the database tables, and implementing a policy to restrict queries on each table based on the context of the logged in user. VPD provides highly secure, full subscriber isolation using this method.

WAP

Wireless Application Protocol. A set of open, global protocols for developing applications and services that use wireless networks.

Web Cache

See **Oracle Application Server Web Cache**.

Web clipping

Enables page designers to collect Web content into a single centralized portal. It can be used to consolidate content from hundreds of different Web sites scattered throughout a large organization.

WebDAV

Web-based Distributed Authoring and Versioning. A protocol extension to **HTTP** 1.1 that supports distributed authoring and versioning. With WebDAV, the Internet becomes a transparent read and write medium, where content can be checked out, edited, and checked into a **URL** address.

Web Services for Remote Portlets

See **WSRP**.

Web provider

An entity that is called, using an **HTTP** request, by OracleAS Portal and returns **portlet** content in **HTML**, **XML**, or **WSRP**. A Web provider acts as a proxy for one or more portlets that are defined within a particular application environment (for example, Java, ASP, or Perl) and executed as applications external to OracleAS Portal. Web providers are particularly appropriate for Web-accessible information sources.

See also **provider**. Contrast with **database provider**.

Web server

A program that delivers Web pages.

Wireless Application Protocol

See **WAP**.

Wireless Markup Language

See **XML**.

wireless portal

A **portal** accessible from wireless devices, such as cellular telephones.

See also **Oracle Application Server Wireless**.

wizard

Graphical interface that guides a user step-by-step through a process. In OracleAS Portal, wizards are used for creating **database provider**s, **portlet**s, database **object**s, **page**s, **page group**s, and **item**s.

WML

Wireless Markup Language. An **XML**-based markup language used to define hypertext-like content and applications for handheld devices.

WSRP

Web Services for Remote Portlets (WSRP). Web services standard that allows the plug-and-play of visual, user-facing Web services with portals or other intermediary Web applications. Being a standard, WSRP enables interoperability between a standards-enabled container based on a particular language (such as JSR 168, .NET, Perl) and any WSRP portal. So, a portlet (regardless of language) deployed to a WSRP-enabled container can be rendered on any portal that supports this standard. WSRP is not supported in OracleAS Portal 10.1.2.

XML

Extensible Markup Language. An open standard for describing data using a subset of the SGML syntax.

XML portlet

An OracleAS Portal **portlet** that displays the executed results of **XML** code. To create the portlet, you either specify XML code or a **URL** that points to the XML code.

zip file item type

One of the default **item type**s that a **content contributor** can add to a **page**. Zip file items enable you to upload many files in a single operation. You can use them to migrate the contents of a file system or Web site into the OracleAS Portal schema of the **Oracle Application Server Metadata Repository**. When you upload a zip file to OracleAS Portal, then unzip the uploaded file, a page is created for each directory and an **item** is created for each file. The items are published in the target page.

Index

A

About mode, 5-7
access control lists
    see ACLs
Access tab for managing portlets, A-83
ACLs, 5-65
    portlet privileges, 5-66
    privileges, 5-66
    provider privileges, 5-66
ALIGN_LEFT substitution tag, A-112
ALIGN_RIGHT substitution tag, A-113
Apache struts
    creating a portlet, 5-105
    overview, 5-103
APIs, 1-11
audio data, A-54
authenticated content
    Single Sign-On and, 1-8, 4-2, 4-9
authentication, 5-62
    comparison of portlet builders, 2-20
    external application, 5-64
    message, 5-63, 5-68
    none, 5-65
    partner application, 5-64
    proxy
        Web Clipping and, B-10
    server, 5-63
    single sign-on, 5-63
authorization, 5-63

B

Batch button, adding, A-93
batch mode, for running portlets, A-92
best practices
    About mode, 5-7
    CSS for Shared Screen mode, 5-3
    Edit Defaults mode, 5-5
    Edit mode, 5-4
    error handling PL/SQL, 6-38
    event handling PL/SQL, 6-41
    for Java portlets, 5-2
    Full Screen mode, 5-7
    Help mode, 5-7
    HTML for Shared Screen mode, 5-2
    Link mode, 5-8
    PL/SQL procedures and functions, 6-2
    Preview mode, 5-6
    security PL/SQL, 6-27
bind variables
    LOV, A-80
    using, A-96
Blank item type, form portlets, A-22
BLOBs, A-22, A-56
BODY substitution tag, A-112
browsers
    recommended cache settings, xviii
    recommended image settings, xix
    recommended versions, xviii
browsing
    Web Clipping and, 4-6
Button item type, form portlets, A-22

C

caching, 5-88
    activation, 5-90
    adding, 5-90, 5-91, 5-95
    expiry-based, 2-12, 5-89
    expiry-based PL/SQL, 6-34
    invalidation port, 5-91
    invalidation-based, 2-12, 5-89
    invalidation-based PL/SQL, 6-35
    manually invalidating the cache, 5-94
    PL/SQL, 6-30
    provider servlet for invalidation based, 5-91
    provider.xml for invalidation based, 5-93
    recommended browser settings, xviii
    style, 2-11
    system-level, 2-11
    user-level, 2-11
    validation-based, 2-12, 5-89
    validation-based PL/SQL, 6-32
calendar portlets, A-12
chart portlets, A-12
    accessible tables and views, A-59
    column conditions, A-62
    common display options, A-65
    comparison, 2-17
    customization form button options, A-72
    customization form display options, A-69

customization form formatting options, A-71
examples, A-120
formatting conditions, A-63
from query wizard, A-57
full page display options, A-66
group functions, A-61
labels, A-56, A-60
links, A-56, A-60
mobile display options, A-68
PL/SQL code options, A-75
portlet display options, A-67
supplementary text options, A-74
values, A-56, A-60
check box, LOV, A-79
CheckBox item type, form portlets, A-22
cipher manager, 5-84
clipping sections, 4-6
code
adding functionality, A-95
bind variables, A-96
viewing body, A-85
viewing call interface, A-85
viewing package spec, A-85
viewing portlet source, A-84
color, creating definitions for, A-107
column conditions, A-36
combo box, LOV, A-79
ComboBox item type, form portlets, A-22
communication
HTTPS, 5-69
security, 5-63
connection
to application server in Oracle JDeveloper, 5-21, 5-38
content
adding to page, 4-2
rendering inline, 2-16
context information
accessing PL/SQL, 6-24
function calls PL/SQL, 6-25
obtaining via wwctx_api package, 6-25
cookies, A-99
Web Clipping and, 4-21
Create shared components privilege, A-105
CSS
guidelines for Shared Screen mode, 5-3
customization
implementing PL/SQL session store, 6-19
preference store for PL/SQL, 6-15
customizations
transporting, 5-74
Customize privilege, A-87
customizing
code generated by wizard, 5-50
Edit and Edit Defaults pages, 5-50
modifying generated code, 5-51
portlets, 5-48
Show page, 5-52

D

data
filtering, 3-9
data component portlets, A-13
data sources
filtering data, 3-9
using a spreadsheet, 3-5
using a Web Service, 3-7
using an existing Web page, 3-8
using SQL, 3-5
using with portlets, 3-3
using XML, 3-6
database providers, 2-8, 2-9
DBA group, A-105
debugging information
Web Clipping and, B-11
default profile, A-3
deploying
WAR file for JPS portlet, 5-22
WAR file for PDK-Java portlet, 5-39
develop in-place portlets, 2-13
Develop tab for managing portlets, A-83
develop-in-place portlets, 2-3, A-11
DIRECTION substitution tag, A-112
directories, virtual, A-109
doDelete, A-99
doInsert, A-99
doUpdate, A-99
dynamic page portlets, A-12

E

Edit Defaults mode, 3-19, 5-5
Edit Defaults page, 5-50
Edit mode, 5-4
Edit page, 5-50
Edit privilege, A-87
encryption
example, 5-84
for transport, 5-83
message, 5-63
error handling
PL/SQL, 6-36
errors
Web Clipping and, B-9, B-11
event handling
PL/SQL, 6-40
events, 5-53
adding to a portlet, 3-20
generated code in provider.xml, 5-57
submitting, 5-56
using with OmniPortlet, 3-20, 3-37
examples
average salaries chart, A-120
team bonuses report, A-121
team details report, A-118
Execute privilege, A-87
expiration
portlet content, 4-9
Expires field

Web Clipping and, 4-9
expiry-based caching, 2-12
export
by reference, 5-83
by reference example, 5-86
customizations, 5-74
encrypting for, 5-83
encryption example, 5-84
example of, 5-77
JNDI variable, 5-83
logging interface, 5-76
programming interface, 5-75
securing communications, 5-82
security, 5-82
exportData, 5-75, 5-76
external applications
authentication, 5-64
Web Clipping and, 1-8, 4-2, 4-9

F

file upload (binary) item types, form portlets, A-22
file upload (interMedia) item types
form portlets, A-22
including in form, A-55
font definitions, creating, A-109
form portlets, A-12, A-15
adding JavaScript, A-107
audio data, A-54
doDelete, A-99
doInsert, A-99
doUpdate, A-99
formatting and validation options, A-19
getting and setting values, A-98
image data, A-54
interMedia rich content, A-54
JavaScript Event Handlers, A-22, A-26
p_session, A-98
p_value, A-98
PL/SQL code options, A-29
PL/SQL Event Handlers, A-23
set_value, A-98
supplementary text options, A-27
video data, A-54
format masks
dates, A-39
numbers, A-40
formatting and validation options
buttons, A-20
columns, A-23
form level, A-19
formatting conditions, A-41
forms
building with the Portlet Builder, A-15
Web Clipping and, 4-9, 4-19
frame driver portlets, A-13
Full Screen mode, 5-6
fuzzy matching
with Web Clipping, 1-8, 4-1

G

generator
PL/SQL, 6-3
grants, A-4
groups
DBA, A-105
PORTAL_DEVELOPERS, A-105
guidelines
About mode, 5-7
CSS for Shared Screen mode, 5-3
Edit Defaults mode, 5-5
Edit mode, 5-4
error handling PL/SQL, 6-38
event handling PL/SQL, 6-41
Full Screen mode, 5-7
Help mode, 5-7
HTML for Shared Screen mode, 5-2
Link mode, 5-8
Preview mode, 5-6
security PL/SQL, 6-27

H

Help mode, 5-7
HELPLINK substitution tag, A-112
HELPSCRIPT substitution tag, A-112
hidden item types, form portlets, A-22
hierarchy portlets, A-13
horizontal rule item types, form portlets, A-22
HTML
guidelines for Shared Screen mode, 5-2
HTTP error code 407
Web Clipping and, B-9
httpd.conf, 5-68
HTTPS, 5-69
configuration, 5-70

I

IFrames
Web Clipping and, B-10
image data, A-54
Image item types, form portlets, A-22
IMAGE_PREFIX substitution tag, A-112
images
creating definitions for, A-108
in Web Clipping clip, B-10
recommended browser settings, xix
virtual directory, A-109
import
by reference, 5-83
by reference example, 5-86
customizations, 5-74
encrypting for, 5-83
encryption example., 5-84
JNDI variable, 5-83
logging interface, 5-76
programming interface, 5-75
securing communications, 5-82
security, 5-82

importData, 5-75, 5-76
init.ora parameters, A-93
inline rendering, 2-16
  Web Clipping and, 1-8, 4-2
input parameters
  Web Clipping and, 4-9, 4-19
interMedia rich content, A-22, A-54, A-55
invalidation-based caching, 2-12
item types
  file upload (interMedia), A-55
item types, form portlets
  blanks, A-22
  buttons, A-22
  check boxes, A-22
  combo boxes, A-22
  file upload (binary), A-22
  file upload (interMedia), A-22
  hidden, A-22
  horizontal rule, A-22
  image, A-22
  label only, A-22
  password, A-22
  text area, A-22
  text box, A-22

J

J2EE application server, 2-6
Java
  portlet guidelines, 5-2
Java Community Process, 2-4
Java Portlet Specification
  see JPS
Java portlets
  authentication for external application, 2-21
  caching style, 2-12
  capturing content, 2-16
  charting, 2-17
  comparing to other portlet builders, 2-2
  development tool, 2-12
  event support, 2-19
  expertise required, 2-6
  multi-lingual support, 2-20
  pagination support, 2-20
  parameter support, 2-19
  rendering content inline, 2-17
  security, 2-19
  usage suitability, 2-4
  user interface, 2-15
JavaScript
  adding to a form portlet, A-107
  creating under shared components, A-107
  validation guidelines, A-106
  Web Clipping and, 4-21
JavaScript Event Handlers, A-96
  examples, A-97
  for buttons, A-22
  for columns, A-26
  writing, A-96
JNDI variable
  for export, 5-83
JNDIvariable
  for import, 5-83
JPS, 5-8
  creating portlet, 5-11
  deploying portlet, 5-20
  wizard in Oracle JDeveloper, 5-12
JSR-168, 2-2

L

Label Only item types, form portlets, A-22
language
  string loading PL/SQL, 6-44
  string retrieval PL/SQL, 6-45
language support
  see multi-lingual portlets
layout
  bullet, 3-17
  chart, 3-13
  form, 3-18
  modifying with OmniPortlet, 3-35
  news, 3-16
  OmniPortlet, 3-11
  tabular, 3-12
LDAP
  see Oracle Internet Directory
limitations
  Web Clipping and, 4-21
Link mode, 5-7
link portlets, A-13
list of values
  portlets, A-13
lists of values
  bind variables, A-80
  check box, A-79
  combo box, A-79
  creating, A-77
  dynamic, A-77
  multiple select, A-79
  pop-up list, A-79
  radio group, A-79
  static, A-77
load balancers
  Web Clipping and, B-9
locks, on portlets, A-85
log files
  Web Clipping and, B-11
logging levels
  Web Clipping and, B-11
logon denied message
  Web Clipping and, B-10

M

Manage privilege, A-87
Manage shared components privilege, A-105
Manage tab for managing portlets, A-83
matrix, portlet technologies, 2-1
menu portlets, A-13

messages
  authentication, 5-68
  encryption, 5-63
migration
  preference store, 5-49
MLS
  see multi-lingual portlets
mobile device display options, A-47
mod_osso
  Web Clipping and, 4-21
mode
  Edit Defaults, 3-19
modes
  About, 5-7
  adding render, 5-45
  Edit, 5-4
  Edit Defaults, 5-5
  Full Screen, 5-6
  Help, 5-7
  Link, 5-7
  list of Show, 5-2
  Preview, 5-6
  Shared Screen, 5-2
multi-lingual
  PL/SQL portlets, 6-43
multi-lingual portlets
  comparison, 2-20
  provider.xml, 5-98
  resource bundles, 5-96
  updating renderer, 5-97
multiple select, LOV, A-79

N

navigation
  with Web Clipping, 1-8, 4-1
NLS
  see multi-lingual portlets

O

OmniPortlet, 1-9
  about, 3-2
  adding to a portal page, 3-22
  authentication for external application, 2-21
  bullet layout, 3-17
  caching style, 2-12
  capturing content, 2-16
  chart layout, 3-13
  charting, 2-17
  comparing to other portlet builders, 2-2
  development tool, 2-12
  event support, 2-19
  Events tab, 3-20
  expertise required, 2-6
  Filter tab, 3-9
  filtering data, 3-9
  form layout, 3-18
  Layout tab, 3-11
  modifying the layout, 3-35
  multi-lingual support, 2-20
  news layout, 3-16
  pagination support, 2-20
  parameter support, 2-19
  rendering content inline, 2-17
  security, 2-19
  Source tab, 3-3
  tabular layout, 3-12
  Type tab, 3-3
  usage suitability, 2-4
  user interface, 2-15
  using data sources, 3-3
  View tab, 3-10
Oracle Internet Directory, 5-70
Oracle JDeveloper
  application server connection, 5-21, 5-38
  deploying JPS portlet, 5-20
  deploying PDK-Java portlet, 5-38
  portal add-in, 5-11
  WAR file deployment, 5-22, 5-39
Oracle struts portlet, 5-103
OracleAS Portal
  add-in for Oracle JDeveloper, 5-11
  repository, 2-10
  using data sources, 3-3
OracleAS Portal pages
  Web Clipping and, 4-21
OracleAS Web Cache
  invalidation port, 5-91, 5-93
  invalidation-based caching, 2-12
ORDAUDIO, A-54
ORDIMAGE, A-54
ORDVIDEO, A-54
owa_cookie package
  data types, A-99
  subprograms, A-99

P

p_session, A-98
p_value, A-98
package
  implementing portlet with PL/SQL, 6-9
  implementing provider with PL/SQL, 6-10
page designer, definition, xv
page manager
  see page designer
page metadata, 2-10
page parameters, 2-18
  using with OmniPortlet, 3-37
PAGE.BASE substitution tag, A-113
PAGE.BASE.URL substitution tag, A-113
PAGE.BGCOLOR substitution tag, A-113
PAGE.BGIMAGE substitution tag, A-113
PAGE.CUSTOMIZEPAGE substitution tag, A-113
PAGE.CUSTOMIZEPAGE.LABEL substitution tag, A-114
PAGE.CUSTOMIZEPAGE.URL substitution tag, A-114
PAGE.EDITPAGE substitution tag, A-114

PAGE.EDITPAGE.LABEL substitution tag, A-114
PAGE.EDITPAGE.URL substitution tag, A-114
PAGE.REFRESH substitution tag, A-114
PAGE.REFRESH.LABEL substitution tag, A-114
PAGE.REFRESH.URL substitution tag, A-114
PAGE.STYLE substitution tag, A-113
PAGE.STYLE.URL substitution tag, A-113
PAGE.SUBPAGELINK substitution tag, A-113
pagination support in portlets, comparison, 2-20
Parallel Page Engine (PPE), 2-10
parameters, 5-53
  adding, 5-54
  generated code in provider.xml, 5-55
  mapping public PL/SQL, 6-22
  passing page PL/SQL, 6-22
  passing private PL/SQL, 6-22
  PL/SQL, 6-21
  qualified PL/SQL, 6-22
  retrieving values PL/SQL, 6-23
  types, 2-18
  unqualified PL/SQL, 6-22
  using with OmniPortlet, 3-20, 3-37
  Web Clipping and, 4-9, 4-19
parameters and events
  using with OmniPortlet, 3-20, 3-37
partner application, 5-64
Password item types, form portlets, A-22
PDK, 1-11
PDK-Java
  comparing to other portlet builders, 2-2
  customizing portlets, 5-48
  deploying portlet, 5-38
  Portlet Wizard, 5-29
  render modes, 5-45
  testing portlet and provider, 5-36
PDK-PL/SQL
  comparing to other portlet builders, 2-2
performance
  caching, 5-88
  caching PL/SQL, 6-30
persistent state variables, A-99
personal pages
  Web Clipping and, 4-15
PL/SQL
  accessing context information, 6-24
  building portlets, 6-8
  caching, 6-30
  error handling, 6-36
  event handling, 6-40
  gateway, A-99
  generating for portlets, A-84
  generator, 6-3
  getting and setting values, A-98
  implementing the portlet package, 6-9
  implementing the provider package, 6-10
  owa_cookie package, A-99
  parameters, 6-21
  preference store, 6-15
  publishing generated portlets, 6-7
  recommended procedures and functions, 6-2
  running the generator, 6-6
  security, 6-27
  session store, 6-19
  starter provider sample, 6-8
  XML for generator, 6-4
PL/SQL code options
  chart portlets, A-75
  form portlets, A-29
  report portlets, A-52
PL/SQL Event Handlers, A-96
  for buttons, A-23, A-97
  writing, A-97
PL/SQL portlets
  authentication for external application, 2-21
  building expertise required, 2-6
  caching style, 2-12
  capturing content, 2-16
  charting, 2-18
  comparing to other portlet builders, 2-2
  development tool, 2-13
  event support, 2-19
  multi-lingual support, 2-20
  pagination support, 2-20
  parameter support, 2-19
  rendering content inline, 2-17
  security, 2-20
  usage suitability, 2-5
  user interface, 2-15
plsql.conf file, A-109
pop-up list, LOV, A-79
portal developer, definition, xv
Portal schema
  password and Web Clipping, B-10
Portal Tools
  about OmniPortlet, 3-2
PORTAL_DEVELOPERS group, A-105
PORTAL.ACCOUNTINFO substitution tag, A-113
PORTAL.ACCOUNTINFO.LABEL substitution tag, A-113
PORTAL.ACCOUNTINFO.URL substitution tag, A-113
PORTAL.COMMUNITY substitution tag, A-113
PORTAL.COMMUNITY.IMAGE substitution tag, A-113
PORTAL.COMMUNITY.LABEL substitution tag, A-113
PORTAL.COMMUNITY.URL substitution tag, A-113
PORTAL.HELP substitution tag, A-113
PORTAL.HELP.IMAGE substitution tag, A-113
PORTAL.HELP.LABEL substitution tag, A-113
PORTAL.HELP.URL substitution tag, A-113
PORTAL.HOME substitution tag, A-113
PORTAL.HOME.IMAGE substitution tag, A-113
PORTAL.HOME.LABEL substitution tag, A-113
PORTAL.HOME.URL substitution tag, A-113
PORTAL.LOGOUT substitution tag, A-113
PORTAL.LOGOUT.LABEL substitution tag, A-113
PORTAL.LOGOUT.URL substitution tag, A-113
PORTAL.NAVIGATOR substitution tag, A-113

PORTAL.NAVIGATOR.IMAGE substitution
    tag, A-113
PORTAL.NAVIGATOR.LABEL substitution
    tag, A-113
PORTAL.NAVIGATOR.URL substitution tag, A-113
portals
  about, 1-1
  using data sources, 3-3
portalstandards.oracle.com, 5-28
Portlet Builder
  building portlets with, A-1
  caching style, 2-12
  charting, 2-18
  comparing to other portlet builders, 2-2
  development tool, 2-12
  event support, 2-19
  expertise required, 2-6
  multi-lingual support, 2-20
  pagination support, 2-20
  parameter support, 2-19
  rendering content inline, 2-17
  security, 2-20
  usage suitability, 2-5
  user interface, 2-15
Portlet Wizard
  generated code for customization, 5-50
  JPS, 5-12
  modifying generated code, 5-51
  PDK-Java, 5-29
PortletDefintion, 5-75
PortletInstance, 5-75
PortletRenderer, 5-49
portlets
  about OmniPortlet, 3-2
  Access tab, A-83
  accessing archived versions, A-86
  adding a Batch button, A-93
  adding a header and footer, 3-10
  adding an OmniPortlet to a page, 3-22
  adding events, 3-20
  adding functionality, A-95
  building, A-13
  building with PL/SQL, 6-8
  bullet layout, 3-17
  caching, PL/SQL, 6-30
  calendar, A-12
  changing the layout, 3-11
  changing view options, 3-10
  chart, A-12
  chart layout, 3-13
  context information PL/SQL, 6-24
  copying, A-84
  creating a struts portlet, 5-105
  creating JPS-compliant, 5-11
  customization export, 5-74
  Customize privilege, A-87
  customizing, 5-48
  data components, A-13
  defined, 1-1
  defining based on a Web page, 3-28
  defining based on a Web Service, 3-26
  defining based on SQL, 3-23
  deleting, A-83
  deploying JPS compliant, 5-20
  deploying PDK-Java compliant, 5-38
  deployment, 2-6
  Develop tab, A-83
  develop-in-place, A-11
  development tool, 2-12
  dynamic page, A-12
  Edit privilege, A-87
  editing, A-81, A-86
  editing as new, A-86
  error handling PL/SQL, 6-36
  event handling PL/SQL, 6-40
  Execute privilege, A-87
  export of customizations, 5-74
  features and characteristics, 2-1
  form, A-12
  form layout, 3-18
  forms, A-15
  frame driver, A-13
  generating PL/SQL, A-84
  granting privileges, A-88
  guidelines, Java, 5-2
  guidelines, PL/SQL
    best practices
      PL/SQL, 6-1
  hierarchy, A-13
  Java, 5-1
  Java Standards, 2-2
  JPS wizard in Oracle JDeveloper, 5-12
  link, A-13
  lists of values, A-13
  Manage privilege, A-87
  Manage tab, A-83
  managing locks, A-85
  managing versions, A-86
  menu, A-13
  modes list, 5-2
  multi-lingual, 5-95
  multi-lingual PL/SQL, 6-43
  news layout, 3-16
  OmniPortlet, 1-9
  out of the box, 1-5
  package, PL/SQL, 6-9
  PDK-Java, 5-29
  PL/SQL, 6-1
  PL/SQL parameters, 6-21
  Portlet Builder, 1-10, A-1 to A-123
  portlet building privileges, A-87
  portlet management page, A-82
  privileges, 5-66
  providers, 2-7, A-1
  publishing generated, 6-7
  renaming, A-83
  report, A-12
  running, A-90
  running as full page, A-91
  running as portlets, A-91

running in batch mode, A-92
running with Customization Form, A-91
schemas, A-1, A-2
security, 5-62
security managers, 5-66
security PL/SQL, 6-27
tabular layout, 3-12
technologies, 2-1
testing, A-90
testing PDK-Java, 5-36
types of, A-12
URLs, A-13
View Source privilege, A-87
viewing call interface, A-85
viewing package spec and body, A-85
viewing source code, A-84
Web Clipping, 1-6
Web source, 1-1
XML component, A-12
preference store, 5-49
creating and accessing in PL/SQL, 6-15
migration utility, 5-49
PL/SQL, 6-15
PrefStoreTransporter, 5-87
Preview mode, 5-6
private portlet parameters, 2-18
privileges, 5-66
access to shared components, A-105
Create shared components, A-105
for building portlets, A-87
granting on portlets, A-88
granting to individuals, A-89
inheriting from provider, A-88
Manage shared components, A-105
portlet, 5-66
provider, 5-66
properties
Web Clipping portlet, 4-8, 4-18
provider servlet
for invalidation based caching, 5-91
ProviderInstance, 5-76
providers, A-1
architecture, 2-10
database providers, 2-9
inheriting privileges from, A-88
locally built providers, A-8
package, PL/SQL, 6-10
Portal DB Provider, A-18
portlet deployment, 2-6
privileges, 5-66
schemas, A-1
shared components, A-104
provider.xml
activating invalidation based caching, 5-93
events, 5-57
multi-lingual, 5-98
parameters, 5-55
portlet resource bundles, 5-100
preference information, 5-52
provider resource bundles, 5-98
session, 5-62
updating for render modes, 5-46
proxy authentication
URL-based portlets and, 4-21
Web Clipping and, 1-8, 4-2, B-10
proxy server setting
Web Clipping and, B-9, B-10
public portlet parameters, 2-18

Q

query wizard, charts, A-57

R

radio group, LOV, A-79
registering
JPS portlet, 5-22
on a local OracleAS Portal, 5-23
on portalstandards.oracle.com, 5-28
PDJ-Java portlet, 5-41
render modes, 5-45
updating provider.xml for, 5-46
report portlets, A-12
audio data, A-54
building, A-31
column break display options, A-45
column conditions, A-36
column formatting options, A-38
common display options, A-43
customization forms, A-48, A-50
date format masks, A-39
examples, A-118, A-121
formatting conditions, A-41
full Web page display options, A-44
image data, A-54
interMedia rich content, A-54
layout types, A-37
mobile device display options, A-47
number format masks, A-40
PL/SQL code options, A-52
row order display options, A-46
supplementary text options, A-52
templates, A-51
video data, A-54
resource bundles, 5-96
updating renderer, 5-97
roles, A-7

S

schemas, A-1, A-2
access privileges, A-4, A-7
creating, A-2
default profile, A-3
default tablespace, A-3
enrolling in roles, A-7
granting privileges, A-4
naming, A-2
temporary tablespace, A-3
sections

Web Clipping and, 4-6
security, 5-62
authentication, 5-62
authorization, 5-63
coding in PL/SQL, 6-28
communication, 5-63
features, 5-62
for export, 5-82
for import, 5-82
for transport communications, 5-82
guidelines for PL/SQL, 6-27
implementing PL/SQL, 6-27
managers, 5-66
message authentication, 5-68
server, 5-68
session
persistent states, A-99
session information, 5-59
checking for valid session, 5-61
enabling in provider.xml, 5-62
storage implementation, 5-60
session store
creating and accessing in PL/SQL, 6-19
PL/SQL, 6-19
set_value, A-98
shared components, A-104
creating JavaScript under, A-107
editing, A-104
granting access, A-105
shared components provider
creating color definitions, A-107
creating image definitions, A-108
font definitions, A-109
user interface templates, A-110
Shared Screen mode, 5-2
Show modes
list, 5-2
Show page, 5-52
Simple Object Access Protocol (SOAP), 2-6
Single Sign-On
Web Clipping and, 1-8, 4-2, 4-9
single sign-on, 5-63
comparison of portlet builders, 2-20
external application, 2-20, 5-64
partner application, 5-64
SOAP, 2-6
spreadsheet
using as a data source, 3-5
SQL
defining for a portlet, 3-23
using as a data source, 3-5
SSL, 5-69
configuration, 5-70
starter provider sample, 6-8
status
checking for Web Clipping, B-9
strings
loading language PL/SQL, 6-44
retrieving language PL/SQL, 6-45
struts
building portlet in JDeveloper, 5-101
creating a portlet, 5-105
OracleAS Portal integration, 5-103
overview, 5-103
substitution tags, A-112
ALIGN_LEFT, A-112
ALIGN_RIGHT, A-113
BODY, A-112
DIRECTION, A-112
HELPLINK, A-112
HELPSCRIPT, A-112
IMAGE_PREFIX, A-112
PAGE.BASE, A-113
PAGE.BASE.URL, A-113
PAGE.BGCOLOR, A-113
PAGE.BGIMAGE, A-113
PAGE.CUSTOMIZEPAGE, A-113
PAGE.CUSTOMIZEPAGE.LABEL, A-114
PAGE.CUSTOMIZEPAGE.URL, A-114
PAGE.EDITPAGE, A-114
PAGE.EDITPAGE.LABEL, A-114
PAGE.EDITPAGE.URL, A-114
PAGE.REFRESH, A-114
PAGE.REFRESH.LABEL, A-114
PAGE.REFRESH.URL, A-114
PAGE.STYLE, A-113
PAGE.STYLE.URL, A-113
PAGE.SUBPAGELINK, A-113
PORTAL.ACCOUNTINFO, A-113
PORTAL.ACCOUNTINFO.LABEL, A-113
PORTAL.ACCOUNTINFO.URL, A-113
PORTAL.COMMUNITY, A-113
PORTAL.COMMUNITY.IMAGE, A-113
PORTAL.COMMUNITY.LABEL, A-113
PORTAL.COMMUNITY.URL, A-113
PORTAL.HELP, A-113
PORTAL.HELP.IMAGE, A-113
PORTAL.HELP.LABEL, A-113
PORTAL.HELP.URL, A-113
PORTAL.HOME, A-113
PORTAL.HOME.IMAGE, A-113
PORTAL.HOME.LABEL, A-113
PORTAL.HOME.URL, A-113
PORTAL.LOGOUT, A-113
PORTAL.LOGOUT.LABEL, A-113
PORTAL.LOGOUT.URL, A-113
PORTAL.NAVIGATOR, A-113
PORTAL.NAVIGATOR.IMAGE, A-113
PORTAL.NAVIGATOR.LABEL, A-113
PORTAL.NAVIGATOR.URL, A-113
USER, A-112
USER.FULLNAME, A-112
VERSION, A-112

T

tablespace
default, A-3
temporary, A-3
tags, substitution, A-112

technologies, portlets, 2-1
templates, A-51
  see also user interface templates, A-110
test page
  Web Clipping, B-9
testing
  portlet and provider for PDK-Java, 5-36
  portlets, A-90
text options
  chart portlets, A-74
  form portlets, A-27
  report portlets, A-52
TextArea item types, form portlets, A-22
TextBox item types, form portlets, A-22
Time Out field
  Web Clipping and, 4-9
transport
  by reference, 5-83
  by reference example, 5-86
  customizations, 5-74
  encrypting for, 5-83
  encryption example, 5-84
  example, 5-77
  JNDI variable, 5-83
  logging interface, 5-76
  programming interface, 5-75
  securing communications, 5-82
  security, 5-82
troubleshooting
  Web Clipping, B-5, B-9

U

URL rewriting field
  Web Clipping and, 4-9
URL-based portlets
  about, A-13
  migrating to Web Clipping, 1-8, 4-2
URLs
  specifying for Web Clipping, 4-5, 4-13
user interface templates, A-110
  structured, A-110
  substitution tags, A-112
  unstructured, A-111
USER substitution tag, A-112
USER.FULLNAME substitution tag, A-112
users
  page designer, xv
  portal developer, xv

V

validation
  field-level, A-106
  form-level, A-106
  JavaScript guidelines, A-106
  JavaScripts, A-106
  writing field-level, A-106
  writing form-level, A-106
validation options, form portlets
  Default Value, A-25
  Default Value Type, A-25
  Field Level Validation, A-25
  Form Level Validation, A-25
  Format Mask, A-25
  Insertable, A-25
  Mandatory, A-25
  Updatable, A-25
validation-based caching, 2-12
variables, persistent state, A-99
version management, A-86
VERSION substitution tag, A-112
video data, A-54
View mode, 5-2
View Source privilege, A-87
virtual directories, images, A-109

W

WAR file
  for JPS, 5-22
  for PDK-Java, 5-39
Web browsers
  recommended cache settings, xviii
  recommended image settings, xix
  recommended versions, xviii
Web Clipping
  authentication for external application, 2-21
  caching style, 2-12
  capturing content, 2-15
  charting, 2-17
  comparing to other portlet builders, 2-2
  development tool, 2-12
  event support, 2-19
  expertise required, 2-5
  forms and, 4-9, 4-19
  limitations, 4-21
  load balancers and, B-9
  logging levels and, B-11
  multi-lingual support, 2-20
  parameter support, 2-19
  personal pages and, 4-15
  proxy authentication, B-10
  proxy servers and, B-9, B-10
  rendering content inline, 2-16
  searching, 4-16
  security, 2-19
  test page, B-9
  troubleshooting, B-5, B-9
  usage suitability, 2-4
  user interface, 2-15
Web Clipping portlet, 1-6, 4-1
  adding, 4-2
  properties, 4-8, 4-18
Web Clipping provider, 4-1
Web Clipping Studio, 4-2
  using, 4-4
Web content
  adding to page, 4-2
  browsing for, 4-4

expiration, 4-9
reuse, 1-8, 4-1
specifying, 4-5, 4-13
timeout value, 4-9
Web page
using as a data source, 3-8, 3-28
Web providers, 2-7
Web Service
defining for a portlet, 3-26
using as a data source, 3-7
Web Services for Remote Portlets
see WSRP
Web source, 1-1
web.xml file
for Web Clipping, B-10
WSRP, 2-2, 2-6, 5-8
providers, 2-8
wwctx_api
context information, 6-25

X

XML
component portlets, A-12
for PL/SQL Generator, 6-4
using as a data source, 3-6
XML provider definition
see provider.xml