Oracle® Fusion Middleware

Developer's Guide for Oracle WebCenter

11*g* Release 1 (11.1.1)

E10148-02

July 2009

ORACLE®

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

E10148-02

Primary Author: Vanessa Wang

Contributing Authors: Lalithashree Rajesh, Michele Cyran, Peter Jacobsen, Promila Chitkara, Rosie Harvey, Savita Thakur, Sue Highmoor

Contributor: Randy Akl, Maheswaran Anantharaman, Sivakumar Balagopalan, Rahmathulla Baig, Ravishankar Belavadi, Chris Broadbent, Steve Burns, Chris Carter, Vince Casarez, Chung Cheng, Demetris Christou, Vicki Chun, Ross Clewley, Manish Devgan, Marcus Diaz, Sumit Dubey, Frans Effendi, Paul Encarnacion, Jeni Ferns, Robin Fisher, Michael Freedman, Nick Greenhalgh, Christian Hauser, Hsing Huang, Clayton Jung, Medini Kakade, Hrishikesh Karambelkar, Seshan Kennan, Sanjay Khanna, Vasant Kumar, Paul Lin, Yueh-Hong Lin, Alison Macmillan, George Maggessy, Pankaj Mittal, Peter Moskovits, Dan Mullen, Lei Oh, Nicolas Pombourcq, Rajesh Ramachandran, Sripathy Rao, Shubha Rangarajan, Shakeb Sagheer, Raghu Sampathkrishna, Skip Sauls, Andrew Sefkow, Jennifer Shu, Ved Singh, Paul Spencer, Stephen Thornhill, Deepthi Umakanth, Kundan Vyas, Alistair Wilson, Stewart Wilson, Hui Zeng

Contents

30 Coding Portlets

31 Creating Portlets with OmniPortlet

List of Examples

List of Figures

List of Tables

Preface

This guide provides in-depth information for all of the following tasks:

- How to plan, build, deploy, and manage a custom WebCenter application with Oracle WebCenter Framework.
- How to administer, monitor, and maintain a custom WebCenter application and all of its associated components with Oracle WebCenter Framework.
- How to prepare an application for, configure, and integrate WebCenter Web 2.0 Services.
- How to plan, build, deploy, and manage portlets for a custom WebCenter application.

Note: For the portable document format (PDF) version of this manual, when a URL breaks onto two lines, the full URL data is not sent to the browser when you click it. To get to the correct target of any URL included in the PDF, copy and paste the URL into your browser's address field. In the HTML version of this manual, you can click a link to directly display its target in your browser.

Audience

This guide assumes the audience has already read and performed the steps in the Oracle Fusion Middleware Tutorial for Oracle WebCenter Developers.

This manual is written for all of the following developers:

- The JSF application developer, who wants to build a custom WebCenter application or add Oracle WebCenter functionality to their application.
- The portal application developer, who wants to integrate Oracle WebCenter functionality into their application
- The component developer, who wants to build portlets from Oracle WebCenter Web 2.0 services.

This guide also assumes that the audience has already read the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework* and is familiar with the following concepts:

- Java
- Oracle JDeveloper
- JavaServer Faces

- Oracle Application Development Framework (Oracle ADF) (purpose, basic architecture, basic development skills)
- Oracle ADF Faces components
- Oracle WebLogic Server

Documentation Accessibility

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

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at `http://www.fcc.gov/cgb/consumerfacts/trs.html`, and a list of phone numbers is available at `http://www.fcc.gov/cgb/dro/trsphonebk.html`.

Related Documents

For more information, see the following documents in the Oracle Fusion Middleware 11*g* Release 1 (11.1.1) documentation set or on Oracle Technology Network (OTN) at `http://www.oracle.com/technology/index.html`.

- *Oracle Fusion Middleware Tutorial for Oracle WebCenter Developers*
- *Oracle Fusion Middleware Error Messages Reference*
- *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*
- *Oracle Fusion Middleware Developer's Guide for Oracle Portal*

Conventions

The following text conventions are used in this document:

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

Part I

Introduction

Part I contains the following chapters:

1 Understanding Oracle WebCenter

This chapter introduces you to Oracle WebCenter and helps you understand what you must consider when building a custom application using the Oracle WebCenter Framework.

This chapter includes the following sections:

Audience

This chapter is intended for developers of custom WebCenter applications. Developers should review this chapter carefully to determine what options are available to them.

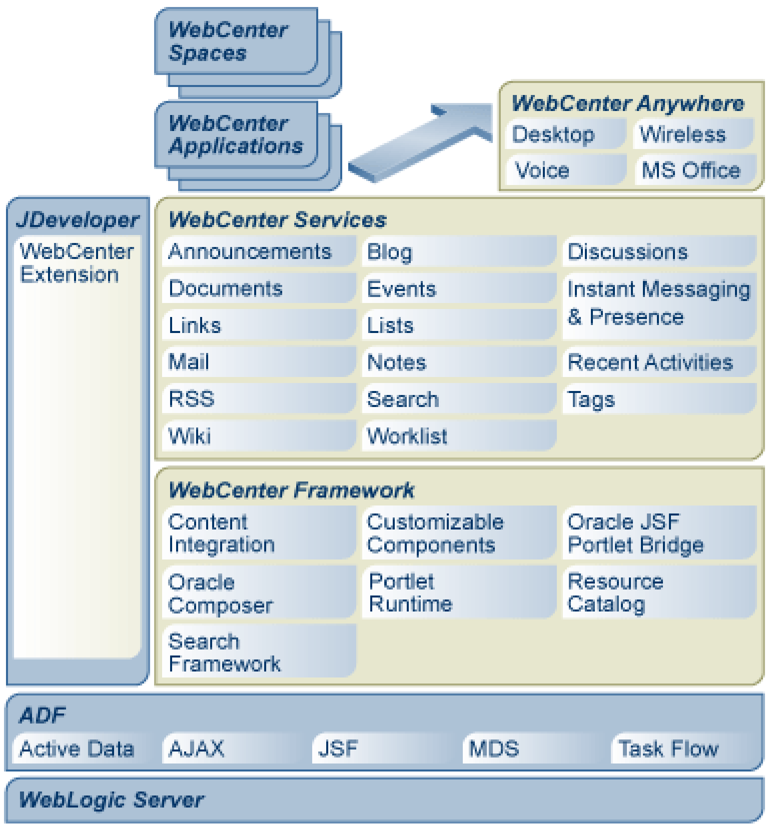
1.1 Introduction to Oracle WebCenter

With the advent of portals, users were suddenly able to interact with the information they were viewing and personalize their experience to match their exact requirements. These users are now demanding the same level of interaction with Web applications. The line between portals and other Web applications has become blurred, and with the proliferation of Web 2.0 technologies such as wiki, RSS, and blogs, this user demand for highly interactive applications can be met.

One way to simplify transactions is to provide everything users need to support a given task within the application itself. Since it is almost impossible to anticipate everything that users need to complete the tasks associated with their particular jobs, the best solution is often to enable users to evolve the application or mash it up with information from other enterprise sources. Oracle WebCenter provides you with a set of features and services (for example, portlets, customization, and content integration) that help you provide such options to your users, and help simplify the transactions users perform through your JSF applications.

Figure 1–1 provides an overview of the Oracle WebCenter architecture, showing the major components that comprise the product and the features and services offered.

Figure 1–1 Overview of Oracle WebCenter Architecture



The next few sections describe the WebCenter components depicted in Figure 1–1:

- Introduction to Oracle Application Development Framework
- Introduction to Oracle JDeveloper and the WebCenter Extension
- Introduction to Oracle WebCenter Framework
- Introduction to Oracle WebCenter Web 2.0 Services
- Introduction to Custom WebCenter Applications

1.1.1 Introduction to Oracle Application Development Framework

The Oracle Application Development Framework (Oracle ADF) is an end-to-end application framework that builds on Java Platform, Enterprise Edition (Java EE) standards and open-source technologies to simplify and accelerate implementing service-oriented applications. Used in tandem, Oracle JDeveloper and Oracle ADF give you an environment that covers the full development lifecycle from design to deployment, with drag-and-drop data binding, visual UI design, and team development features built in.

Oracle ADF uses a set of standard JSF components that include built-in Ajax functionality. Ajax is a combination of asynchronous JavaScript, dynamic HTML (DHTML), and XML. This combination allows requests to be made to the server without fully re-rendering the page. While Ajax allows rich client-like applications to use standard Internet technologies, JSF provides server-side control.

Oracle ADF provides over 100 rich components, including hierarchical data tables, tree menus, in-page dialogs, accordions, dividers, and sortable tables. Oracle ADF also provides data visualization components, which are Flash and SVG-enabled and capable of rendering dynamic charts, graphs, gauges, and other graphics that can provide a real-time view of underlying data. Each component also supports skinning, along with internationalization and accessibility.

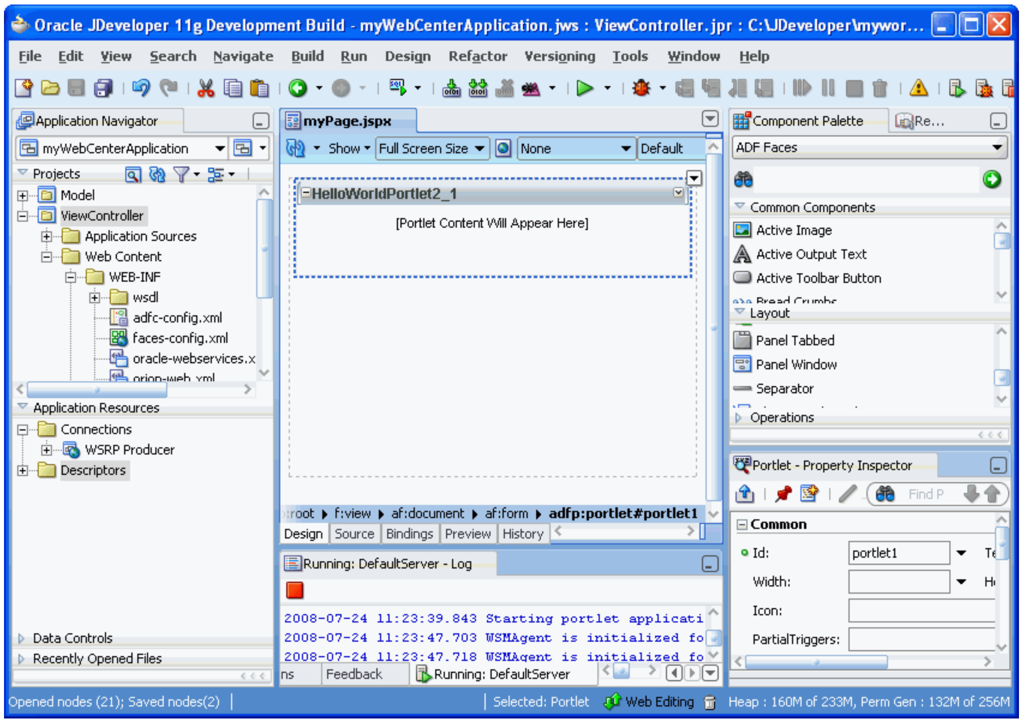
ADF task flows represent a critical new component for you to understand and use in your application development. They provide a modular approach for defining control flow in an application. Instead of representing an application as a single large JSF page flow, you can break it up into a collection of reusable task flows. In each task flow, you identify application activities, the work units that must be performed in order for the application to be complete. An activity represents a piece of work that can be performed when running the task flow. Task flows also have the advantage of being able to be packaged and deployed with the application, rather than requiring a separate deployment like remote portlets. For more information about task flows, see the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

1.1.2 Introduction to Oracle JDeveloper and the WebCenter Extension

Oracle JDeveloper is an integrated development environment (IDE) for building service oriented applications using the latest industry standards for Java, XML, Web services, portlets, and SQL. JDeveloper supports the complete software development life cycle, with integrated features for modeling, coding, debugging, testing, profiling, tuning, and deploying applications. JDeveloper's visual and declarative approach and Oracle ADF work to simplify application development and to reduce mundane coding tasks. For example, code for many standard user interface widgets, such as buttons, lists of values, and navigation bars, are prepackaged for you. All you need to do is select the appropriate widget from the Component Palette and drop it into your application.

The WebCenter Extension packaged with JDeveloper provides all the development functionality you need for building custom WebCenter applications. Oracle WebCenter components are readily accessible from a catalog of resources.

Figure 1–2 Oracle JDeveloper and the WebCenter Extension



```
2008-07-24 11:23:39.843 Starting portlet applicati
2008-07-24 11:23:47.703 WSMAgent is initialized fo
2008-07-24 11:23:47.718 WSMAgent is initialized fo
```

Id: portlet1
Width:
Icon:
PartialTriggers:

The WebCenter Extension also provides several wizards to help you with essential development tasks such as building portlets, consuming an existing portlet, creating a data control to a content repository, and securing your application. By significantly reducing the amount of coding you do, JDeveloper and the WebCenter Extension dramatically increase your productivity as a developer.

In JDeveloper, the easiest way to ensure that you properly define an application and its projects with the appropriate technology scope is to apply an application template. An application template automatically partitions the application into projects that reflect a logical separation of the overall work. The WebCenter Extension provides two templates optimally configured for building custom WebCenter applications:

- **WebCenter Application template.** Prepopulates the application with projects and libraries optimally scaled for the creation of data controls and consumption of portlets and Web 2.0 services.
- **Portlet Producer Application template.** Prepopulates the application with a project and libraries scoped for the creation of portlets.

It is not required that you use these templates. If you prefer, you can create your own custom WebCenter applications and portlet applications by manually scoping the application technologies and creating the relevant projects. For more information, see Chapter 3, "Preparing Your Development Environment."

For more information about JDeveloper, access the many educational aids from the JDeveloper Start Page, accessible from JDeveloper's Help menu.

1.1.3 Introduction to Oracle WebCenter Framework

Oracle WebCenter Framework augments the Oracle ADF environment by providing additional integration and runtime customization options. In essence, it integrates capabilities historically included in portal products, such as portlets, customization, personalization, and integration, directly into the fabric of the JSF environment. This eliminates artificial barriers for the user and provides the foundation for developing context-rich applications.

You can selectively add only desired Oracle WebCenter components or services to your custom WebCenter application. For example, you might only want to add the Instant Messaging and Presence (IMP) service. In this case, you could add just that service without adding all of the other services available with Oracle WebCenter.

This section includes the following subsections:

- Oracle Composer
- Portlets
- Content Integration
- Search Framework
- Resource Catalog
- Customizable Components

1.1.3.1 Oracle Composer

Page *customization*, making updates that affect all users, is very important when branding an application for each deployment or each department. Allowing each user to modify the page to meet their specific needs is commonly referred to as *personalization.* Having an infrastructure that can handle all these customizations and personalizations requires a core set of components to manage and retrieve all the relevant metadata. More importantly, these customizations should be easy to change when the application has been deployed. JSF provides not only the running application code but also a description of the application. Hence, it is key to save customizations and personalizations as layers. This approach ensures that customizations are not lost because you can carry them over when you deploy a new version of the application. As an application developer, you require a set of JSF components that have this customization architecture built in to minimize the complexity of implementing such a solution.

Oracle Composer is an easy, browser-based environment that you can add to existing JSF applications. End users can then use Oracle Composer to edit the page at runtime to create their own mashups. To make pages editable at runtime in this way, you simply add Oracle Composer design time components to a page in JDeveloper. If you create a blank page in JDeveloper with just Oracle Composer components, users have the ability to redesign the page while in Edit mode using Oracle Composer. Some of the tasks they can perform include:

- Editing page settings and parameters
- Adding content to the page
- Editing component properties and parameters
- Wiring components to page parameters
- Arranging content on the page
- Changing the layout for components on the page
- Removing components from the page

The Oracle Composer tag library provides the components that you can add to make a page editable at runtime and define the behavior of content on the page (for example, move, sequence, or hide components).

In Oracle Composer, the Component Catalog dialog box, which is displayed by clicking the Add button on the page, displays the default runtime Resource Catalog Viewer. Users can browse the components in the viewer and then add them to the page. By default, the viewer displays the default Resource Catalog, which contains all of the Oracle ADF components and portlets available to the application. To control what components are visible to users in the viewer, you can modify the default Resource Catalog, or create one or more of your own Resource Catalogs. Resource Catalogs can contain items such as portlets, layout components, task flows, documents, and Oracle ADF Faces components.

For more information about implementing Composer components in your application, see Chapter 4, "Enabling Runtime Editing of Pages Using Oracle Composer."

1.1.3.2 Portlets

Portlets enable users to access a wide range of functions and services. Oracle WebCenter Framework supports deployment and execution of both standards-based portlets (JSR 168, WSRP 1.kj

-0 and 2.0) and traditional Oracle PDK-Java based portlets. It also provides the runtime functions necessary to consume those portlets from within JSF applications. Any existing portlet can be leveraged by a custom WebCenter application with no changes. Developers simply register portlet producers within JDeveloper, and then drag and drop portlets directly onto their JSF pages.

You should be aware that all portlets run remotely from the application in the Oracle WebCenter Framework environment, meaning that there are no local portlets. You must always deploy the producer and register it with the application before consuming its portlets.

Several prebuilt portlets are available for use through a sample producer that you can register with your application. You can use the portlets that Oracle or third parties provide you, or you can create your own portlets programmatically. The prebuilt portlets that the Oracle WebCenter Framework provides include:

- Web Clipping is a browser-based, declarative tool that enables you to navigate to any Web site and clip all or part of the displayed page. For more information, see Chapter 32, "Creating Content-Based Portlets with Web Clipping."
- OmniPortlet is a declarative portlet-building tool through which you can select from any number of data sources, including XML files, character-separated value files (for example, spreadsheets), Web services, database tables, RSS, and Web pages. For more information, see Chapter 31, "Creating Portlets with OmniPortlet."

Packaged applications also often come with their own set of portlets that enable you to access particular data or functions of the application. Assuming that they were built with compatible technology (WSRP, JSR 168, or PDK-Java), you can include these portlets in your custom WebCenter application as well.

Following the Oracle-submitted standard JSR-301, you can expose your task flows as standards based portlets. In this way, one application encompasses both the base application functionality and the portlets to be consumed for integration. When you revise your application, the portlets are naturally and automatically updated right along with it, rather than requiring a separate development project. To support JSR-301, the Oracle WebCenter Framework provides the Oracle JSF Portlet Bridge. For more information about the Oracle JSF Portlet Bridge, see Chapter 28, "Creating Portlets with the Oracle JSF Portlet Bridge."

In addition to consuming task flows as portlets, you can consume task flows as shared libraries in a JSF application to enable you to embed these JSF fragments directly. You can wrap the transactions around the task flows along with the other functionality in their application.

You can link WSRP 2.0 portlets such that parameters are passed between portlets and Faces components, and between portlets and the page. In this fashion, you can create a context-sensitive application, where the data displayed by the portlets changes depending upon the page context. In effect, it enables you to create an enterprise mashup for your users and, if you take advantage of Oracle Composer, your end users can create their own enterprise mashups too.

For information about the different ways you might create portlets and how you might use them, see Chapter 27, "Overview of Portlets." For information about consuming portlets on pages and linking them, see Chapter 9, "Consuming Portlets."

1.1.3.3 Content Integration

JCR (Java Content Repository API, also known as JSR 170) adapters enable you to make data stored in content management systems, such as Oracle Universal Content Management, Oracle Portal, or even your file system, available to your application.

Using JDeveloper, you can use the prebuilt JCR data control to grab the content and drop it onto your page. You can leverage the provided prebuilt user interface to display the content in your custom WebCenter application. This architecture enables you to build your user interface once and then at deployment time or during runtime, switch to whatever back end is required without having to recompile or rebuild the application. In addition, you do not need to learn the intricacies of each content management system's custom APIs. For example, you could create a data control that selects content from any JCR 1.0 compliant repository or file system. When the data control is created, you can drop it onto a JSP document as a table.

If you retrieve data from a content repository other than Oracle Portal or the file system, then you can create your own JCR adapter. From the Content Repository Configuration page of the Create Data Control wizard, you can choose the content repository from which you want to retrieve data.

For more information, see Chapter 8, "Integrating Content."

1.1.3.4 Search Framework

Search is one of the most common and useful features of any application. Oracle WebCenter Framework has a unified, extensible Search framework that enables the discovery of information and people through an intuitive user interface. The powerful Search framework enables you to seamlessly include enterprise-wide search capabilities into your application. With all relevant and secure information easily navigable, users do not need to switch between applications performing multiple searches.

1.1.3.5 Resource Catalog

The Resource Catalog provides a federated view of the contents of one or more otherwise unrelated repositories within a unified search and browse user interface. Resources are created and published in their source repository, then exposed to the developer through the JDeveloper Resource Palette and to the end user through the Resource Catalog Viewer.

Resource Catalogs can contain the following components:

- **Layout components.** The primary layout component is a `Box,` which is a container that can hold all other types of components. At runtime, you must have a `Box` into which you can drag and drop components. You can also add and arrange child components and delete components from a `Box`.

- **Oracle ADF Faces components.** You can add `Text`, `Image`, `Page Link`, `Web Page`, and `Website Link` components to your page. These are analogous to the JDeveloper design time components `Rich Text Editor`, `Image`, `Command Link`, `Web Page`, and `Go Link`, respectively.

  The `Text` component enables you to add rich text on the page. Adding this component invokes a text editor that can be resized and is similar to an HTML editor. You can add text and format it using the options available in the editor.
- **Portlets.** You can add WSRP, JSF, or PDK-Java portlets from any producer that was registered in JDeveloper. For more information about registering portlet producers and adding portlets to a page, see Chapter 9, "Consuming Portlets."
- **Task Flows.** If you have created task flows in JDeveloper, you can add these task flows from JAR files onto your page at runtime. For more information about ADF task flows, see the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.
- **Documents.** If you have configured the Documents service in your application, you can add documents from this service. For more information about the Documents service, see Chapter 14, "Integrating the Documents Service."

1.1.3.6 Customizable Components

To make any JSF JSP document (`.jspx`) editable at runtime, you must add Oracle Composer components to your page in JDeveloper. Depending on whether you want to enable personalization or customization of a page, there are two separate libraries to consider.

- **Oracle Composer Components.** Use these components if your page contains ADF Faces components. These components also make a page editable at runtime and define the behavior of content on a page at runtime (for example, move, sequence, or hide components).
- **Oracle Composer HTML Components.** Use these component if your page contains Trinidad components. These components are also useful for enabling personalization only on the page at runtime.

For more information, see Chapter 4, "Enabling Runtime Editing of Pages Using Oracle Composer."

1.1.4 Introduction to Oracle WebCenter Web 2.0 Services

Oracle WebCenter Web 2.0 Services expose social networking and personal productivity features through services, which, in turn, expose subsets of their features and functionality through task flows.

Note: Oracle WebCenter Services refers to a specific packaging option when purchasing WebCenter. WebCenter Web 2.0 Services refers to the services provided by WebCenter.

WebCenter Web 2.0 Services can be organized into three groups:

- **Social Networking Services:** These services link users and facilitate communication. For example, use the Announcement service to broadcast a team-wide message instantly or at a time you choose. Use the Wiki service to collaborate on team content and the Discussions service to bring experts together to answer questions and solve problems for all team members.

- **Personal Productivity Services:** These services are focused on the requirements of the individual rather than the group. Some of these services are designed to work with standard tools, such as a mail application. Others provide custom functionality, such as the Worklist, which displays the tasks assigned to you from external applications.
- **Shared Services:** These services provide features for both social networking and personal productivity. For example, the Links service enables users to create instant navigation between two application objects. In a multicolumn list that itemizes a project's development effort, you can link each list row to the relevant design document, functional specification, assignment matrix, and so on. All users benefit from the associations you make between two application objects.

Figure 1–3 shows how the WebCenter Web 2.0 Services are classified.

Figure 1–3 Classification of Oracle WebCenter Web 2.0 Services

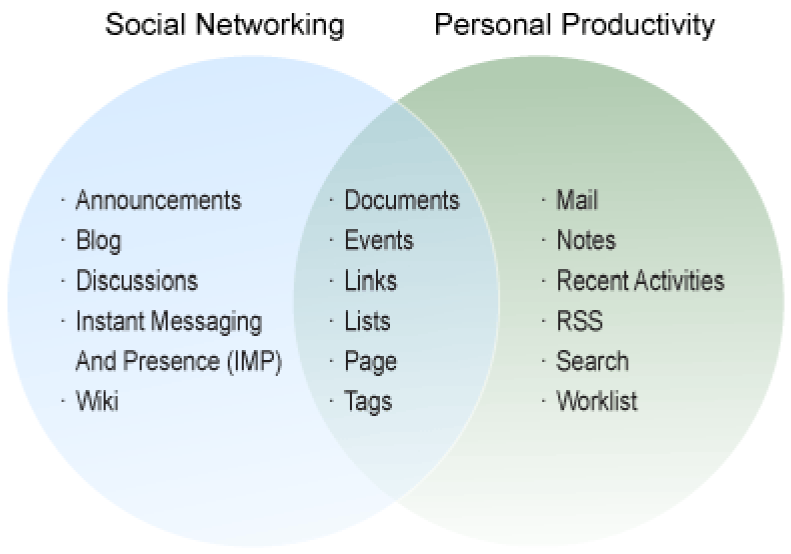


Table 1–1 describes each of the available services.

Table 1–1 Oracle WebCenter Web 2.0 Services

Service	Description	Chapter
Announcements	Provides the ability to post announcements about important activities and events to all authenticated users.	Chapter 12, "Integrating the Announcements Service"
Blog	Provides easy integration of a blog application within the context of your application.	Chapter 22, "Integrating Oracle WebCenter Wiki and Blog Server"
Discussions	Provides the ability to create threaded discussions, posing and responding to questions and searching for answers. Also provides an effective group communication mechanism for important activities and events.	Chapter 13, "Integrating the Discussions Service"

Table 1–1 (Cont.) Oracle WebCenter Web 2.0 Services

Service	Description	Chapter
Documents	Provides content management and storage capabilities, including content upload, file and folder creation and management, file check out, versioning, and so on.	Chapter 14, "Integrating the Documents Service"
Events	Provides the ability to create and maintain a schedule of events relevant to a wider group of users. Events are published to all authenticated users. **Note:** This service is available only in Oracle WebCenter Spaces.	"Working with the Events Service" in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*
Instant Messaging and Presence (IMP)	Provides the ability to observe the status of other authenticated users (whether online, offline, busy, or idle) and to contact them instantly.	Chapter 15, "Integrating the Instant Messaging and Presence Service"
Links	Provides the ability to view, access, and associate related information; for example you can link to a solution document from a discussion thread.	Chapter 16, "Integrating the Links Service"
Lists	Provides the ability to create, publish, and manage lists. Users can create lists from prebuilt structures or create their own custom lists. **Note:** This service is available only in Oracle WebCenter Spaces.	"Working with the Lists Service" in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*
Mail	Provides easy integration with IMAP and SMTP mail servers to enable users to perform simple mail functions such as viewing, reading, creating, and deleting messages, creating messages with attachments, and replying to or forwarding existing messages.	Chapter 17, "Integrating the Mail Service"
Notes	Provides the ability to "jot down" and retain quick bits of personally relevant information. **Note:** This service is available only in Oracle WebCenter Spaces.	"Working with the Notes Service" in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*
Page	Provides the ability to create and manage pages at runtime.	Chapter 7, "Enabling Runtime Creation and Management of Pages"
Recent Activities	Provides a summary view of recent changes to documents, discussions, and announcements.	Chapter 18, "Integrating the Recent Activities Service"
RSS	Provides the ability to access the content of many different Web sites from a single location—a news reader.	Chapter 19, "Integrating the RSS Service"
Search	Provides the ability to search tags, services, the application, or an entire site. This includes integrating Oracle Secure Enterprise Search for WebCenter searches.	Chapter 20, "Integrating the Search Service"
Tags	Provides the ability to assign one or more personally relevant keywords to a given page or document.	Chapter 21, "Integrating the Tags Service"
Wiki	Provides the ability for geographically diverse teams to originate and collaborate on Web documents.	Chapter 22, "Integrating Oracle WebCenter Wiki and Blog Server"

Table 1–1 (Cont.) Oracle WebCenter Web 2.0 Services

Service	Description	Chapter
Worklists	Provides a personal, at-a-glance view of business processes that require attention. These can include a request for document review, and other types of business process that come directly from enterprise applications.	Chapter 23, "Integrating the Worklist Service"

WebCenter provides task flows for these services through the WebCenter Services Catalog. For information about how to prepare your application to consume these task flows, see Chapter 11, "Preparing Your Application for Oracle WebCenter Web 2.0 Services."

How WebCenter Web 2.0 Services Integrate with Each Other

WebCenter Web 2.0 Services are designed to be integrated with each other. Here are some examples of the ways WebCenter Web 2.0 Services work together:

- The Search service is designed to search any WebCenter Web 2.0 service in your application. For example, if you have the Documents service in your application, then the search operation looks for terms in document names and in the documents themselves. The results include tag hits, so any matching tags on your documents and pages are also returned, with relevance based on the quality and frequency of tags.
- The Links service lets you view, access, and associate related information across services. For example, in a discussion forum, you can link to a related discussion forum, document, URL, or event. The Links service also supports bidirectional links between objects, such as a link from a discussion topic to a document and back to the discussion topic.
- The Presence icon can appear in any WebCenter Web 2.0 service where a user is indicated. For example, click the author's name on a document to invoke a context menu and send an instant message or mail to that person.
- The Recent Activities service lets you view the most recent additions or changes to the announcements, documents, discussions, and pages in your application.

1.1.5 Introduction to Custom WebCenter Applications

An application built using Oracle WebCenter can employ some or all of the following elements:

- The Oracle WebCenter Framework to provide declarative security, life cycle management tools, and the ability to consume portlets, WebCenter Web 2.0 Services, and content in a JSF application.
- Oracle Composer to provide a runtime customization tool that enables your users to participate in and evolve the application in a managed way. Composer also enables users at runtime to add the WebCenter Web 2.0 Services that they need to tailor their applications to meet their specific needs.
- WebCenter Web 2.0 Services to provide the foundation of a social network and enable improved communication, content management capabilities, customization, and advanced searching, tagging, and linking support.
- Content repository data controls to integrate content from multiple content repositories into your application.

- Portlets to display, personalize, and reuse dynamic content.
- JSF skins to define a consistent look and feel for your application.
- Policy- and role-based security to control end user access and privileges.

Oracle WebCenter's commitment to Service Oriented Architecture (SOA), and to the JCR 1.0 Java Content Repository and other community standards, means that front-end labor, for example needing to become familiar with things such as the APIs of the back-end systems of content stores, is no longer necessary. You get a wide range of plug-and-play products, tools, and services that make it easy to build the applications that your users need.

After you have built and tested your custom WebCenter application, you must still deploy it for your end users. When deployed and running, users begin to access the application and administrators to maintain it.

This section includes the following subsections:

- Application Security
- Application Life Cycle

1.1.5.1 Application Security

With the Oracle ADF extensions provided in Oracle WebCenter, you can define security for an entire application, a page within the application, or for individual actions provided by customizable components. In addition, for WebCenter Web 2.0 Services that connect to back-end servers using Web Services, you can provide secure identity propagation with WS-Security.

Because Oracle WebCenter security is based on the JAAS and J2EE standards, enterprise roles defined in the existing identity management store can be leveraged directly when securing a custom WebCenter application. You need not synchronize roles within the application being built. It just references and uses the defined users and roles directly. Note also that you can use file-based security for the development phase of the application and then easily switch over to enterprise identity management at deployment time.

Oracle WebCenter also provides application roles that you can use to represent the policy of an application. By associating permissions with an application role defined within the policy store, you can keep them self contained within the application. On deployment, you can then associate users and enterprise roles with the application roles to grant those permissions to end users.

In some cases, it is desirable to leverage existing applications that have their own authentication mechanism, such as email. The email system is often on a different authentication system (with different user names and passwords) than the new application. The application must map the email user to the application user such that end users do not have to enter their user names and passwords each time they need information. Oracle WebCenter Framework provides the means to securely manage these user names and passwords with the External Application functionality.

For more information about options for securing your custom WebCenter application, see Chapter 24, "Securing Your WebCenter Application."

1.1.5.2 Application Life Cycle

After you have created and tested your custom WebCenter application in the design time environment (JDeveloper), you must deploy it to your production system. When deployed, you must then maintain the system, which includes monitoring performance and availability, editing or refreshing portlet producers, undeploying applications, and perhaps migrating customization data. Inevitably, at some point you will also want to further enhance the application, stage it again, and then redeploy it to your production system.

WebCenter is built on top of JDeveloper and Oracle ADF, which provide several benefits in the lifecycle management of your application:

- **Development framework:** JDeveloper and Oracle ADF provide the tools and framework you can use to build and update your application. Adding portlets, content, and customization capabilities to your custom WebCenter application is simply a matter of dragging and dropping the appropriate objects in either a source or WYSIWYG environment.
- **Enterprise deployment:** When you are ready to deploy your application to a production environment, JDeveloper packages everything that the JSF/Oracle ADF application needs in the EAR file for easy deployment though Enterprise Manager to the production system. For more information, see Chapter 25, "Testing and Deploying Your WebCenter Application."

  Note: WebCenter Web 2.0 Services typically require some back end, such as Oracle Discussions Server or Oracle Wiki Server, to be available in the deployment environment.

- **Standards-based administration:** Browser-based tools enable administrators to deploy, configure, and manage custom WebCenter applications and WebCenter Web 2.0 Services. In addition, tools built on industry standards-based JMX methods offer administrators granular control and monitoring mechanisms for health status, performance, and popularity. Tools for obtaining historical performance and status reporting over time (within a single Oracle Application Server context) are also provided. Custom WebCenter application metrics are delivered using the familiar Application Server Control monitoring and management interface. For more information, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

1.2 Design Questions to Consider Before You Start

When you design your custom WebCenter application, you must consider the needs of your audience. In particular, it is important to think about what features and capabilities your custom WebCenter application end users, administrators, and developers need. Before you begin to actually build a custom WebCenter application, take a look through the questions listed below and use the answers to help plan your application.

This section includes the following subsections:

- User Considerations
- Site Administrator Considerations
- Developer Considerations

1.2.1 User Considerations

The following list suggests questions you should ask about the end users of your application:

- **How many users are there?** If your custom WebCenter application must serve a large number of users, then you must take that into account when you define your deployment environment.That is, you must choose a topology that can support the number of users you expect to access your custom WebCenter application and set the configuration parameters of the different services accordingly.

To help you decide which Oracle Application Server configuration best supports your custom WebCenter application, see the *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle WebCenter*.

- **Do users need to edit pages and create enterprise mashups by adding resources?** If you plan to let users edit pages, you must add Oracle Composer components to your application. You should also consider issues such as which areas you want to make editable and what resources to make available to users.

  When you design your application, you must choose where to insert the Oracle Composer components to make those areas of the page editable. Components that allow users to add to the page must appear in a Resource Catalog at runtime. You can also choose to have different catalogs available depending upon the context of the application or the user's privileges.

  If you choose to allow page editing, the application must be able to associate the personalizations with particular users, which means some form of authentication is required for users of the application. Furthermore, since the user's personalizations are stored in Metadata Storage (MDS), you must ensure that your deployment system has access to MDS for the storage an retrieval of personalizations.

  For more information about the Oracle Composer and exposing resource catalogs, see Chapter 4, "Enabling Runtime Editing of Pages Using Oracle Composer."

- **Do users need to personalize their portlets?** Portlets can optionally include a personalization mode (Edit mode) that enables an authenticated user to personalize the portlet. The Edit mode might allow the user to enter things such as the portlet's title or a parameter that affects the content of the portlet. For example, you could implement an Edit mode that enables the user to enter ticker symbols for a stock portlet. When the user's changes are applied, the portlet displays the prices for their chosen ticker symbols.

  If you choose to implement personalization, then you must also implement some form of security for the application consuming the portlet. The Personalize option appears only to authenticated users.

  In considering portlet personalization, you should also consider whether the portlets are load balanced. If they are load balanced, then you must use database backed personalization to ensure that all of your middle tiers can access to the user's personalization data. Otherwise, the user has to personalize the same portlet multiple times.

  For more information, see Chapter 27, "Overview of Portlets."

- **Which services do your users need to access?** WebCenter provides a variety of Web 2.0 services, such as Documents, Discussions, Search, and Instant Messaging and Presence. You must decide which of these services your users need and then configure those services. When the desired services are installed and configured, you can add task flows to your application pages that access those services. The services that you choose to implement can also impact the topology you choose. For example, you might choose to have your discussions server and content repositories reside on different systems, which must both be accessible to your application.

  For more information, see Chapter 11, "Preparing Your Application for Oracle WebCenter Web 2.0 Services."

- **Will there be both authenticated and unauthenticated (public) users?** Several of the WebCenter Web 2.0 services offer limited functionality for unauthenticated users. For example, unauthenticated users can view but not contribute to discussions. Due to this loss of functionality, it sometimes works better to create special public pages designed just for unauthenticated users.

1.2.2 Site Administrator Considerations

The following list suggests questions you should ask about the administrators of your application:

Note: It is critical that the custom WebCenter application administrator and the developers communicate when the application is under construction. At design time, developers must make many choices that determine what the administrator can do to the application at runtime. For example, if the developers choose not to implement skins, then the administrator has no control over the look and feel of the application. Hence, the administrator and the developers should ensure that they consult over these decisions at design time.

- **Do administrators need to customize pages and portlets for users or user groups?** You may want to enable your custom WebCenter application administrator to customize pages and portlets to provide default views of the custom WebCenter application for users.
  - **Custom WebCenter application look and feel.** By implementing skins for your custom WebCenter application, you enable administrators to choose the look of the pages in the application.

    For information about how to implement skins, see Chapter 19 "Customizing the Appearance Using Styles and Skins" in *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*.
  - **Customizable Pages.** The administrator can customize the layout of the page.

    For information about customizable components, see Chapter 4, "Enabling Runtime Editing of Pages Using Oracle Composer."
  - **Portlets.** Portlets can optionally include a customization mode (Edit Defaults mode) that enables the administrator to customize the portlet. The administrator's customization is treated as the default view of the portlet for all other users.

    For more information about portlets, see Chapter 27, "Overview of Portlets."

1.2.3 Developer Considerations

The following list suggests questions you should ask about the developers who will contribute to your application:

- **Should developers build task flows or portlets as reusable components?** Thanks to the Oracle JSF Portlet Bridge, developers can make their task flows into JSR 168 portlets. Hence, they have the option of creating reusable components either as task flows, which can potentially be made into JSR 168 portlets through the bridge, or directly as JSR 168 portlets. If the primary purpose of the component is inclusion in a portal, then developers may prefer to build it directly as a JSR 168 portlet, without going through the bridge. If the primary purpose of the component is local execution in a JSF application, then the developer may prefer to create it as a task flow first and then, optionally, use the bridge to run it as a JSR 168 portlet. Building task flows means that the components can be used in other applications without having to repackage them as portlets first.

- **Do developers need to integrate content from content repositories (for example, Oracle Content Server, file system, or Oracle Portal)?** If you have content located in content repositories to expose in the custom WebCenter application, then you must use JCR adapters and data controls to make that content available.

  For information about including content through JCR data controls and adapters, see Chapter 8, "Integrating Content."

- **Will the visible/editable content of the custom WebCenter application vary depending upon the identity of the user?** If you plan to have content in your custom WebCenter application that is not for everyone, then you must set up a security model with login and user roles that enable you to control access. Components, portlets, and pages can display or not display, depending upon a user's identity. If the content is open to everyone (for example, a Human Resources custom WebCenter application with content for all employees), then you might not need to implement access control.

  Furthermore, with Oracle Composer, you can allow users to edit the content within the application. Again, you must consider whether you want to restrict that capability to certain users and, if so, you must factor that into your security model.

  For more information about implementing a security model for your custom WebCenter application, see Chapter 24, "Securing Your WebCenter Application."

- **Does your custom WebCenter application need to provide access to external applications?** In many cases, you might have external applications to surface in your custom WebCenter application. For example, you might have email, Human Resources, or financial applications that you would like users to be able to view from within the application. To provide seamless interaction for the user, you must employ an external application credential store of some kind.

  For more information about accessing external applications from your custom WebCenter application, see Chapter 24, "Securing Your WebCenter Application."

  If you have legacy external application portlets that were built for Oracle Portal, then you can reuse those portlets in a custom WebCenter application. This feature allows users to view these portlets from their custom WebCenter application rather than having to open each application separately or go back to Oracle Portal. When building such portlets, though, you must remember that they typically must authenticate themselves to the external application before retrieving and displaying any data. As previously mentioned, such authentication requires a credential vault, where the custom WebCenter application can store the credentials necessary for logging into the external application. Oracle WebCenter Framework provides choices in Oracle JDeveloper for incorporating portlets based on external applications.

  For more information about external application portlets, see Section 24.2.3, "Managing External Applications."

- **Will the custom WebCenter application be developed by a team of developers?** If your custom WebCenter application is being developed by a team, then you must think about some additional design considerations. For example, when doing team development, you must be much more aware of which files various JDeveloper actions touch, and may want to take advantage of JDeveloper's ability to integrate code management systems.

For more information about team development, see Chapter 26, "Working Productively in Teams."

1.3 Accessibility Features

Accessibility involves making your application usable by persons with disabilities such as low vision or blindness, deafness, or other physical limitations. In the simplest of terms, this means creating applications that can be used without a mouse (keyboard only), used with a screen reader for blind or low-vision users, and used without reliance on sound, color, or animation and timing.

Oracle software implements the standards of Section 508 and WCAG 1.0 AA using an interpretation of the standards at `http://www.oracle.com/accessibility/standards.html`.

This section describes accessibility features that are specific to WebCenter. For general information about creating accessible ADF Faces pages, see the Developing Accessible ADF Faces Components and Pages section in the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework.* For information about accessibility features in JDeveloper, see the help topics available by selecting the JDeveloper Accessibility node under JDeveloper Basics in the online help table of contents.

1.3.1 Generating Accessible HTML

WebCenter provides several Composer components that you can add to your application pages to make them editable at runtime. These components provide attributes that are used to generate accessible HTML. To ensure that the pages you create are accessible, you must set these attributes, listed in Table 1–2.

Table 1–2 Accessibility Attributes for Oracle Composer Components

Component	Accessibility Attributes
`pe:changeModeButton`	No accessibility attributes.
`pe:changeModeLink`	No accessibility attributes.
`pe:imageLink`	`shortDesc`—Mandatory. This attribute transforms into an HTML `alt` attribute. `accessKey`—Optional. This attribute sets the mnemonic character used to gain quick access to the component.
`pe:pageCustomizable`	No accessibility attributes
`pe:layoutCustomizable`	`shortDesc`—Mandatory. This attribute transforms into an HTML `alt` attribute. `accessKey`—Optional. This attribute sets the mnemonic character used to gain quick access to the component.
`cust:panelCustomizable`	No accessibility attributes
`cust:showDetailFrame`	`shortDesc`—Mandatory. This attribute transforms into an HTML `alt` attribute.

1.3.2 Accessibility Features at Runtime

When you enable users to customize a page at runtime, you must ensure that any customizations are also accessible to all users. For all components that users can create at runtime, all accessibility-related attributes are shown in the Property Inspector where users can set them appropriately.

For a list of accessibility-related attributes for WebCenter-specific components, see Table 1–2. For a list of accessibility-related attributes for other components, see the Developing Accessible ADF Faces Components and Pages section in the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework.*

1.3.3 Accessibility Considerations for Portlets

HTML standards do not allow the use of nested `form` tags in page content. However, JavaServer Faces pages almost always include a `form` tag so the JSF reference implementation, which is what is used to generate the markup, puts the entire page body inside a `form` tag. If you then have a portlet on that Faces page that includes a `form` tag in its content, which is frequently the case, you have a nested `form` tag. The common resolution for nested `form` tags is to put the inner form inside an iframe.

However, iframes are not very well accommodated by today's screen readers and so are not permitted by some accessibility standards.

To address the situation of nested `form` tags and iframes, WebCenter offers two solutions:

- WebCenter automatically parses the HTML markup from portlets looking for `form` tags and rewrites them and the corresponding `submit` tags to be Faces compliant using JSF's `subform` mechanisms. However, this HTML parsing can fail, especially given the growing use of complex, AJAX style markup.
- WebCenter provides an optional attribute in the `adf:portlet` tag called `renderPortletInIFrame` that has three settings:
  - `auto` (default) - parse the markup and, if the parsing gets an error because it cannot understand it properly, render the content inside an iframe, otherwise render the content inline.
  - `true` - always render an iframe.
  - `false` - never render an iframe.

1.3.4 Accessibility Features in WebCenter Spaces

WebCenter Spaces pages has been designed to meet accessibility guidelines.

2 Introduction to the WebCenter Sample Application

As a companion to this guide, a sample enterprise application was developed using Oracle Fusion Middleware 11*g* Release 1 (11.1.1), including Oracle WebCenter Framework. This sample application, called the Fusion Order Demo for WebCenter, is used as an example throughout this guide to illustrate points and provide samples.

In this chapter, you will learn about the Fusion Order Demo for WebCenter, as well as take a quick tour of the components built using Oracle WebCenter Framework.

This chapter includes the following sections:

- Section 2.1, "About the Fusion Order Demo for WebCenter"
- Section 2.2, "Browsing the Oracle WebCenter Framework Components in the Fusion Order Demo for WebCenter at Runtime"
- Section 2.3, "Oracle WebCenter Framework Components in the Fusion Order Demo for WebCenter"

2.1 About the Fusion Order Demo for WebCenter

The Fusion Order Demo for WebCenter showcases Oracle Fusion Middleware, including Oracle WebCenter Framework, in an enterprise, integrated application. In this sample application, electronic devices are sold through a storefront-type web application. Customers can browse various products, place orders, and check the status of their orders. In this application, you can see Oracle WebCenter Framework features being used to provide Web 2.0, customization, and personalization capabilities.

In order to view and run the demo, you need to install Oracle JDeveloper 11*g* (11.1.1) and the Oracle WebCenter extension (11.1.1), as described in Section 3.1, "Installing the WebCenter Extension Bundle." You then need to download the application for this demonstration. Once the application is installed and running, you can view the application at design time in Oracle JDeveloper and at runtime by logging in as an existing customer.

For information on downloading and installing the sample application and the associated schema, refer to Chapter 2, "Introduction to the ADF Sample Application" in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

For more details on the Fusion Order Demo for WebCenter, refer to the Oracle WebCenter Suite 11*g* Demonstrations and Samples page on the Oracle Technology Network at `http://webcenter.oracle.com`, which contains information about the Fusion Order Demo for WebCenter, as well as other sample applications created with Oracle WebCenter Framework.

2.2 Browsing the Oracle WebCenter Framework Components in the Fusion Order Demo for WebCenter at Runtime

Once you have downloaded and installed the demo, you can run the application and navigate to the components built using Oracle WebCenter. This section will take you on a visual tour of the application and show you the components built using Oracle WebCenter Framework.

This section focuses on how the various Oracle WebCenter components display at runtime. You can examine the application at design time in Oracle JDeveloper. For more information on how you can add a service or feature to your application, refer to the appropriate chapter in this guide. Table 2–1 provides a quick reference to the relationship between the features you will see in this section and the chapters in the guide that show you how to implement them.

2.2.1 Exploring the Home.jspx Page in the Fusion Order Demo for WebCenter

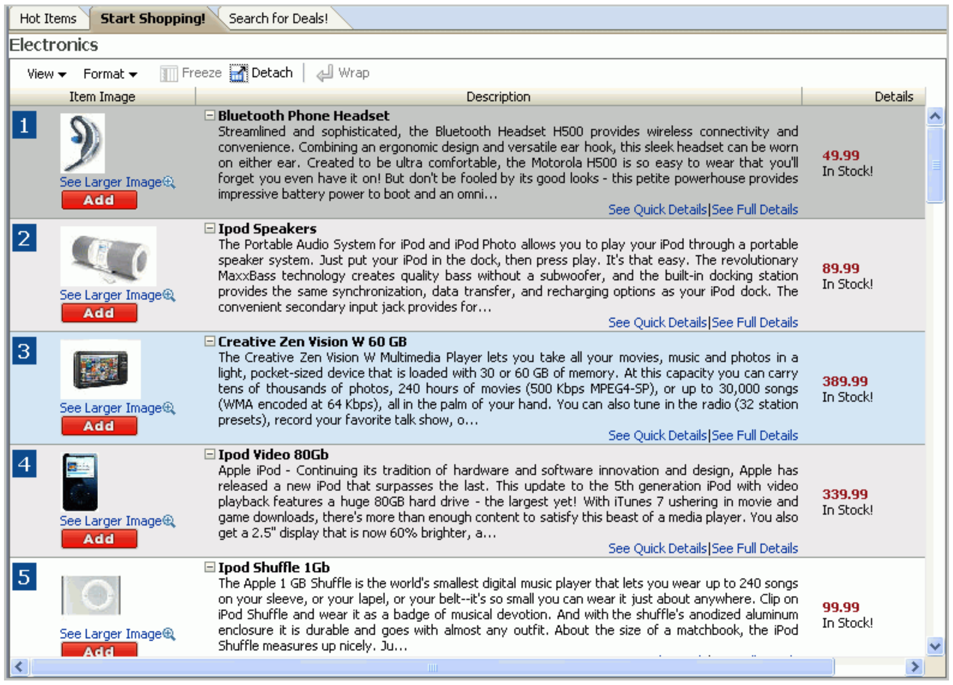
Let's take a look at the Home page, which showcases the following Oracle WebCenter Framework features:

- Documents service
- Links service
- Tags service
- Discussions service

To view these features in the Fusion Order Demo for WebCenter application:

1. Open the application in Oracle JDeveloper and run the `home.jspx` page to view the components built with Oracle WebCenter. You'll notice that the center of the page looks like Figure 2–1.

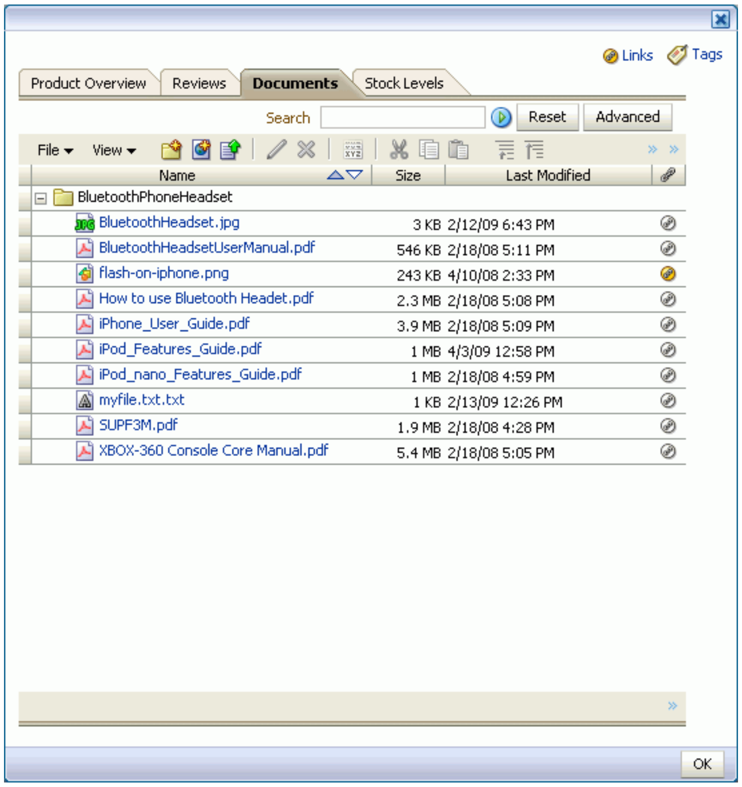
Figure 2–1 Sample Fusion Order Demo for WebCenter Home Page



	Item Image	Description	Details
1	See Larger Image Add	**Bluetooth Phone Headset** Streamlined and sophisticated, the Bluetooth Headset H500 provides wireless connectivity and convenience. Combining an ergonomic design and versatile ear hook, this sleek headset can be worn on either ear. Created to be ultra comfortable, the Motorola H500 is so easy to wear that you'll forget you even have it on! But don't be fooled by its good looks - this petite powerhouse provides impressive battery power to boot and an omni... See Quick Details|See Full Details	49.99 In Stock!
2	See Larger Image Add	**Ipod Speakers** The Portable Audio System for iPod and iPod Photo allows you to play your iPod through a portable speaker system. Just put your iPod in the dock, then press play. It's that easy. The revolutionary MaxxBass technology creates quality bass without a subwoofer, and the built-in docking station provides the same synchronization, data transfer, and recharging options as your iPod dock. The convenient secondary input jack provides for... See Quick Details|See Full Details	89.99 In Stock!
3	See Larger Image Add	**Creative Zen Vision W 60 GB** The Creative Zen Vision W Multimedia Player lets you take all your movies, music and photos in a light, pocket-sized device that is loaded with 30 or 60 GB of memory. At this capacity you can carry tens of thousands of photos, 240 hours of movies (500 Kbps MPEG4-SP), or up to 30,000 songs (WMA encoded at 64 Kbps), all in the palm of your hand. You can also tune in the radio (32 station presets), record your favorite talk show, o... See Quick Details|See Full Details	389.99 In Stock!
4	See Larger Image Add	**Ipod Video 80Gb** Apple iPod - Continuing its tradition of hardware and software innovation and design, Apple has released a new iPod that surpasses the last. This update to the 5th generation iPod with video playback features a huge 80GB hard drive - the largest yet! With iTunes 7 ushering in movie and game downloads, there's more than enough content to satisfy this beast of a media player. You also get a 2.5" display that is now 60% brighter, a... See Quick Details|See Full Details	339.99 In Stock!
5	See Larger Image Add	**Ipod Shuffle 1Gb** The Apple 1 GB Shuffle is the world's smallest digital music player that lets you wear up to 240 songs on your sleeve, or your lapel, or your belt--it's so small you can wear it just about anywhere. Clip on iPod Shuffle and wear it as a badge of musical devotion. And with the shuffle's anodized aluminum enclosure it is durable and goes with almost any outfit. About the size of a matchbook, the iPod Shuffle measures up nicely. Ju...	99.99 In Stock!

2. Below one of the product names, click **See Full Details** to view a few of the features built with Oracle WebCenter Framework. You'll first notice the **Documents** tab selected, and a list of contents. This component was built using the Documents service (Chapter 14, "Integrating the Documents Service").

Figure 2–2 Documents tab



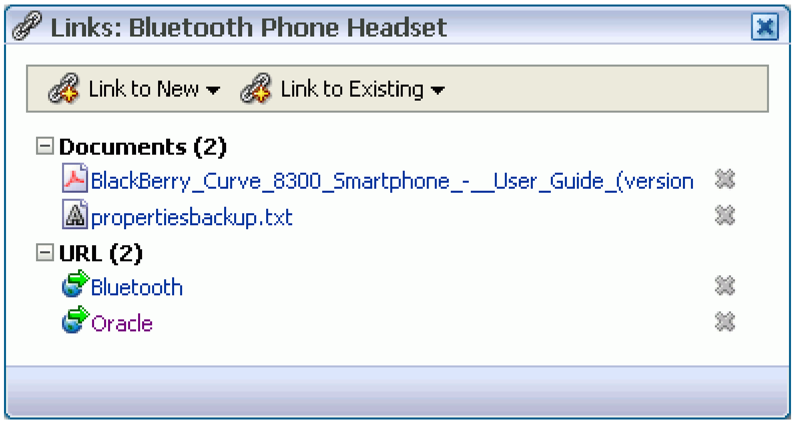
Name	Size	Last Modified
BluetoothPhoneHeadset		
BluetoothHeadset.jpg	3 KB	2/12/09 6:43 PM
BluetoothHeadsetUserManual.pdf	546 KB	2/18/08 5:11 PM
flash-on-iphone.png	243 KB	4/10/08 2:33 PM
How to use Bluetooth Headet.pdf	2.3 MB	2/18/08 5:08 PM
iPhone_User_Guide.pdf	3.9 MB	2/18/08 5:09 PM
iPod_Features_Guide.pdf	1 MB	4/3/09 12:58 PM
iPod_nano_Features_Guide.pdf	1 MB	2/18/08 4:59 PM
myfile.txt.txt	1 KB	2/13/09 12:26 PM
SUPF3M.pdf	1.9 MB	2/18/08 4:28 PM
XBOX-360 Console Core Manual.pdf	5.4 MB	2/18/08 5:05 PM

3. Notice two components above the tabs, called **Links** and **Tags** (built using the Links service and the Tags service)

Figure 2–3 Links and Tags

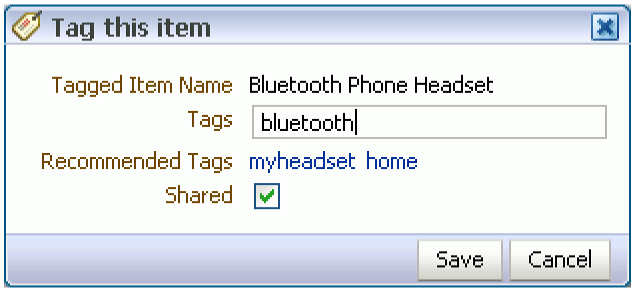
4. You can click each of these links to see what displays, but to learn more about each of these features, refer to Chapter 16, "Integrating the Links Service" and Chapter 21, "Integrating the Tags Service."

Figure 2–4 Links in the Fusion Order Demo for WebCenter



Documents (2)
BlackBerry_Curve_8300_Smartphone_-__User_Guide_(version
propertiesbackup.txt
URL (2)
Bluetooth
Oracle

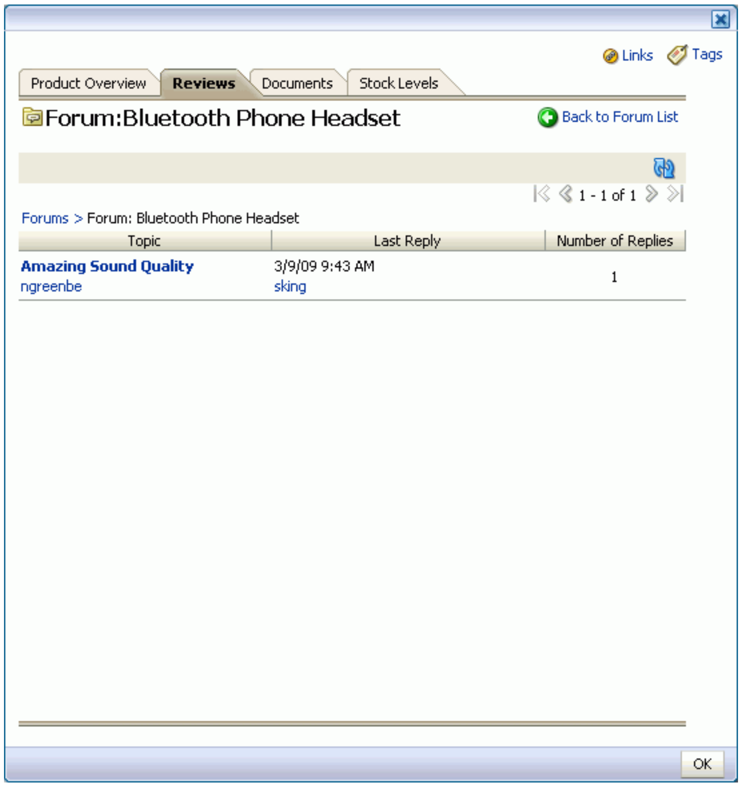
Figure 2–5 Tags in the Fusion Order Demo for WebCenter



Tagged Item Name: Bluetooth Phone Headset
Tags: bluetooth
Recommended Tags: myheadset home
Shared: ✓

5. Now, let's browse the other tabs. Click the **Reviews** tab. Here, you'll see product reviews by users, which is an example of the Discussions service (Chapter 13, "Integrating the Discussions Service").

Figure 2–6 Reviews Tab of the Fusion Order Demo for WebCenter



Topic	Last Reply	Number of Replies
Amazing Sound Quality ngreenbe	3/9/09 9:43 AM sking	1

6. At the bottom of the page, click **OK** to exit this portion of the demo.

Now, let's take a look at some of the other portions of this application that illustrate Oracle WebCenter Framework.

2.2.2 Exploring MyPage.jspx in the Fusion Order Demo for WebCenter

Let's take a look at the personal page, MyPage, which showcases the following Oracle WebCenter Framework features:

- Search service
- Recent Activities service
- RSS Viewer service
- Announcements service
- Tags service (tag cloud)
- Standards-Based Java (JSR 168) Portlet portlet

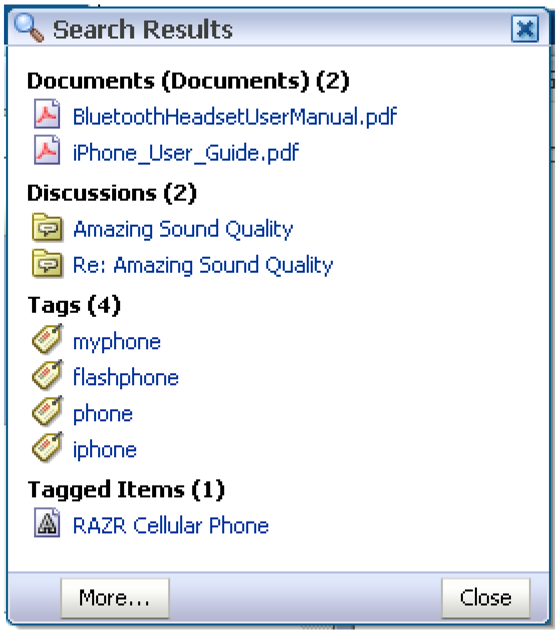
To view these features in the Fusion Order Demo for WebCenter:

1. Before we look at the personal page, take a quick look at the top of the application. You'll notice a search toolbar, which was built using the Search service (Chapter 20, "Integrating the Search Service").

Figure 2–7 Search Toolbar

2. If you enter a keyword, for example `phone`, a list of results displays. The Search service returns all instances of the keyword across the entire application. Figure 2–8 shows an example of possible search results.

Figure 2–8 Search Results



Search Results

Documents (Documents) (2)
BluetoothHeadsetUserManual.pdf
iPhone_User_Guide.pdf

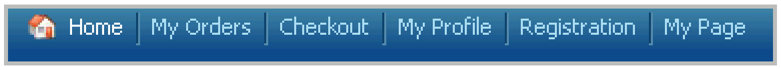
Discussions (2)
Amazing Sound Quality
Re: Amazing Sound Quality

Tags (4)
myphone
flashphone
phone
iphone

Tagged Items (1)
RAZR Cellular Phone

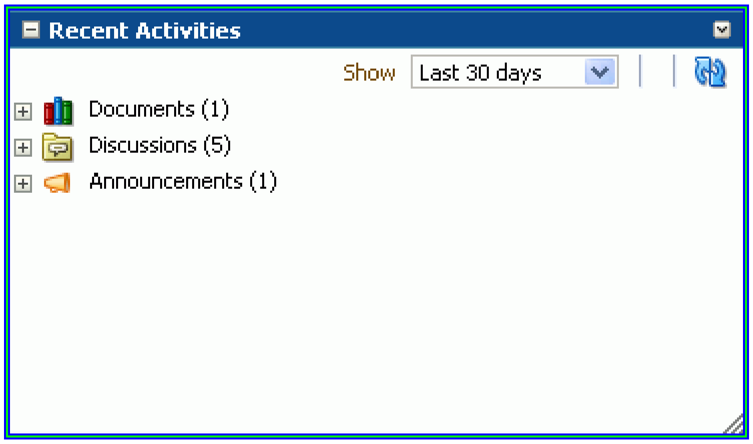
3. To the left of the Search toolbar, click the **My Page** link.

Figure 2–9 My Page Link



4. In the top left corner of My Page, notice the component called **Recent Activities**, which was built using the Recent Activities service (Chapter 18, "Integrating the Recent Activities Service").

Figure 2–10 Recent Activities Component



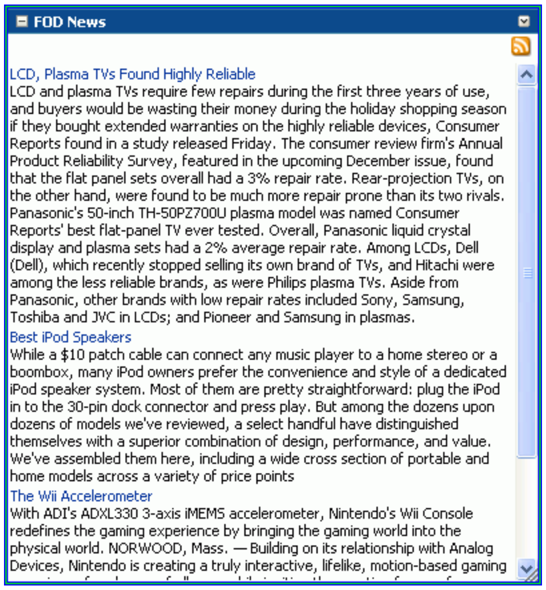
Recent Activities

Show: Last 30 days

Documents (1)
Discussions (5)
Announcements (1)

5. Below the Recent Activities service, you'll see an **FOD News** section, which receives an RSS feed local to the application. This component was built using the RSS Viewer service (Chapter 19, "Integrating the RSS Service").

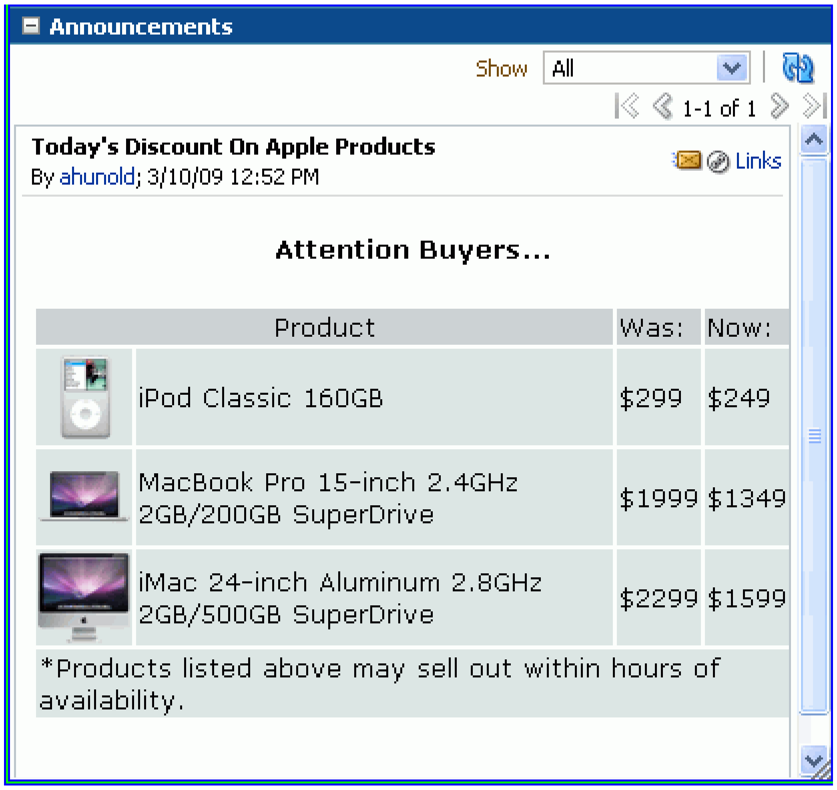
Figure 2–11 RSS Viewer in the Fusion Order Demo for WebCenter


FOD News

LCD, Plasma TVs Found Highly Reliable
LCD and plasma TVs require few repairs during the first three years of use, and buyers would be wasting their money during the holiday shopping season if they bought extended warranties on the highly reliable devices, Consumer Reports found in a study released Friday. The consumer review firm's Annual Product Reliability Survey, featured in the upcoming December issue, found that the flat panel sets overall had a 3% repair rate. Rear-projection TVs, on the other hand, were found to be much more repair prone than its two rivals. Panasonic's 50-inch TH-50PZ700U plasma model was named Consumer Reports' best flat-panel TV ever tested. Overall, Panasonic liquid crystal display and plasma sets had a 2% average repair rate. Among LCDs, Dell (Dell), which recently stopped selling its own brand of TVs, and Hitachi were among the less reliable brands, as were Philips plasma TVs. Aside from Panasonic, other brands with low repair rates included Sony, Samsung, Toshiba and JVC in LCDs; and Pioneer and Samsung in plasmas.
Best iPod Speakers
While a $10 patch cable can connect any music player to a home stereo or a boombox, many iPod owners prefer the convenience and style of a dedicated iPod speaker system. Most of them are pretty straightforward: plug the iPod in to the 30-pin dock connector and press play. But among the dozens upon dozens of models we've reviewed, a select handful have distinguished themselves with a superior combination of design, performance, and value. We've assembled them here, including a wide cross section of portable and home models across a variety of price points
The Wii Accelerometer
With ADI's ADXL330 3-axis iMEMS accelerometer, Nintendo's Wii Console redefines the gaming experience by bringing the gaming world into the physical world. NORWOOD, Mass. — Building on its relationship with Analog Devices, Nintendo is creating a truly interactive, lifelike, motion-based gaming

6. To the right of the Recent Activities component, notice the **Announcements** component, built using the Announcements service. This service connects to the same back-end Discussions server as the Reviews you saw in Section 2.2.1, "Exploring the Home.jspx Page in the Fusion Order Demo for WebCenter."

Figure 2–12 Announcements in the Fusion Order Demo for WebCenter



Today's Discount On Apple Products
By ahunold; 3/10/09 12:52 PM

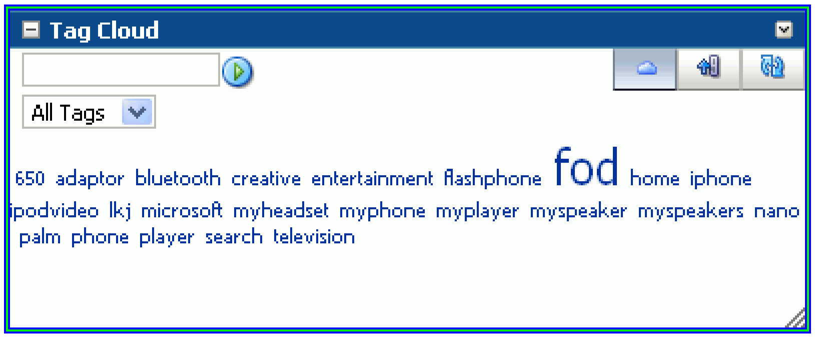
Attention Buyers...

Product		Was:	Now:
	iPod Classic 160GB	$299	$249
	MacBook Pro 15-inch 2.4GHz 2GB/200GB SuperDrive	$1999	$1349
	iMac 24-inch Aluminum 2.8GHz 2GB/500GB SuperDrive	$2299	$1599

*Products listed above may sell out within hours of availability.

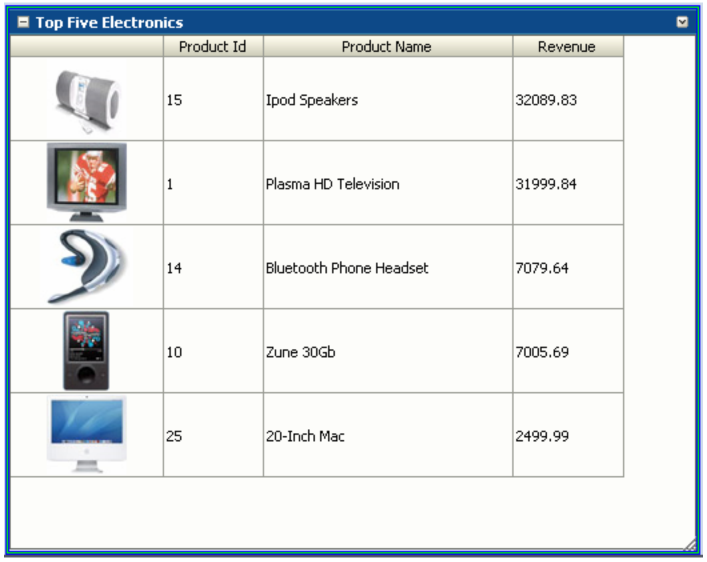
7. Next to the Announcements component is a **Tag Cloud**, which is built using the **Tagging - Tag Cloud** task flow in the Tags service (Chapter 21, "Integrating the Tags Service").

Figure 2–13 Tag Cloud in the Fusion Order Demo for WebCenter


Tag Cloud
All Tags
650 adaptor bluetooth creative entertainment flashphone fod home iphone
ipodvideo lkj microsoft myheadset myphone myplayer myspeaker myspeakers nano
palm phone player search television

8. Below the Tag Cloud is a Standards-Based Java (JSR 168) portlet that is based on an ADF task flow. This ADF task flow was turned into a portlet by using the Oracle JSF Portlet Bridge (Chapter 28, "Creating Portlets with the Oracle JSF Portlet Bridge").

Figure 2–14 Top Five Electronics (Standards-Based Java (JSR 168) Portlet)


Top Five Electronics

	Product Id	Product Name	Revenue
	15	Ipod Speakers	32089.83
	1	Plasma HD Television	31999.84
	14	Bluetooth Phone Headset	7079.64
	10	Zune 30Gb	7005.69
	25	20-Inch Mac	2499.99

Now that we've explored the pages and their components that were built using Oracle WebCenter Framework, let's take a look at some of the other tasks you can do.

2.2.3 Editing MyPage.jspx in the Fusion Order Demo for WebCenter

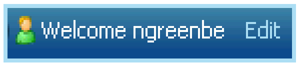
You'll notice that, at the top of the window, there is an **Edit** link. Clicking this link enables you to customize the page. This capability was implemented using Oracle Composer, one of Oracle WebCenter Framework's components. This section describes the following Oracle WebCenter Framework features:

- Oracle Composer
- Oracle Composer extension
- Resource Catalog

To use Oracle Composer in the sample application:

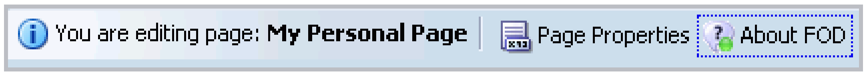
1. Next to the "Welcome *username*" text, click **Edit**.

Figure 2–15 Edit Link



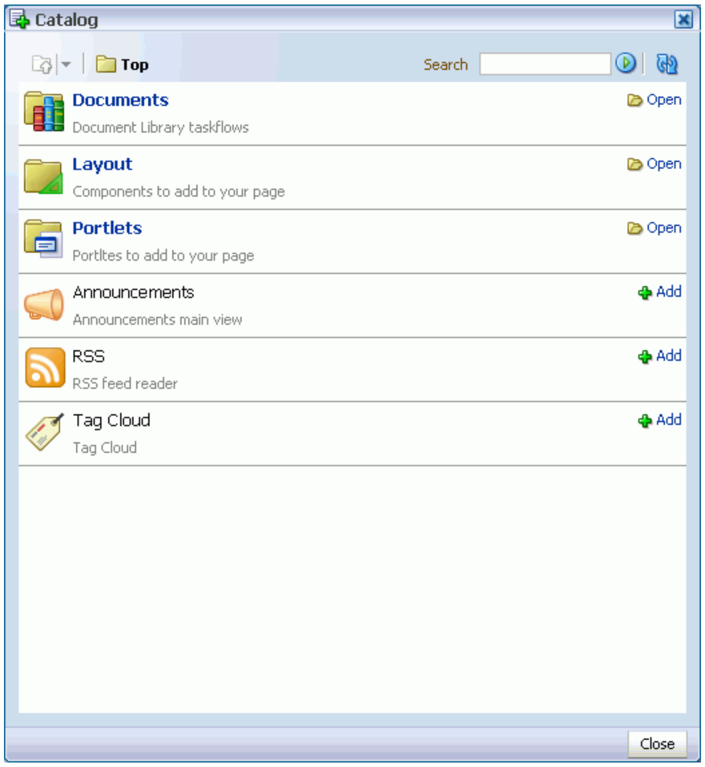
2. At the top of the page, click the **About FOD** link to display an example of how you can use the Oracle Composer extension to add information about the application for your users.

Figure 2–16 About FOD Link in the Fusion Order Demo for WebCenter



3. You can use the Add content link to add new content to your page. When you click **Add content**, a catalog displays. This catalog is an example of a customized Resource Catalog (Chapter 6, "Configuring the Resource Catalog for Oracle Composer").

Figure 2–17 Customized Resource Catalog in the Fusion Order Demo for WebCenter



Now that you've taken the tour of the Fusion Order Demo for WebCenter, you can check out these features in depth and use them yourself by referring to the relevant chapters in this manual. You can find the references to these chapters throughout the tour, or, for a quick reference guide, refer to Table 2–1.

2.3 Oracle WebCenter Framework Components in the Fusion Order Demo for WebCenter

Table 2–1 describes at a high level the Oracle WebCenter features are included in the Fusion Order Demo for WebCenter. To learn more about the different features, refer to the relevant chapter.

Table 2–1 WebCenter Features in the Fusion Order Demo for WebCenter Sample Application

Feature	Location in the Demo	Chapter
Oracle Composer	Users can customize their own personal pages by adding published portlets that display information, such as the "Top Electronic Sellers" and "My Recommendations" portlets. Users can view properties of components at runtime. This runtime customization is enabled by the Oracle Composer. See Section 2.2.3, "Editing MyPage.jspx in the Fusion Order Demo for WebCenter" for the location in the demo.	Chapter 4, "Enabling Runtime Editing of Pages Using Oracle Composer"

Table 2–1 (Cont.) WebCenter Features in the Fusion Order Demo for WebCenter Sample Application

Feature	Location in the Demo	Chapter
Oracle Composer extension	Users can view a custom panel (the "About FOD" panel in the demo), which lets you view details about the demo itself. This is an example of how you can create and register a custom panel in a custom WebCenter application. See Section 2.2.3, "Editing MyPage.jspx in the Fusion Order Demo for WebCenter" for the location in the demo.	Chapter 5, "Extending Runtime Editing Capabilities Using Oracle Composer"
Resource Catalog	You can log in as different users and view customized resource catalogs. See Section 2.2.3, "Editing MyPage.jspx in the Fusion Order Demo for WebCenter" for the location in the demo.	Chapter 6, "Configuring the Resource Catalog for Oracle Composer"
Discussion Forums and Announcements	Users can read the reviews of a particular product, as well as post their own comments using the Discussions service. The application administrator can use the Announcements service to display news about the products, such as discounts or recall information. See Section 2.2.1, "Exploring the Home.jspx Page in the Fusion Order Demo for WebCenter" for the location in the demo.	Chapter 13, "Integrating the Discussions Service"
Documents	When you view information about a particular product in the application, you can access documents such as product manuals in different formats (PDF, Microsoft Word, GIF, JPG, and so on). These documents are stored in a content management system. See Section 2.2.1, "Exploring the Home.jspx Page in the Fusion Order Demo for WebCenter" for the location in the demo.	Chapter 14, "Integrating the Documents Service"
Links	Using the Links service, you can quickly access, as well as create links to and from related product information from the document library, the discussion forums, and announcements. For example, if you are viewing a PDF that mentions an iPod, you can click a link that takes you directly to more information about iPods and their accessories. See Section 2.2.1, "Exploring the Home.jspx Page in the Fusion Order Demo for WebCenter" for the location in the demo.	Chapter 16, "Integrating the Links Service"

Table 2–1 (Cont.) WebCenter Features in the Fusion Order Demo for WebCenter Sample Application

Feature	Location in the Demo	Chapter
Recent Activities	This service tracks recent updates to some of the services in the application: Discussions, Announcements, and Documents. See Section 2.2.2, "Exploring MyPage.jspx in the Fusion Order Demo for WebCenter" for the location in the demo.	Chapter 18, "Integrating the Recent Activities Service"
RSS Viewer	You can view RSS feeds, in this case news about the Fusion Order Demo for WebCenter. See Section 2.2.2, "Exploring MyPage.jspx in the Fusion Order Demo for WebCenter" for the location in the demo.	Chapter 19, "Integrating the RSS Service"
Search	This search feature enables you to search the content from all the services in the Fusion Order Demo for WebCenter, including discussion forums, announcements, the document library, and tagged content. See Section 2.2.2, "Exploring MyPage.jspx in the Fusion Order Demo for WebCenter" for the location in the demo.	Chapter 20, "Integrating the Search Service"
Tags	You can use the Tags service to tag content in the document library. Custom tagging is used to tag a sample product. See Section 2.2.1, "Exploring the Home.jspx Page in the Fusion Order Demo for WebCenter" and Section 2.2.2, "Exploring MyPage.jspx in the Fusion Order Demo for WebCenter" for the locations in the demo.	Chapter 21, "Integrating the Tags Service"
Oracle JSF Portlet Bridge	This portlet is based on an ADF task flow and displays the "Top Five Electronics" in the demo. See Section 2.2.2, "Exploring MyPage.jspx in the Fusion Order Demo for WebCenter" for the location in the demo.	Chapter 28, "Creating Portlets with the Oracle JSF Portlet Bridge"
Standards-Based Java (JSR 168) Portlet	This portlet contains product recommendations in a standards-based JSR 168 portlet, which you can add from a customized Resource Catalog. See Section 2.2.2, "Exploring MyPage.jspx in the Fusion Order Demo for WebCenter" for the location in the demo.	Chapter 29, "Creating Portlets with the Portlet Wizard"

Part II

Using the Oracle WebCenter Framework

Part II contains the following chapters:

3 Preparing Your Development Environment

The WebCenter Extension bundle provides tools for adding, managing, and securing content in a JSP application. Such tools include a declarative security tool; a life cycle management tool; and tools for consuming portlets, WebCenter Web 2.0 services, and content. All of these are designed to enable you to better organize your information around the task at hand. After following a few preparatory procedures, you can use these tools to simplify the creation of WebCenter applications and projects.

This chapter steps you through the process of installing the WebCenter Extension bundle and getting started on WebCenter and portlet applications.

This chapter includes the following sections:

3.1 Installing the WebCenter Extension Bundle

The WebCenter extension bundle is a JDeveloper add-in that provides all WebCenter capabilities to the JDeveloper Studio Edition. To create a WebCenter application, you must first install the WebCenter extension bundle in an instance of Oracle JDeveloper Studio Edition.

To install the WebCenter extension bundle:

1. Start Oracle JDeveloper.
2. If the Select Default Roles dialog opens, select **Default Role** to enable all technologies, and click **OK**.
3. If a dialog box opens asking if you want to migrate settings from an earlier version, click **No**.

4. From the **Help** menu, select **Check for updates**.
5. Click **Next** in the Welcome page of the Check for Updates wizard.
6. On the Source page, under Search Update Centers, search for the WebCenter extension, select it, then click **Finish**.
7. When prompted, restart JDeveloper.

JDeveloper is now configured to create WebCenter and portlet applications.

For more information on obtaining and installing Oracle WebCenter Framework, see the Oracle WebCenter page on OTN at:

`http://webcenter.oracle.com`

3.2 Creating a WebCenter Application

To readily find and use WebCenter components in your application, you must ensure that the right technology scopes are set and tag libraries added. Once you do this, relevant tags are included in the Component Palette and relevant context menus become available in JDeveloper. The easiest way to ensure that scopes are set properly and the right tag libraries are added is to create your application using a template.

Oracle WebCenter provides two templates for creating applications:

- WebCenter Application template—Prepopulates the application with projects optimally scaled for the creation of Java Content Repository (JCR) data controls and consumption of portlets and WebCenter Web 2.0 Services.
- Portlet Producer Application template—Prepopulates the application with a project scoped for creation of JSR 168 (standards-based) and Oracle PDK-Java portlets.

It is not required that you use these templates. If you prefer, you can create your own WebCenter applications and portlet applications by manually scoping the application technologies and creating the relevant projects.

If you are working with an existing application that was created with some other template, you can also extend it with WebCenter components. To do this, you must manually add any required projects, technology scopes, and tag libraries. For information about extending existing applications, see Section 3.6, "Extending Non-WebCenter Applications to Include WebCenter Capabilities."

This section describes how to create an application using the WebCenter Application template.

Note: Technology scopes control the options that appear by default in the New Gallery dialog. They can also be used to control the context menus that appear in JDeveloper and to add tag libraries to the project so that tags appear in the Component Palette.

3.2.1 How to Create a WebCenter Application Using a Template

The easiest way to ensure that the right tag libraries are included and the right technology scopes are set for a new WebCenter application is to create the application using the WebCenter Application template. Application templates provide a means of easily creating an application with a pre-defined set of projects and technology scopes. Templates also provide a means of partitioning an application into projects that reflect a logical separation of the overall work.

A project typically represents a group of related components that can be deployed into their own logical—and possibly physical—tier. You can deploy application projects to the same WebLogic Server instance on a single host or to different WebLogic Server instances, potentially running on different hosts.

The WebCenter Application template consists of a *Model* project for the data model and a *ViewController* project for consuming portlets, components, and data controls. The WebCenter Application template folds both of these projects into one application for simplicity, but you can arrange your applications and projects the way that best fits your circumstances. For example, if you are either the portlet developer or the application developer—but not both—then it is more likely that you would benefit from a separate application for the WebCenter application and for the portlets the application consumes.

To create an application using the WebCenter Application template:

1. Access the application creation wizard in any of the following ways:
   - From the **File** menu, choose **New**. In the New Gallery dialog, expand **Applications**, select **WebCenter Application**, and click **OK**.
   - From the Application Navigator, select **New Application**.
   - If you have an existing application that is open, then in the Application Navigator, right-click ***<Application-Name>*** and choose **New**. In the New Gallery dialog, expand **Applications**, select **WebCenter Application**, and click **OK**.
   - If you have an existing application that is open, then in the Application Navigator, from the ***<Application-Name>*** drop down menu, select **New Application**.
2. In the Name your application page of the creation wizard, enter a name for the application in the **Application Name** field.
3. In the **Directory** field, enter a path to the directory where the application should be stored, or accept the default path.

   For example:

   ```
   C:\JDeveloper\mywork\Application1
   ```

   Optionally, click the **Browse** button to navigate to the desired directory.
4. If required, in the **Application Package Prefix** field, enter a prefix to use for packages created within this application.
5. From the **Application Template** list in the creation wizard, select **WebCenter Application**. Perform this step only if you invoked the wizard by clicking **New Application** in the Application Navigator.
6. Click **OK**.

   The template prepopulates the application with two projects:
   - **Model**, scoped for data modeling and content sourcing
   - **ViewController**, scoped for creation of WebCenter application pages and page elements and for registering portlet producers
7. To create the new WebCenter application with the default settings for the two projects, click **Finish**.

   Alternatively, configure the projects according to your requirements by navigating through the remaining pages of the wizard. Then click **Finish**.

For more information about the projects and technology scope, see Section 3.2.2, "What Happens When You Use the WebCenter Application Template."

3.2.2 What Happens When You Use the WebCenter Application Template

Using the WebCenter Application template generates default projects with unique technology scopes, libraries, and default files. Additionally, it limits the types of options that are exposed in the JDeveloper user interface to those appropriate to the project type.

Note: In addition to populating the application with default files and folders, the template populates the Resource Palette with the WebCenter Services Catalog. The WebCenter Services Catalog contains data controls and task flows that you can use to integrate Oracle WebCenter Services in your application. For more information, see Part III, "Integrating Services".

This section provides an overview of what occurs when you use the WebCenter Application template. It includes the following subsections:

- Section 3.2.2.1, "Template Projects, Technology Scopes, and Libraries"
- Section 3.2.2.2, "WebCenter Application Template Default Files and Folders"

3.2.2.1 Template Projects, Technology Scopes, and Libraries

By default, the WebCenter Application template names its two projects *Model* and *ViewController*. If you prefer, you can rename the projects before or after you create the application. Rename projects before you create the application through the **Manage Templates** command on the **Tools** menu. Rename projects after you create the application through the **Rename** command on the **File** menu.

When you use the WebCenter Application template, JDeveloper does the following for you:

- Creates *Model* and *ViewController* projects, each with their own relevant technology scopes and libraries.

  Table 3–1 lists the default technology scopes and libraries that are included with applications created with the WebCenter Application template. As you start building the application and adding components to the project, the list of libraries grows rapidly.

Table 3–1 WebCenter Application Template Projects

Project	Purpose	Technology Scopes	Libraries
Model	Use this project to define data controls that pull in content from selected data repositories.	ADF Business Components Content Repository Java	None
ViewController	Use this project to build WebCenter application pages that consume portlets and contain your other WebCenter application components. You can also use the ViewController project as the starting point for registering portlet producers. Note, however, that portlet producers are created at the application level irrespective of the starting point from which they were created. For more information, see Section 9.2, "Registering Portlet Producers with a Custom WebCenter Application."	ADF Faces ADF Page Flow HTML Instant Messaging and Presence (IMP) Service Java JSF JSP and Servlets Links Service Oracle Composer Portlet Bridge Service Portlet View Tag Service WebCenter Customizable Components XML	ADF Data Visualization 1.0 ADF Faces Components 11 Customizable Components (HTML) 11 JSF Core 1.2 JSF HTML 1.2 Oracle Composer 1.0 WebCenter Customizable Components Rich 11 WebCenter IM and Presence Tags 11.1.1.1.0 WebCenter Links Service 11.1.1.1.0 WebCenter Tagging 11.1.1.1.0

When you work in each project, the New Gallery is filtered to show only choices that are relevant to the selected technologies. For example, the option to create a Content Repository data control is available in the New Gallery while the *Model* project is selected, but that option is not available while the *ViewController* project is selected. Similarly, options for creating JSP pages are available while the *ViewController* project is selected, but those options are not available while the *Model* project is selected.

- Creates a starter application deployment descriptor `web.xml` file with default settings. You will find the `web.xml` file in the Application Navigator under the ViewController project's `WEB-INF` folder. For more information about `web.xml`, see Section 3.2.2.2.2, "The WebCenter Application Template Default web.xml File."
- Creates an empty `faces-config.xml` file in the ViewController project's `WEB-INF` folder. The `faces-config.xml` file is a JSF configuration file where JSF application resources, such as custom validators and managed beans, are registered and all page-to-page navigation rules are defined. For more information about `faces-config.xml`, see Section 3.2.2.2.3, "The WebCenter Application Template Default faces-config.xml File."
- Adds `jsf-impl.jar` in the ViewController project's `WEB-INF\lib` folder. This JAR file is the JSF reference implementation that JDeveloper includes by default.
- Creates an empty `adfc-config.xml` file. This file can be used to set up routers to conditionally control flow between pages.

  See *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework* for more information about ADF task flows.

3.2.2.2 WebCenter Application Template Default Files and Folders

When you use the WebCenter Application template to create an application, it creates a folder hierarchy with an accompanying set of default files. This section provides detailed information about default files and the file hierarchy. It contains the following subsections:

- Section 3.2.2.2.1, "The WebCenter Application Default Folder Hierarchy"
- Section 3.2.2.2.2, "The WebCenter Application Template Default web.xml File"
- Section 3.2.2.2.3, "The WebCenter Application Template Default faces-config.xml File"
- Section 3.2.2.2.4, "The WebCenter Application Template Default trinidad-config.xml File"
- Section 3.2.2.2.5, "The WebCenter Application Template Default adfc-config.xml File"

3.2.2.2.1 The WebCenter Application Default Folder Hierarchy Because there are several possible views of the Oracle JDeveloper Application Navigator, the hierarchy of folders in a WebCenter application may differ when you compare folders in the Application Navigator with the same folders in your file system. For example, Figure 3–1 illustrates one view of a newly created application in the Oracle JDeveloper Application Navigator.

Figure 3–1 Application from WebCenter Application Template in JDeveloper

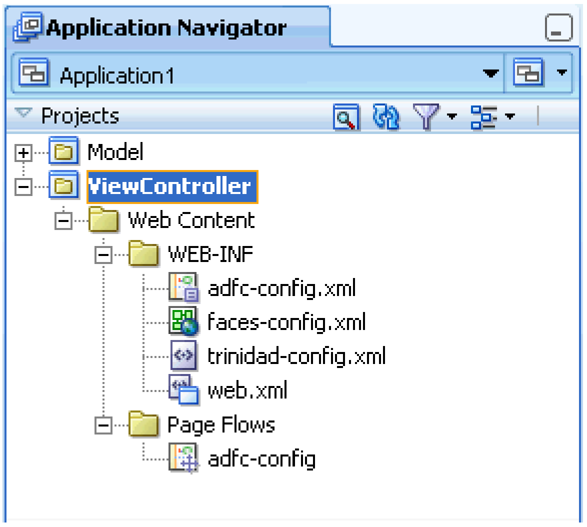
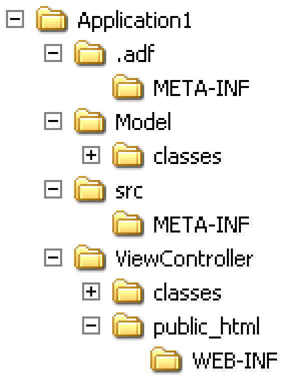


Figure 3–2 illustrates the folder hierarchy of the same application in a Windows file system.

Figure 3–2 Application from WebCenter Application Template in a Windows File System



If you prefer to view the Windows file system hierarchy in the Application Navigator, click the **Navigator Display Options** icon in the Projects panel and select **Group by Directory**.

3.2.2.2.2 The WebCenter Application Template Default web.xml File Part of WebCenter application configuration is determined by the content of its J2EE application deployment descriptor file: `web.xml`. The `web.xml` file defines many application settings that are required by the application server.

> **Note:** Rather than being specified in `web.xml`, the context root path is assigned when the application is deployed.

Typical runtime settings include initialization parameters, custom tag library location, and security settings. Depending on the technology you use, other settings may be added to `web.xml` as appropriate.

> **Note:** For standard J2EE files like `web.xml`, there is usually a counterpart Oracle-specific file with additional Oracle-specific options, for example, `weblogic.xml`.

Example 3–1 illustrates the default `web.xml` file provided through the WebCenter Application template. This file is located in the `/public_html/WEB-INF` directory relative to the ADF application's ViewController project.

Example 3–1 Default web.xml File Provided Through the WebCenter Application Template

```
<?xml version = '1.0' encoding = 'windows-1252'?>
@ <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
@ xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
@ http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5"
@ xmlns="http://java.sun.com/xml/ns/javaee">
  <context-param>
    <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
    <param-value>client</param-value>
  </context-param>
  <context-param>
    <description>If this parameter is true, there will be an automatic check
of the modification date of your JSPs, and saved state will be discarded when
JSP's change. It will also automatically check if your skinning css files
```

```
have changed without you having to restart the server. This makes development
easier, but adds overhead. For this reason this parameter should be set to
false when your application is deployed.</description>
<param-name>org.apache.myfaces.trinidad.CHECK_FILE_MODIFICATION</param-name>
    <param-value>false</param-value>
  </context-param>
  <context-param>
    <description>Whether the 'Generated by...' comment at the bottom of ADF
Faces HTML pages should contain version number information.</description>
    <param-name>oracle.adf.view.rich.versionString.HIDDEN</param-name>
    <param-value>false</param-value>
  </context-param>
  <filter>
    <filter-name>trinidad</filter-name>
<filter-class>org.apache.myfaces.trinidad.webapp.TrinidadFilter</filter-class>
  </filter>
  <filter>
    <filter-name>ServletADFFilter</filter-name>
    <filter-class>oracle.adf.share.http.ServletADFFilter</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>trinidad</filter-name>
    <servlet-name>Faces Servlet</servlet-name>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>REQUEST</dispatcher>
  </filter-mapping>
  <filter-mapping>
    <filter-name>ServletADFFilter</filter-name>
    <servlet-name>Faces Servlet</servlet-name>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>REQUEST</dispatcher>
  </filter-mapping>
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet>
    <servlet-name>resources</servlet-name>

<servlet-class>org.apache.myfaces.trinidad.webapp.ResourceServlet</servlet-cla
ss>
  </servlet>
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>/faces/*</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>resources</servlet-name>
    <url-pattern>/adf/*</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>resources</servlet-name>
    <url-pattern>/afr/*</url-pattern>
  </servlet-mapping>
</web-app>
```

When you first create your WebCenter application, configuration settings for JSF servlet and mapping and for resource servlet and mapping are automatically added to the starter `web.xml` file.

The `Faces Servlet` entry inside `<servlet></servlet>` tags (Example 3–1) provides information about the JSF servlet, `javax.faces.webapp.FacesServlet`. This servlet manages the request processing life cycle for Web applications that use JSF to construct the user interface. The configuration setting maps the JSF servlet to a symbolic name, *Faces Servlet*.

The `resources` entry inside `<servlet></servlet>` tags (Example 3–1) provides information about the ADF resource servlet used to serve up web application resources (images, style sheets, JavaScript libraries) by delegating to a `ResourceLoader`.

The `<servlet-mapping></servlet-mapping>` tags map the URL pattern to a servlet's symbolic name. You can use either a path prefix or an extension suffix pattern.

By default, JDeveloper uses the path prefix `/faces/*`. That is, when a URL, for example, `http://localhost:8080/SRDemo/faces/index.jsp`, is issued, the URL activates the JSF servlet, which strips off the `faces` prefix and loads the file `/SRDemo/index.jsp`.

Note: If you prefer to use the extension `jsf` for web pages instead of `jsp` or `jspx`, then you must set the `javax.faces.DEFAULT_SUFFIX` context parameter to `jsf`, for example:

```
<context-param>
   <param-name>javax.faces.DEFAULT_SUFFIX</param-name>
   <param-value>.jsf</param-value>
</context-param>
```

Then add a servlet mapping in `web.xml` that invokes the JSP servlet for files with the extension `jsf`.

To edit `web.xml` in Oracle JDeveloper, right-click `web.xml` in the Application Navigator and choose **Open** from the context menu. This opens `web.xml` in the Web Application Deployment Descriptor editor, in Overview mode. If you are familiar with configuration element names, then you can also use the XML editor to modify `web.xml`.

For information about the configuration elements you can use in the `web.xml` file, see "Oracle ADF XML Files" in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework.*

3.2.2.2.3 The WebCenter Application Template Default faces-config.xml File Use the `faces-config.xml` file to register a WebCenter application's resources, such as custom validators and managed beans, and to define all page-to-page navigation rules. The default name this file is `faces-config.xml`, though this naming convention is not required.

Depending on how resources are packaged, an application can have one or multiple `faces-config.xml` files. For example, you can create individual JSF configuration files for:

- Different areas of your application
- Each library containing custom components or renderers

For more information about creating multiple `faces-config.xml` files, see *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework.*

Example 3–2 illustrates the default `faces-config.xml` file provided through the WebCenter Application template. This file is located in the `/public_html/WEB-INF` directory relative to the ADF application's ViewController project.

Example 3–2 Default faces-config.xml File Provided Through the WebCenter Application Template

```
<?xml version="1.0" encoding="windows-1252"?>
<faces-config version="1.2" xmlns="http://java.sun.com/xml/ns/javaee">
  <application>
    <default-render-kit-id>oracle.adf.rich</default-render-kit-id>
  </application>
  <lifecycle>

<phase-listener>oracle.adf.controller.faces.lifecycle.ADFPhaseListener</phase-list
ener>
  </lifecycle>
</faces-config>
```

To edit the `faces-config.xml` file, double-click it in the Application Navigator. By default, the file is opened in the Editor window in *Diagram* mode, as indicated by the active **Diagram** tab at the bottom of the Editor window. When creating or modifying JSF navigation rules, Diagram mode enables you to visually create and manage page flows. For more information, see *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework.*

To create or modify configuration elements other than navigation rules, use the Editor's Overview mode. Enter this mode by clicking the **Overview** tab at the bottom of the Editor window.

JSF allows multiple `<application>` elements in a single `faces-config.xml` file. When you use the JSF Configuration Editor, you can edit only the first instance. For any other `<application>` elements, you must edit the file directly using the XML editor. To use the XML editor, open the `faces-config.xml` file and go to the **Source** tab in the Editor window.

For reference information about the configuration elements you can use in the `faces-config.xml` file, see "ADF Faces Configuration" in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework.*

3.2.2.2.4 The WebCenter Application Template Default trinidad-config.xml File The default `trinidad-config.xml` file, available when you create an application, contains information about the application skin family. Additionally, it can contain information about the level of page accessibility support, page animation, time zone, enhanced debugging output, and the URL for Oracle Help for the Web (OHW). Like `faces-config.xml`, the `trinidad-config.xml` file has a simple XML structure that enables you to define element properties using JSF Expression Language (EL) or static values.

Example 3–3 illustrates the default `trinidad-config.xml` file provided through the WebCenter Application template. This file is located in the `/public_html/WEB-INF` directory relative to the ADF application's ViewController project.

Example 3–3 Default trinidad-config.xml File Provided Through the WebCenter Application Template

```
<?xml version="1.0" encoding="windows-1252"?>
<trinidad-config xmlns="http://myfaces.apache.org/trinidad/config">
  <skin-family>blafplus-rich</skin-family>
</trinidad-config>
```

In addition to the skin family, you can define the following application values in the `trinidad-config.xml` file:

- Page animation
- Level of page accessibility support
- Time zone
- Enhanced debugging output
- Oracle Help for the Web (OHW) URL

For reference information about the `trinidad-config.xml` file, see "ADF Faces Configuration" in the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*.

3.2.2.2.5 The WebCenter Application Template Default adfc-config.xml File The `adfc-config.xml` file is the configuration file for an ADF unbounded task flow. This file contains metadata about the activities and control flows contained in the unbounded task flow. The default name for this file is `adfc-config.xml`, though this naming convention is not required.

If **ADF Page Flow** is specified as a Selected Technology on the Technology Scope page of the Project Properties dialog, a new `adfc-config.xml` source file is automatically created within the project. The `adfc-config.xml` file is the main source file for an unbounded task flow.

The `adfc-config.xml` file is located in the `/public_html/WEB-INF` directory relative to the ADF application's ViewController project.

Note: If you do not plan to use task flows in your application, you can delete the `adfc-config.xml` file.

For more information about task flows and the `adfc-config.xml` file, see "Oracle ADF XML Files" in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework.*

3.3 Creating WebCenter Application-Enabled Pages

Create WebCenter application pages in JDeveloper as JSP *documents* (`.jspx`) rather than JSP *pages* (`.jsp`). For page customizations to be stored, the page must be represented in XML. By choosing to create an XML representation of the JSP document (`.jspx`), you ensure that customizations are always possible for the WebCenter application page.

If you are not planning to enable customization, you can create a JSP page; however this choice might cause problems if you later decide that you want to enable customization.

This section explains how to create WebCenter application-enabled pages. It contains the following subsections:

- Section 3.3.1, "How to Create a Page in a WebCenter Application Based on a Template"
- Section 3.3.2, "How to Create a Page in a Manually-Created WebCenter Application"

Note: For detailed information about creating pages in JDeveloper with Oracle ADF, see the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*. It contains all the information you need about Oracle ADF Faces and building basic and complex pages.

3.3.1 How to Create a Page in a WebCenter Application Based on a Template

When you create a WebCenter application using the WebCenter Application template, all the libraries necessary for adding WebCenter Web 2.0 Services and Composer components to your page are available by default. Additionally, runtime page customizations are enabled by default.

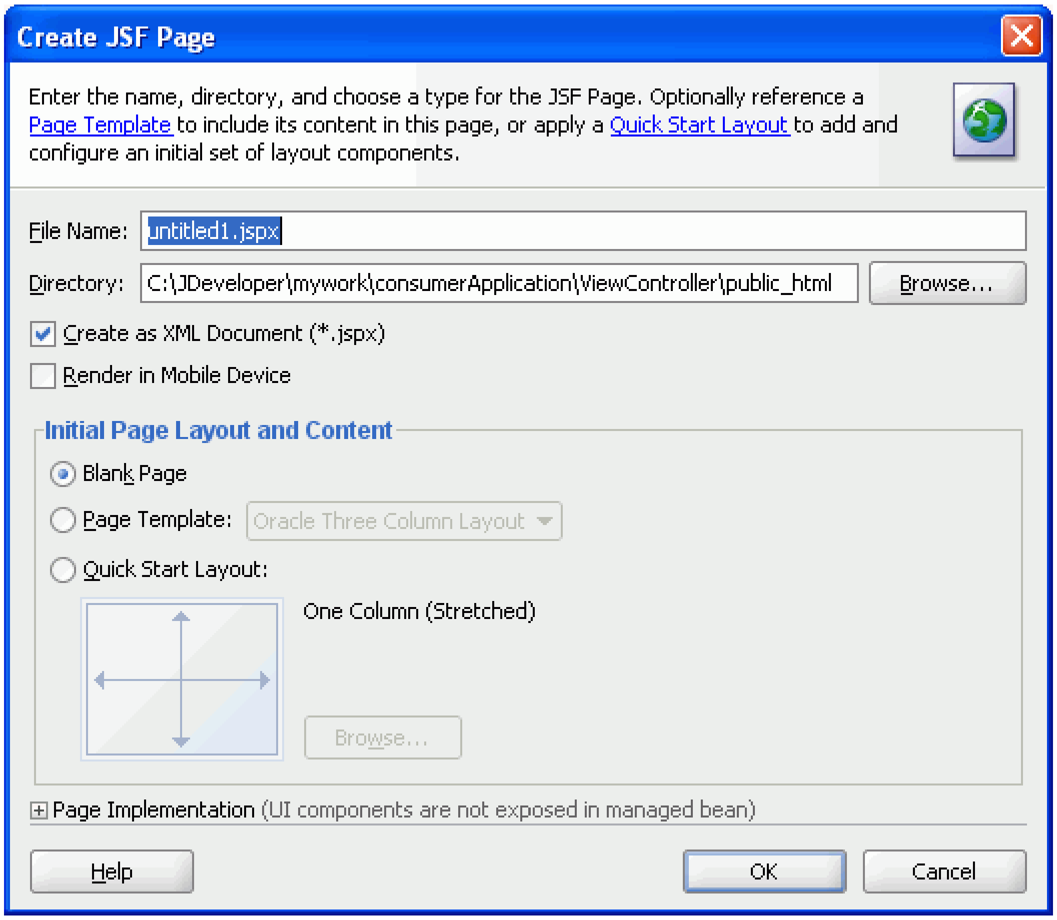
To create a customizable WebCenter application-enabled JSPX page:

1. Start JDeveloper.
2. In the Application Navigator, go to the ViewController project of the application in which you want to create a customizable page, right click the ViewController project, and choose **New**.

   Note: For information about how to create a WebCenter application if you have not yet created one, see Section 3.2, "Creating a WebCenter Application."

3. In the New Gallery, expand **Web Tier,** select **JSF** and then **JSF Page,** and click **OK.**
4. In the Create JSF Page dialog (Figure 3–3), enter a file name for the page.

Figure 3–3 The Create JSF Page Dialog


Create JSF Page

Enter the name, directory, and choose a type for the JSF Page. Optionally reference a Page Template to include its content in this page, or apply a Quick Start Layout to add and configure an initial set of layout components.

File Name: untitled1.jspx

Directory: C:\JDeveloper\mywork\consumerApplication\ViewController\public_html

[✓] Create as XML Document (*.jspx)

[ ] Render in Mobile Device

Initial Page Layout and Content

(●) Blank Page

( ) Page Template: Oracle Three Column Layout

( ) Quick Start Layout: One Column (Stretched)

Page Implementation (UI components are not exposed in managed bean)

5. Select **Create as XML Document (*.jspx)** and click **OK.**

For information about the other settings in this dialog that are not specific to WebCenter-enabled pages, see the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*.

3.3.2 How to Create a Page in a Manually-Created WebCenter Application

To create a JSPX page in a manually-created WebCenter application or non-WebCenter application, you must first add the necessary libraries to the page's home project. Adding the required libraries ensures that Web 2.0 Services and Oracle Composer component tag libraries are available in the Component Palette. You can then drag and drop components from these libraries onto your page.

Add the libraries specified in this section to enable page customization in the following types of applications:

- A WebCenter application created manually rather than through a template. For more information, see Section 3.6, "Extending Non-WebCenter Applications to Include WebCenter Capabilities."
- A non-WebCenter application that was created using no template or any template other than the WebCenter Application template.

To add the necessary libraries:

1. Right-click your project and choose **Project Properties** from the context menu.

2. In the left panel of the Project Properties dialog, click **JSP Tag Libraries**.
3. Click **Add**.
4. Add the following libraries, and click **OK**:
   - Oracle Composer 1.0
   - WebCenter Customizable Components Rich 11
   - Customizable Components (HTML) 11
   - WebCenter IM and Presence Tags 11.1.1.1.0
   - WebCenter Links Service 11.1.1.1.0
   - WebCenter Tagging 11.1.1.1.0

You can now create a WebCenter Application-enabled JSPX page by performing the steps in Section 3.3, "Creating WebCenter Application-Enabled Pages."

3.4 Creating a Portlet Application

Use portlets to provide controlled display of and access to local and remote content sources. You can create portlets as part of your application development effort when you create an application and project scoped for portlets. With the appropriate scoping in place, Oracle JDeveloper provides options that are tailored to creating portlets. For example, in an application scoped for portlets the Web Tier node appears in the New Gallery, and portlet creation is allowed.

Note: Portlet applications are not automatically scoped for portlet producer registration. For information about registering a portlet producer, see Section 9.2, "Registering Portlet Producers with a Custom WebCenter Application."

You can scope an application and its projects for portlet creation in either of the following ways:

- Use the Portlet Producer Application template
- Extend an application by optimizing one of its projects for portlet creation

Notes: Use separate applications to build and consume portlets. Building and consuming portlets within the same application can cause problems during packaging and deployment.

This section describes the steps to create a portlet application using the Portlet Producer Application template. It contains the following subsections:

- Section 3.4.1, "How to Create a Portlet Application Using a Template"
- Section 3.4.2, "What Happens When You Use a Portlet Producer Application Template"

For information about extending an existing application to enable portlet creation, see Section 3.6, "Extending Non-WebCenter Applications to Include WebCenter Capabilities."

3.4.1 How to Create a Portlet Application Using a Template

The Portlet Producer Application template consists of a single project, *Portlets*. Use the Portlets project to create JSR 168 (standards-based) and Oracle PDK-Java portlets.

To create an application using the Portlet Producer Application template:

1. Access the application creation wizard in any of the following ways:
   - From the **File** menu, choose **New**. In the New Gallery dialog, expand **Applications**, select **Portlet Producer Application**, and click **OK**.
   - From the Application Navigator, select **New Application**.
   - If you have an application open, then in the Application Navigator open the application name drop down menu and select **New Application**.
   - In the Application Navigator, right-click the application name and choose **New**. In the New Gallery dialog, expand **Applications**, select **Portlet Producer Application**, and click **OK**.
2. In the Name your application page of the creation wizard, enter a name for the application in the **Application Name** field.
3. In the **Directory** field, accept the default path or enter a path to the directory where the application should be stored.

   For example:

   ```
   C:\JDeveloper\mywork\Application1
   ```

   Optionally, click the **Browse** button to navigate to the desired directory.
4. If required, in the **Application Package Prefix** field, enter a prefix to use for packages to be created within this application.
5. From the **Application Template** list, select **Portlet Producer Application**.

   Note: The Application Template list appears only when you invoke the wizard by clicking **New Application** in the Application Navigator as described in step 1.

6. Click **Finish** to create the portlet application with default project configurations.

3.4.2 What Happens When You Use a Portlet Producer Application Template

Using the Portlet Producer Application template generates the default *Portlets* project with the Java, JSP and Servlets, and Portlet technology scopes. It limits the types of options that are exposed in the JDeveloper user interface to those appropriate for creating portlets.

When you right-click this new project, options to create JSR 168 and Oracle PDK-Java portlets are provided.

3.5 Implementing Security in Your Application

The use of Oracle ADF security enables WebCenter applications to easily adjust to real-world business security requirements because, rather than securing paths, you secure actions with JAAS. JAAS-based Oracle ADF Security provides:

- More granular security. Because J2EE security is URL-based or page-based, it is not possible to have a finer level of access control. With Oracle ADF security, different roles can perform different levels of activity at the same URL.
- The ability to create new roles and their associated access privileges at runtime. This capability flows from the fact that policies storage is external to the application.
- Simplified permission assignment through hierarchical roles, which enable the inheritance of permissions. While the roles used by J2EE security constraints are flat, JAAS permissions reference enterprise roles, which can be nested.

Note: For details on how to implement a complete security solution for your custom application, see the section on *enforcing ADF Security in a Fusion Web application* in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

Some services must know the identity of the user. For example, for saving searches the Search service requires the user's identity. For these services, you must at least configure your application to authenticate users so that they have distinct identities for personalization and preferences. Such a security configuration is useful in testing how user privileges impact runtime customization capabilities.

This section covers the basic security configuration required to help in testing your application in Integrated WebLogic Server (WLS). It contains the following subsections:

- Section 3.5.1, "How to Configure ADF Security"
- Section 3.5.2, "How to Create an Application Role"
- Section 3.5.3, "How to Add Login and Logout Links"

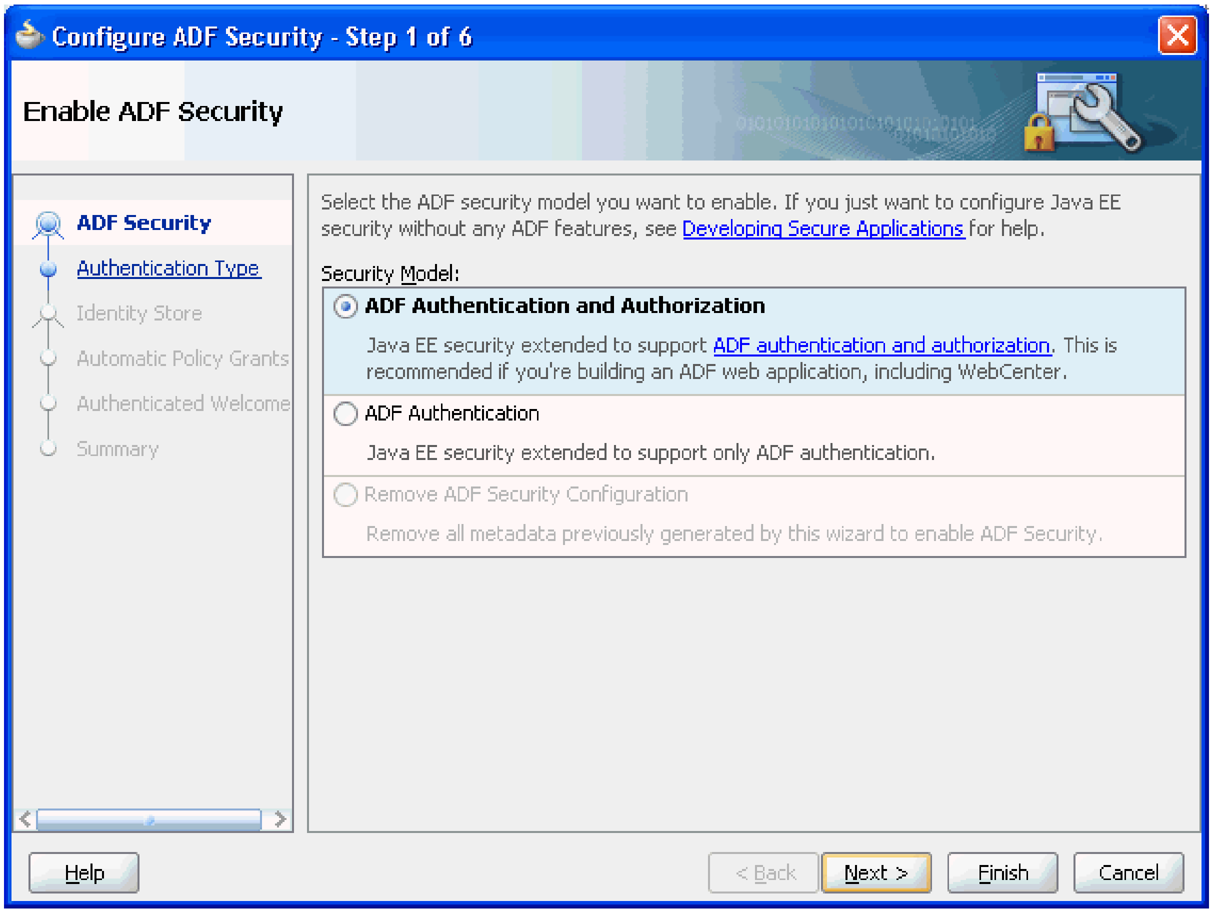
3.5.1 How to Configure ADF Security

This section provides information about implementing a basic security solution.

To enforce Oracle ADF Security for the application:

1. Open your application, and select the **ViewController** project.
2. Access the Configure ADF Security Wizard using one of the following methods:
   - From the **Application** menu, select **Secure** and then **Configure ADF Security**.
   - Right-click the application name, select **Secure** and then **Configure ADF Security**.
3. On the Enable ADF Security page, select **ADF Authentication and Authorization**, which is the typical choice for WebCenter applications.

Figure 3–4 ADF Security Page of the ADF Security Wizard



Configure ADF Security - Step 1 of 6

Enable ADF Security

Select the ADF security model you want to enable. If you just want to configure Java EE security without any ADF features, see Developing Secure Applications for help.

Security Model:

(●) **ADF Authentication and Authorization**

Java EE security extended to support ADF authentication and authorization. This is recommended if you're building an ADF web application, including WebCenter.

( ) ADF Authentication

Java EE security extended to support only ADF authentication.

( ) Remove ADF Security Configuration

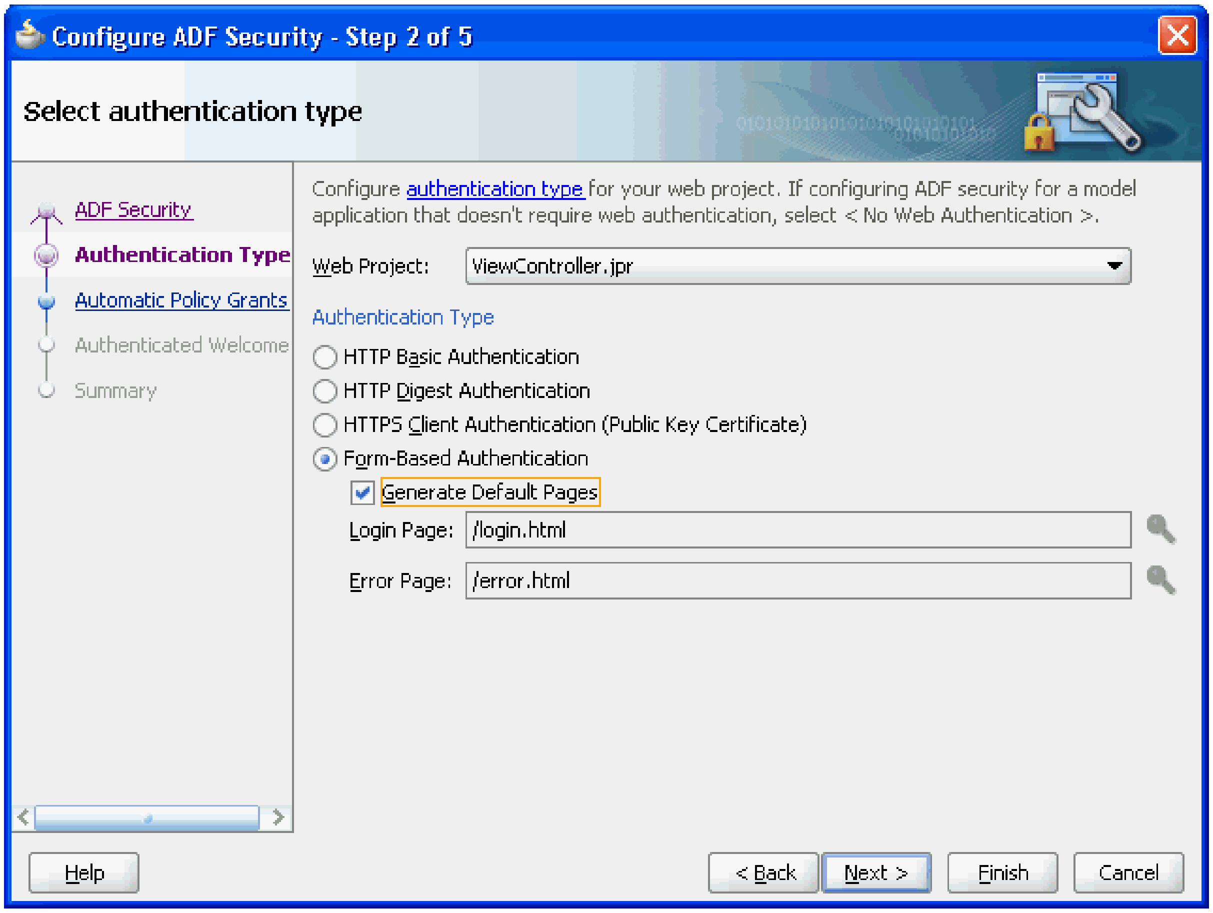
Remove all metadata previously generated by this wizard to enable ADF Security.

4. Click **Next**.
5. Select the authentication type you want your application to use when users submit their login information.

   To make the process of securing your application easy, select **Form-Based Authentication** and then the **Generate Default Pages** option, as shown in Figure 3–5.

   Alternatively, if you do not select Generate Default Pages, you can create custom login and error pages. For information about creating custom login pages, see Chapter 24, "Securing Your WebCenter Application."

Figure 3–5 Authentication Page of ADF Security Wizard


Configure ADF Security - Step 2 of 5

Select authentication type

Configure authentication type for your web project. If configuring ADF security for a model application that doesn't require web authentication, select < No Web Authentication >.

Web Project: ViewController.jpr

Authentication Type

- HTTP Basic Authentication
- HTTP Digest Authentication
- HTTPS Client Authentication (Public Key Certificate)
- [✓ Form-Based Authentication]

[✓] Generate Default Pages

Login Page: /login.html

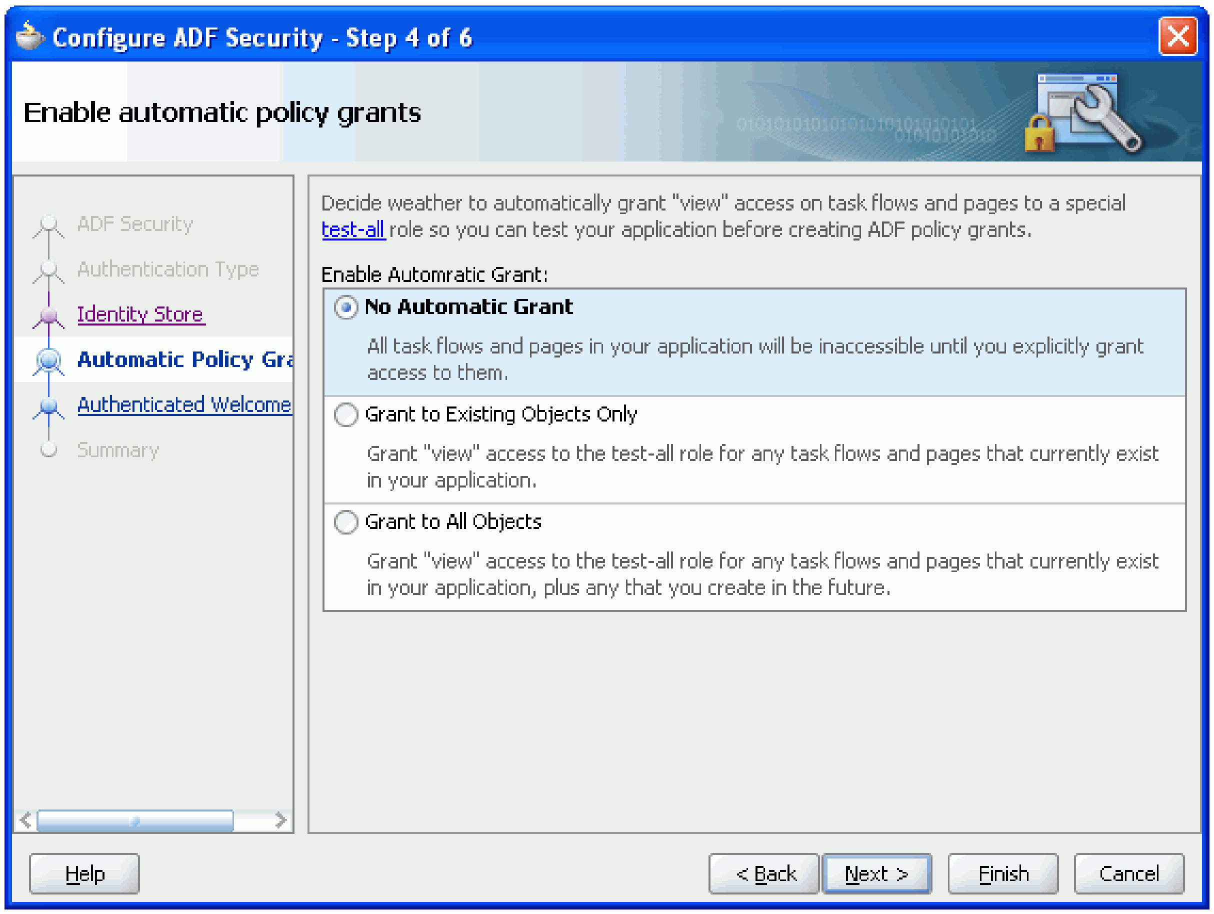
Error Page: /error.html

6. Click **Next**.

7. On the Enable automatic policy grants page, shown in Figure 3–6, select whether to make ADF resources public without requiring application developers to first define ADF policy grants. For a basic security implementation, you can select the **No Automatic Grant** option.

   However, in this case, you must individually grant permissions on each page in the application.

Figure 3–6 Automatic Policy Grants Page



Decide weather to automatically grant "view" access on task flows and pages to a special test-all role so you can test your application before creating ADF policy grants.

Enable Automratic Grant:

- [✓ **No Automatic Grant**]
  All task flows and pages in your application will be inaccessible until you explicitly grant access to them.
- Grant to Existing Objects Only
  Grant "view" access to the test-all role for any task flows and pages that currently exist in your application.
- Grant to All Objects
  Grant "view" access to the test-all role for any task flows and pages that currently exist in your application, plus any that you create in the future.

When you enable automatic policy grants, a *View* grant is made to all members of the `test-all` application role. This option provides a convenient way to run and test application resources without the restricted access that ADF authorization enforces. You can also disable automatic policy grants to secure all ADF resources by default. For more information about the specific grants provided through task flows, see Section 11.1.3, "Automated Task Flow Grants."

Select **No Automatic Grant** when you want to explicitly configure a policy store to grant access to ADF resources. No automatic grants to the `test-all` application role are granted. However when WebCenter Services task flows are consumed in an application, the WebCenter extension automatically adds appropriate grants for these task flows, giving View access to the role `authenticated-role`.

Note: Grants made to the role `authenticated-role` are visible in the source view of the `jazn-data.xml` file. If the default grant to `authenticated-role` is not appropriate to the application, you can change it by manually updating the source view of the application's `jazn-data.xml`.

Select **Grant to Existing Objects Only** when you want JDeveloper to grant View privileges to the `test-all` application role and you want this policy to apply to all your application's existing ADF task flows and web pages in the user interface project.

Select **Grant to All Objects** when you want JDeveloper to grant View privileges to the `test-all` application role and you want this policy to apply to all ADF task flows and web pages that developers create in the user interface project. Note that the **Grant to All Objects** option appears when you run the wizard for the first time. Subsequently, the **Grant to New Objects** option appears each time you run the wizard.

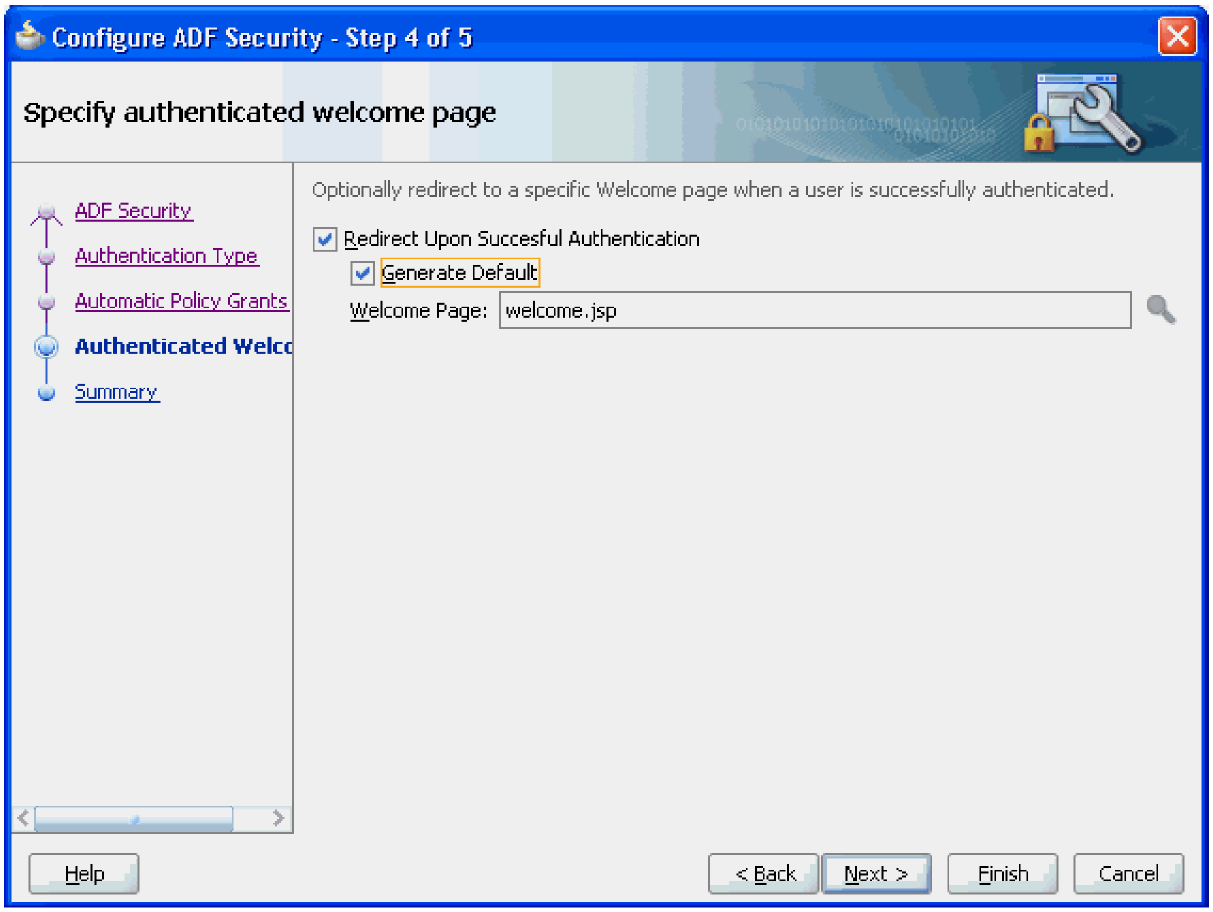
8. Click **Next**.

9. Optionally, if you want to display a specific welcome page upon authentication, then select the **Redirect Upon Successful Authentication** option as shown in Figure 3–7.

   You can either create a custom welcome page or let the wizard create a default one. To keep the process simple, select **Generate Default**.

   For information about creating a custom welcome page, see Chapter 24, "Securing Your WebCenter Application."

Figure 3–7 Specify an Authenticated Welcome Page



Optionally redirect to a specific Welcome page when a user is successfully authenticated.

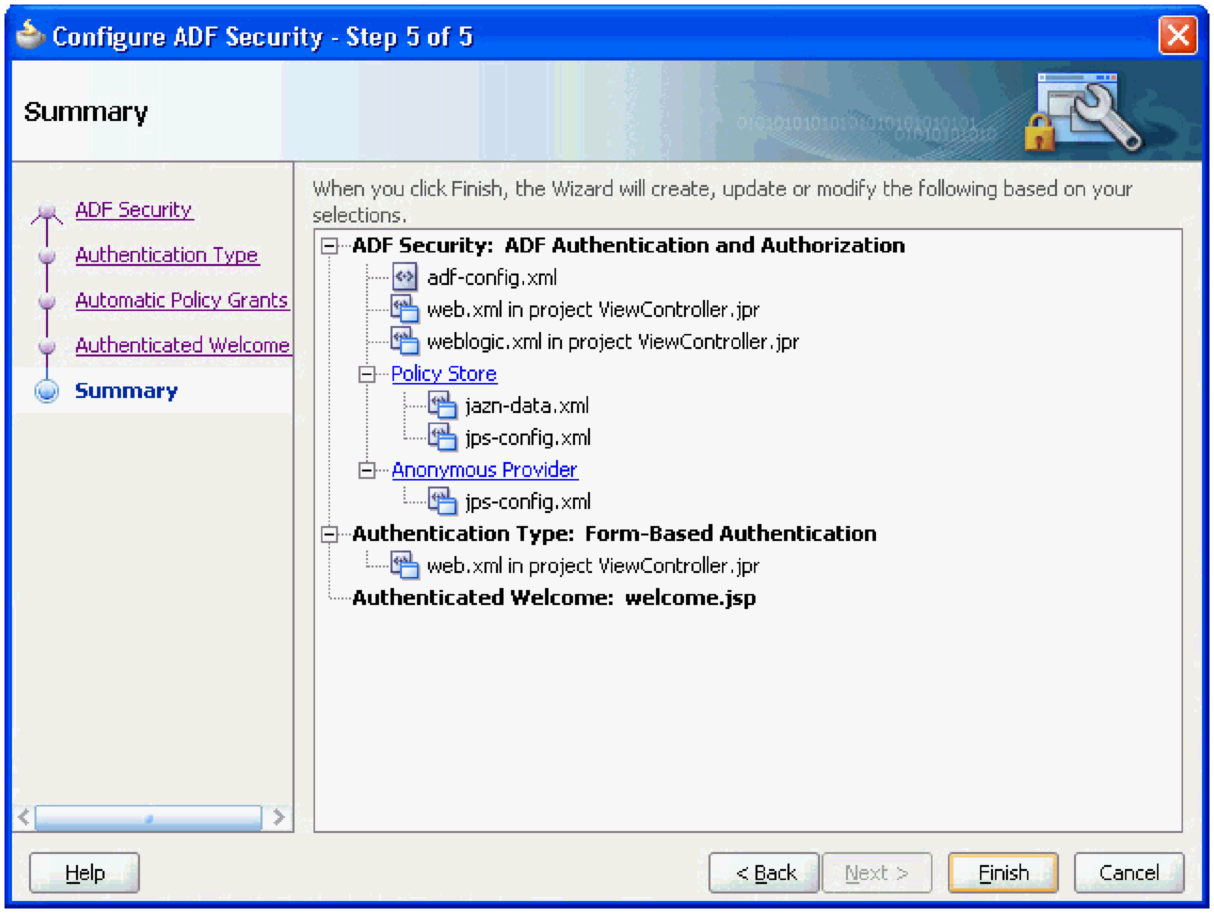
☑ Redirect Upon Succesful Authentication

☑ Generate Default

Welcome Page: welcome.jsp

10. The Summary page opens, as shown in Figure 3–8.

Figure 3–8 Summary Page



When you click Finish, the Wizard will create, update or modify the following based on your selections.

```
ADF Security:  ADF Authentication and Authorization
    adf-config.xml
    web.xml in project ViewController.jpr
    weblogic.xml in project ViewController.jpr
    Policy Store
        jazn-data.xml
        jps-config.xml
    Anonymous Provider
        jps-config.xml
Authentication Type:  Form-Based Authentication
    web.xml in project ViewController.jpr
Authenticated Welcome:  welcome.jsp
```

11. Click **Finish**.

    It may take several moments for the wizard to complete.

12. On the Security Infrastructure Created dialog, click **OK**.

3.5.2 How to Create an Application Role

You can test your ADF Security settings by creating a role and a user, and giving the user privileges on your JSF page. Then, as you add services, you can use this user's credentials to test the service at runtime.

To create an application role:

1. In the Application Resources panel, navigate to the `jazn-data.xml` file, which is available under the META-INF folder.
2. Access the Edit JPS Identity and Policy Store dialog using any of the following methods:
   - Right-click **jazn-data.xml** and select **Properties** from the context menu.
   - Right-click **jazn-data.xml** and select **Open** from the context menu. Click **Application Roles** on the page.
   - Right-click the application name, select **Secure** and then **Users and Groups**.
3. In the Edit JPS Identity and Policy Store dialog, click **Roles** under the jazn.com realm and then click **Add**.

4. In the Add Role dialog, enter a name for the role, for example, `admin`.
5. Click **OK**, then click **OK** again.

   You have now created a sample role for testing purposes.
6. Save the file.

Similarly, you can add users to the application by specifying the user name and password. For example, if you have a user named `fmwadmin` for your Discussions server, you may want to use the same user ID, so that this user can authenticate with the Discussions server. You can then associate this user with the `admin` role you created.

3.5.3 How to Add Login and Logout Links

You can add login and logout links to your public welcome page so that users can explicitly log in and out while they are in the application. For information about adding login and logout links, see the chapter titled "Enforcing ADF Security in a Fusion Web application" in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

3.5.4 What Happens at Runtime

When you run the application, a login page is rendered. Subsequently, a welcome page is rendered for successfully-authenticated users, and an error message is rendered for users without the required privileges.

3.6 Extending Non-WebCenter Applications to Include WebCenter Capabilities

If your business requirement was to create a non-WebCenter application, say a Fusion Web application, but you want to also include WebCenter capabilities in that application, you can do so by adding the required technology scopes and libraries. The technology scopes limit UI options to those appropriate to the type of application project you are building. The libraries provide the components and elements useful in constructing application content. You can extend an existing application to integrate content from various repositories, create portlets, or consume portlets and WebCenter Web 2.0 Services. You can extend ADF applications from the current release or from a 10.1.x.x release.

The easiest way to understand what to specify for a WebCenter application or portlet application is to review the out-of-the-box template's characteristics. You can then mimic these in your own applications or templates.

While extending your application, you may decide to create projects that are optimized for specific WebCenter capabilities. This section steps you through the process of manually creating projects like those provided with the WebCenter templates and adding the required technology scopes and libraries to projects.

To enable WebCenter capabilities in an application:

1. Open your application.

   Click **Yes** if you are prompted to migrate your application settings.

Note: If you migrated a 10.1.3.x WebCenter application, then the application continues to have the Model, ViewController, and Portlets projects.

---

2. Optionally, create new projects for defining the application data model, consuming content, or creating portlets:

   a. In the Application Navigator, right-click the application and select **New Project** from the context menu.

   b. In the New Gallery, expand **General**, select **Projects**, then **Generic Project**, and click **OK**.

   c. In the Create Generic Project wizard, in the **Project Name** field, enter a name for the project.

      For example `View`.

   d. In the **Directory** field, specify the directory path for storing the project or accept the default path.

   e. Click **Finish** to create the new project.

3. Right-click a project and select **Project Properties**.

4. In the Project Properties dialog, add the required technologies and libraries depending on which WebCenter-related capabilities you want to enable in the selected project:

   ■ To define the application data model:

     Include the technologies and libraries listed against the *Model* project in Table 3–1.

   ■ To consume content:

     Include the technologies and libraries listed against the *ViewController* project in Table 3–1.

   ■ To create portlets:

     Include the **WebCenter Portlet Creation Service** technology.

5. Save your work.

3.7 Creating a WebCenter Application Project By Importing a WAR File

This procedure is included in the event that you have an existing WAR file you want to import into your WebCenter application through JDeveloper.

To import a WAR file into JDeveloper:

1. Right-click your application in the Application Navigator, and choose **New Project** from the context menu.

2. In the New Gallery, expand **General**, select **Projects**, then **Project from War File**, and click **OK**.

3. Follow the wizard instructions to complete creating the project.

4. Right-click the newly created project in the Application Navigator, and select **Project Properties** from the context menu.

5. Add the required technologies and libraries depending on which WebCenter-related capabilities you want to enable in the selected project.

   For information about required technologies and libraries, see Table 3–1.

6. Click **OK** to close the Project Properties dialog.

3.8 Using Integrated WLS

Installation of Oracle WebCenter Framework reconfigures the Integrated WLS domain in JDeveloper to include additional libraries and several prebuilt portlets. This section discusses the Integrated WLS, including how to start and stop it, and describes some of the preconfigured portlet producers and prebuilt portlets it provides. It contains the following subsections:

- Section 3.8.1, "How to Start and Stop Integrated WLS"
- Section 3.8.2, "What You May Need to Know About Integrated WLS"
- Section 3.8.3, "The WebCenter Preconfigured Server Readme File"
- Section 3.8.4, "What You May Need to Know About Preconfigured Portlet Producers"

3.8.1 How to Start and Stop Integrated WLS

Options for starting Integrated WLS are available in the **Run** menu in Oracle JDeveloper.

- To start Integrated WLS in debug mode, select **Debug Server Instance** from the **Run** menu.

  Running the service in debug mode helps in debugging the service.

- To start Integrated WLS in the regular mode, select **Start Server Instance** from the **Run** menu.

There are a couple of ways to determine if Integrated WLS is running:

- From the **View** menu, select **Log** and look for the following entry:

  ```
  DefaultServer started
  ```

- Access the Integrated WLS console from a browser:

  ```
  http://localhost:7101/console
  ```

Note: Sometimes WebLogic Server is not accessible (for example, if a user tries to restart WebLogic Server too quickly, before it has successfully shut down). In this case, you may have to manually shut down or stop the Java process.

3.8.2 What You May Need to Know About Integrated WLS

Integrated WebLogic Server (Integrated WLS) is a preconfigured WebLogic Server that provides a complete Java 2 Enterprise Edition (J2EE) 1.4-compliant environment. It is written entirely in Java and executes on the Java Virtual Machine (JVM) of the standard Java Development Kit (JDK). You can run WebLogic Server on the standard JDK provided with your operating system or the one provided with Oracle JDeveloper.

You can use Integrated WLS as a platform for pretesting WebCenter application deployments on your local computer by establishing an application server connection to it from Oracle JDeveloper. When you run the application in Integrated WLS, it is actually deployed as if you were deploying it to a WebLogic Server instance in an application server. For more information about Integrated WLS, see Section 25.3, "Deploying a Custom WebCenter Application to an Oracle WebLogic Managed Server Instance".

3.8.3 The WebCenter Preconfigured Server Readme File

The WebCenter Preconfigured Server readme file contains valuable information about how to use Integrated WLS. Additionally, it contains links to the preconfigured portlet producers. You can access the Preconfigured Server readme file by selecting **WebCenter Preconfigured Server Readme** from the Oracle JDeveloper **Help** menu.

3.8.4 What You May Need to Know About Preconfigured Portlet Producers

The WebCenter preconfigured server provides a variety of ready-to-use portlets that you can add to your application pages. Simply register the producers contained in the default server with your WebCenter application, and then select the producers' portlets from Oracle JDeveloper's Application Resources panel.

This section provides a brief description of the preconfigured producers and some of the portlets they provide. It contains the following subsections:

- Section 3.8.4.1, "Preconfigured Portlet Producers and Portlets"
- Section 3.8.4.2, "The WSRP Sample Portlet Producers and Portlets"
- Section 3.8.4.3, "The PDK-Java Sample Portlet Producer and Portlets"

For information about registering portlet producers, see Section 9.2, "Registering Portlet Producers with a Custom WebCenter Application". For information about adding portlets to pages, see Section 9.3, "Adding Portlets to a Page".

3.8.4.1 Preconfigured Portlet Producers and Portlets

PortalTools provides access to the *design time at runtime* OmniPortlet and Web Clipping portlet. Design time at runtime means that users define portlet content after the portlet is placed on an application page and the page is run. This concept is explained more fully in Section 27.2.6, "Portlet Creation Style."

To access OmniPortlet and Web Clipping portlet producers:

1. Start the default server instance (see Section 3.8.1).
2. From the **Help** menu, select **WebCenter Preconfigured Server Readme**.
3. In the Readme file, go to the heading **Preconfigured Portlet Producers**, and click the **PortalTools Welcome Page** link under **PortalTools Portlet Producers**.

   This opens the PortalTools Welcome page.
4. On the PortalTools Welcome page, copy the URL of the Web Clipping Producer link, the OmniPortlet Producer link, or the Sample Portlet Producer link, and use it as the producer URL in the Oracle PDK-Java Portlet Producer Registration Wizard.

Note: For information about registering a portlet producer, see Section 9.2, "Registering Portlet Producers with a Custom WebCenter Application."

Once you have registered a producer, its portlets become available on Oracle JDeveloper's Application Resources panel. In the Application Resources panel, under the Connections node, select a producer name to list its portlets, then drag a portlet onto a WebCenter application page. (For information about adding portlets to pages, see Section 9.3, "Adding Portlets to a Page.".)

The PortalTools Welcome page contains producer URLs for three producers:

- The **Web Clipping producer** provides the Web Clipping portlet, which is a browser-based declarative tool that enables dynamic reuse of content from another Web source. When the source changes, the content in the Web Clipping portlet also changes. With the Web Clipping portlet, you use a Web browser to navigate to the Web page that contains the desired content. Using Web Clipping Studio, which is accessed through the portlet, drill down through a visual rendering of the target page to choose the desired content. For detailed information about the Web Clipping portlet, see Chapter 32, "Creating Content-Based Portlets with Web Clipping."
- The **OmniPortlet producer** provides OmniPortlet, which is a declarative portlet-building tool that enables you to build portlets against a variety of data sources, including XML files, character-separated value files (for example, spreadsheets), Web Services, databases, Web pages, and SAP data sources. OmniPortlet users can also choose a prebuilt layout for the data. Prebuilt layouts include tabular, news, bullet, form, chart, or HTML. For information about OmniPortlet, see Chapter 31, "Creating Portlets with OmniPortlet."
- The **Sample Portlet Producer** is useful for illustrating the potential of an OmniPortlet. The sample producer is for demonstration and should not be used to create real-use portlet instances.

3.8.4.2 The WSRP Sample Portlet Producers and Portlets

To access WSRP sample portlet producers:

1. Start Integrated WLS (see Section 3.8.1).
2. From the **Help** menu, select **WebCenter Preconfigured Server Readme**.
3. In the Readme file, go to the heading **Preconfigured Portlet Producers**, and click the link for the **Rich Text and Parameter Form Portlet Producer** or **Sample Portlets** under **WSRP Portlet Producers**.

   Both links open different WSRP Producer Test Pages—one for different WSRP producer versions of the Rich Text portlet, the other for different WSRP producer versions of sample portlets.
4. Copy the Web Services Description Language (WSDL) URL for a WSRP producer—either WSRP v1 WSDL or WSRP v2 WSDL.

   Use the copied link as the producer URL in the WSRP Producer Registration Wizard.

Note: For information about registering a portlet producer, see Section 9.2, "Registering Portlet Producers with a Custom WebCenter Application".

Depending on where the portlet producer connection was registered, its portlets appear either under Application Resources or in the Resource Palette. The WSRP WSDL URLs on the Rich Text Portlet Producer page include the Rich Text portlet, which offers browser-based rich text editing at runtime. For information about the Rich Text portlet, see Section 27.1.2.2, "Rich Text Portlet."

3.8.4.3 The PDK-Java Sample Portlet Producer and Portlets

To access PDK-Java sample portlet producers:

1. Start Integrated WLS (see Section 3.8.1).
2. From the **Help** menu, select **WebCenter Preconfigured Server Readme**.
3. In the Readme file, go to the heading **Preconfigured Portlet Producers**, and copy the link location for the **PDK-Java Sample Producer** or **PDK-Java Struts Sample Producer** link under **PDK-Java Portlet producers**.

   Use the copied link as the producer URL in the Oracle PDK-Java Portlet Producer Registration Wizard.

Note: For information about registering a portlet producer, see Section 9.2, "Registering Portlet Producers with a Custom WebCenter Application".

Depending on where the portlet producer connection was registered, its portlets appear either under Application Resources or in the Resource Palette. From here, you can drag and drop a variety of sample portlets onto your WebCenter application pages. Use the PDK-Java sample portlets to familiarize yourself with the types of functionality that are available through PDK-Java portlets.

3.9 Accessing Connection Wizards

To consume portlets and certain WebCenter Web 2.0 Services in your application pages, you must first set up connections to portlet producers and various data source repositories. For example, the Documents service requires a Content Repository connection, and the Links service requires a Database connection.

This section describes the different ways to access the wizards for creating new connections. The actual steps to create the connections are discussed in their respective chapters in this guide.

Depending on how you invoke a wizard to create a connection, the connection is created in one of the following locations:

- Under Application Resources in the Application Navigator

  Connections created here can be used in the current application only. This is the most common way to create a repository connection.

  For certain services, you can drag and drop a connection from Application Resources onto a page to create different types of task flow regions. To learn more, refer to the individual WebCenter Web 2.0 service chapters.

- Under IDE Connections in the Resource Palette

  Connections created here can be reused across WebCenter applications. To use these connections in an application, you just have to drag and drop the connection from the Resource Palette onto the Connections node in that application.

To access a connection wizard from the Application Navigator:

1. Right-click the **Connections** node under Application Resources and select **New Connection**, then select the connection type from the context menu.
2. Depending on your selection, the **Create <*Connection_Type*> Connection** dialog or wizard opens.

   The **Create Connection in** option is set to **Application Resources** by default.

To access a connection wizard from the Resource Palette

1. Click the **New** icon in the Resource Palette, select **New Connection**, then select the connection type from the context menu.
2. Depending on your selection, the **Create <*Connection_Type*> Connection** dialog or wizard opens.

   The **Create Connection in** option is set to **IDE Connections** by default.

You can also access a connection wizard from the New Gallery dialog. If you use this method, you must specify where to create the connection—under Application Resources or IDE Connections.

To access a connection wizard from the New Gallery:

1. From the **File** menu, choose **New**.
2. In the New Gallery dialog, expand **Connections**, select the type of connection you want to create, and click **OK**.

   Depending on your selection, the **Create <*Connection_Type*> Connection** dialog opens.
3. The **Create Connection in** option is set to **Application Resources** by default.

   You can select **IDE Connections** to create a connection in the Resource Palette.

See Also: For information about creating and consuming connections:

- Chapter 8, "Integrating Content"
- Chapter 9, "Consuming Portlets"
- Part III, "Integrating Services"

3.10 Integrating Mobile Support

Application Development Framework Mobile (ADF Mobile) is a standards-based framework that enables the rapid development of enterprise mobile applications. You can implement mobile support for your WebCenter application just like you would for any other Oracle ADF application.

Because ADF Mobile is built upon the component model of Java Server Faces (JSF), you can quickly develop applications for PDA browsers. ADF Mobile's mobile-specific extensions to JSF enable you to develop mobile applications using the same methodologies for developing JSF applications for the desktop. For more information about ADF Mobile, see *Oracle Fusion Middleware Mobile Developer's Guide for Oracle Application Development Framework*.

You also can build a mobile user interface using the APIs from various included products, such as Jive or UCM.

Note: Currently, mobile support is not available for WebCenter Web 2.0 Services.

4 Enabling Runtime Editing of Pages Using Oracle Composer

This chapter provides an overview of Oracle Composer and describes the basic tasks involved in enabling runtime editing in your application pages.

This chapter includes the following sections:

4.1 Introduction to Oracle Composer

When you create a service-oriented application, you may want to build in customization features so that one application can serve different audiences and domains. Ideally, users can edit the application at site and save the changes as metadata, separate from the base application definitions. Site-level customization eliminates the need to revise the application and redeploy it to the production environment. Because changes are stored separately, there are no issues when the application is upgraded and redeployed.

Oracle Composer, in conjunction with Metadata Services (MDS), provides a runtime editing tool that enables business users to edit application pages. Users can customize a page in *View mode* and *Edit mode*. Depending on the business requirement, users can save changes as *personalizations* or *customizations*. Personalizations affect an individual user’s view of the application and are available to all users. Customizations affect everyone’s view of the application and are available only to logged-in users with customization-level access privileges.

You can configure MDS to save changes in a single layer or in multiple layers. In a single-layer configuration, changes are saved to a common layer that is accessible to all users. In a multiple-layer configuration, changes are saved at separate locations based on specified criteria, for example, the mode in which the page is edited, the user editing the page, or the user role.

The tasks for creating, deploying, and using applications are performed by different categories of users in different environments:

- Application developers create applications in *design time*, which traditionally represents an IDE-based environment for creating or editing applications. Oracle JDeveloper provides the design-time environment for creating web applications.

- Application administrators deploy these applications to managed servers.
- End users access the deployed applications at *runtime*, which represents a browser-based environment for using web applications.

The concept of *design time at runtime* describes an environment that provides users with the option of editing application pages at runtime. Oracle Composer provides components for controlling the application's design time at runtime behavior. This section describes those components. It contains the following subsections:

- Section 4.1.1, "View and Edit Modes of a Page"
- Section 4.1.2, "Personalizing Capabilities in View Mode"
- Section 4.1.3, "Editing Capabilities in Edit Mode"
- Section 4.1.4, "Oracle Composer Components"
- Section 4.1.5, "Security and Oracle Composer"

4.1.1 View and Edit Modes of a Page

Changes can be made to a page in the following modes:

- View mode—The default mode in effect when you run an application to a browser. In this mode, you can perform tasks such as rearranging components, collapsing and expanding components, and changing the layout. You can save the changes you make in View mode as personalizations or customizations. How you save runtime changes depends on how MDS is configured in your application.
- Edit mode—The mode available to authenticated users with access privileges to edit page content. Oracle Composer provides controls for switching to page Edit mode and performing tasks such as rearranging components, collapsing and expanding components, changing the layout, adding components, deleting components, and editing component properties. You can save changes made to the page in Edit mode as personalizations or customizations. How you save runtime changes using MDS customization layers is described later in this chapter.

Note: To ensure clarity and consistency while discussing runtime editing, this document uses the terms *personalize* and *edit* while referring to the tasks performed in View mode and Edit mode respectively.

4.1.2 Personalizing Capabilities in View Mode

Oracle Composer enables users to personalize pages in View mode and edit pages in Edit mode. This section provides an overview of personalization tasks users can peform in page View mode. Such tasks include rearranging components, collapsing and expanding components, and changing the layout. You can enable the capabilities described in this section by adding the following Oracle Composer design-time components to a page: `Panel Customizable`, `Show Detail Frame`, `Custom Actions`, and `Layout Customizable`. For more information about these components, see Section 4.1.4, "Oracle Composer Components." Changes made to a page in View mode are available only to the user making these changes.

This section includes the following subsections:

- Section 4.1.2.1, "Expand and Collapse Components"
- Section 4.1.2.2, "Rearrange Components"

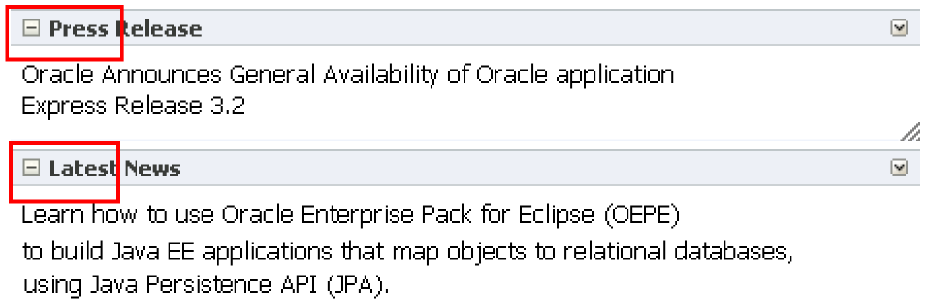
- Section 4.1.2.3, "Change the Layout"
- Section 4.1.2.4, "Customize Portlets"
- Section 4.1.2.5, "Switch Between Task Flow Views"

> **Note:** For a detailed description of the editing tasks that you can perform on a page, see "Personalizing Your Page View" and "Working with Page Content" in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

4.1.2.1 Expand and Collapse Components

A Collapse icon on the header of a `Show Detail Frame` or portlet enables users to collapse such a component and display only its header. Figure 4–1 shows the Collapse icon on a `Show Detail Frame` component.

Figure 4–1 Collapse Icon on a Show Detail Frame

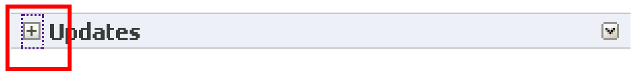


Oracle Announces General Availability of Oracle application Express Release 3.2

Learn how to use Oracle Enterprise Pack for Eclipse (OEPE) to build Java EE applications that map objects to relational databases, using Java Persistence API (JPA).

The Expand icon on a collapsed component enables users to disclose the component such that it displays the header and content. Figure 4–2 shows the Expand icon on a collapsed `Show Detail Frame` component.

Figure 4–2 Expand Icon on a Show Detail Frame



4.1.2.2 Rearrange Components

You can rearrange components in two ways:

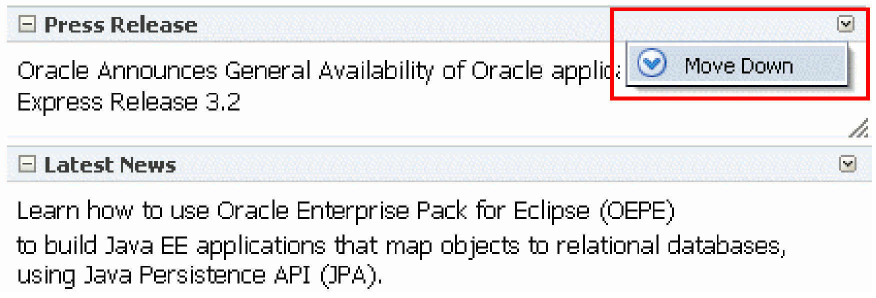
- Dragging and dropping

  Users can rearrange components by dragging and dropping them within the `Panel Customizable` component or across `Panel Customizable` components on the page. As it is difficult to identify a `Panel Customizable` component in View mode, users can simply drag the component over the spot they want to place it. A solid box indicates a receptive drop location.

- Using the Actions menu

The Move actions on a `Show Detail Frame` or portlet component enable users to move these components within a parent `Panel Customizable` component. If a `Panel Customizable` has multiple child components, then a child `Show Detail Frame` or portlet can be moved to the left and right or above and below, relative to the position of other child components. Figure 4–3 shows a sample `Show Detail Frame` component with a Move Down action. In this example, selecting **Move Down** moves the Press Release component immediately below the Latest News component.

Figure 4–3   Actions Menu on a Show Detail Frame



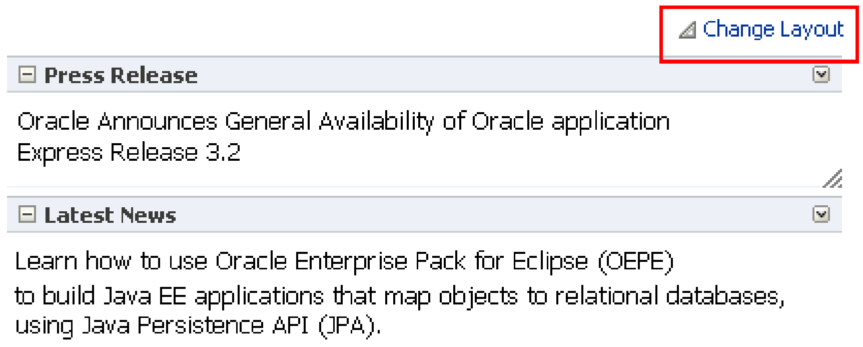
Oracle Announces General Availability of Oracle applic
Express Release 3.2

Learn how to use Oracle Enterprise Pack for Eclipse (OEPE)
to build Java EE applications that map objects to relational databases,
using Java Persistence API (JPA).

4.1.2.3 Change the Layout

A layout changer icon on the page enables users to define a different layout by selecting from a set of eight predefined layouts. The `Layout Customizable` component enables the display of a layout changer icon on the page, as shown in Figure 4–4.

Figure 4–4   Change Layout Icon

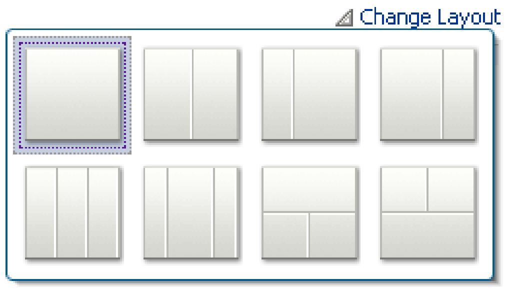


Oracle Announces General Availability of Oracle application
Express Release 3.2

Learn how to use Oracle Enterprise Pack for Eclipse (OEPE)
to build Java EE applications that map objects to relational databases,
using Java Persistence API (JPA).

Figure 4–5 shows the layout options available with the `Layout Customizable` component. A gray colored border highlights the layout currently applied to the page or area. The default page layout is `threeColumn`.

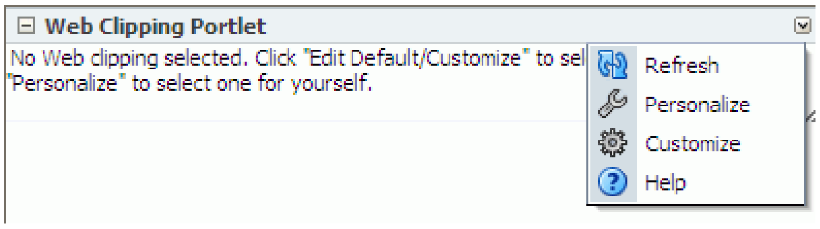
Figure 4–5 Layout Options



4.1.2.4 Customize Portlets

If the application contains portlets, then users can make them display the information that they want to see, hide them, or remove them from the page. Figure 4–6 shows the actions users can perform on a Web Clipping portlet.

Figure 4–6 Actions on a Web Clipping Portlet

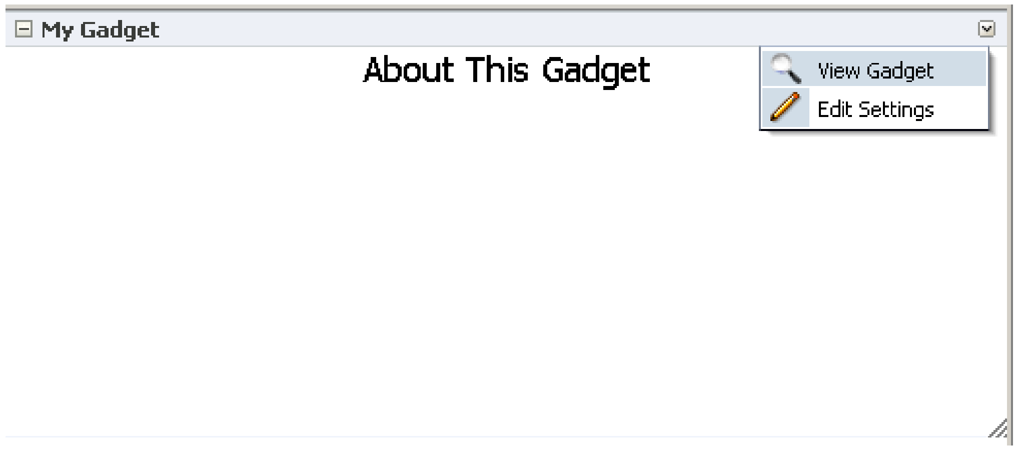


No Web clipping selected. Click "Edit Default/Customize" to sel
"Personalize" to select one for yourself.

4.1.2.5 Switch Between Task Flow Views

If you define custom actions on task flows enclosed in `Show Detail Frame` components, then users can switch between views of a child task flow using these custom actions. Figure 4–7 shows the custom actions available to switch between views of a child task flow.

Figure 4–7 Custom Actions on a Task Flow



4.1.3 Editing Capabilities in Edit Mode

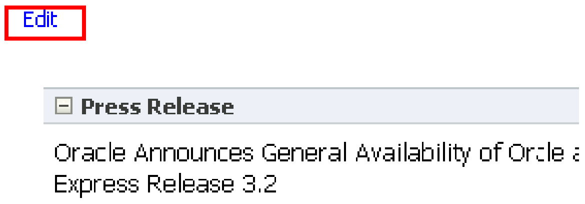
Oracle Composer's Edit mode provides a wide range of page customization capabilities. In Edit mode users can add content, edit page and component properties, delete components, rearrange components, change the page layout, and so on. For a detailed description of the runtime editing tasks that you can perform on a page, see "Personalizing Your Page View" and "Working with Page Content" in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

You can provide the capabilities described in this section to your application by adding the following Oracle Composer design-time components to a page: `Change Mode Link` or `Change Mode Button`, `Page Customizable`, `Panel Customizable`, `Show Detail Frame`, `Custom Actions`, and `Layout Customizable`. For more information about these components, see Section 4.1.4, "Oracle Composer Components."

The Change Mode Link or `Button` switches the page to Edit mode. In Edit mode, users can add content to the page, edit component properties, edit page properties, and so on. When you add the `Page Customizable` component to a page at design time, it enables these editing capabilities at runtime.

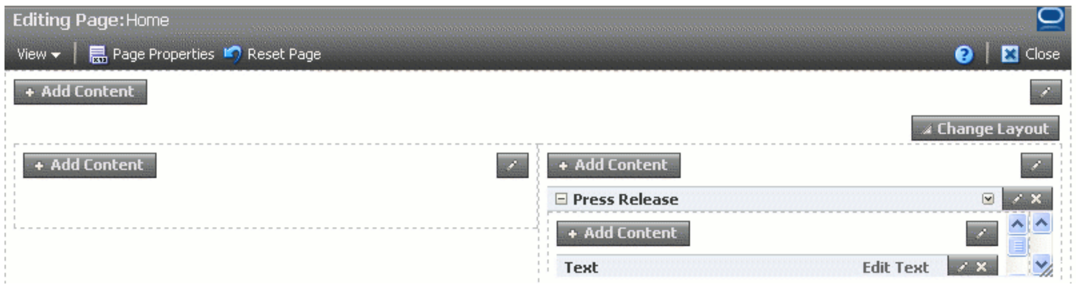
Figure 4–8 shows a `Change Mode Link` on an application page.

Figure 4–8 Edit Link While in View Mode



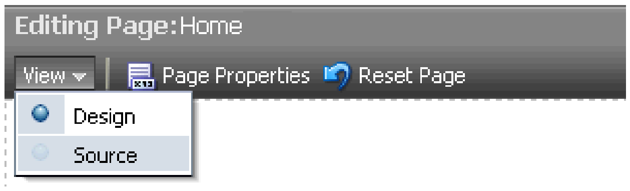
When users click the `Change Mode Link` or `Button`, the page opens in Design view as shown in Figure 4–9. Design view (the default) provides a WYSIWYG rendering of the page and its content, where Edit and Delete controls are directly selectable on each component. In Design view users can also perform such tasks as adding content, editing page and component properties, changing page layout, and deleting components.

Figure 4–9 Design View of a Page



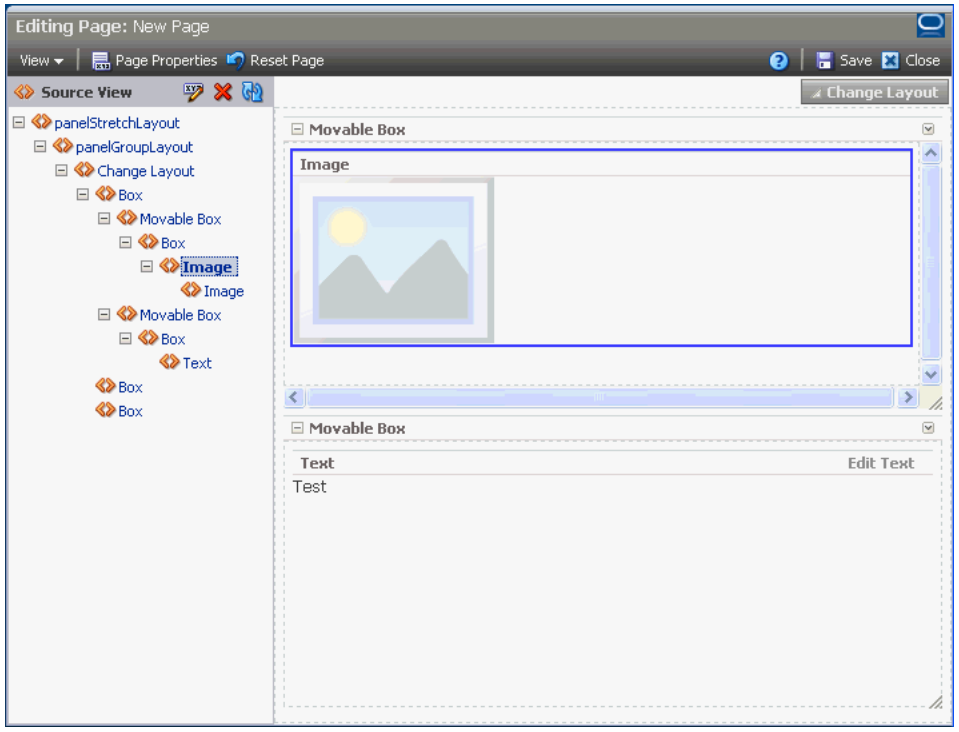
The View menu in page Edit mode (Figure 4–10) provides two viewing options—**Design** and **Source**.

Figure 4–10 View Menu in Oracle Composer



Source view (Figure 4–11) provides a WYSIWYG and a hierarchical rendering of page components, where Edit, Delete, and Refresh controls are available on the header of the hierarchical list. Source view is used to access and modify properties of components that are otherwise not selectable in the Design view. For example, ADF Faces components included in `Show Detail Frames` can be edited only in Source view.

Figure 4–11 Source View of a Page



In Source view, users can perform such tasks as editing page and component properties, changing page layout, and deleting components. However, users cannot move or rearrange components in Source view.

Note: The Save button is displayed on the Oracle Composer toolbar (both in Design view and Source view) only if you enabled a sandbox for the application. For more information about sandboxes, see Section 5.1.6, "Oracle Composer Sandbox."

This section provides an overview of the editing capabilities that Oracle Composer enables in page Edit mode. It includes the following subsections:

- Section 4.1.3.1, "Add Content"

- Section 4.1.3.2, "Rearrange Page Content"
- Section 4.1.3.3, "Edit Component Properties"
- Section 4.1.3.4, "Delete Components"
- Section 4.1.3.5, "Change the Layout"
- Section 4.1.3.6, "Edit Page Properties"
- Section 4.1.3.7, "View Component Hierarchy"
- Section 4.1.3.8, "Wire Components to Page Parameters"
- Section 4.1.3.9, "Reset Page"

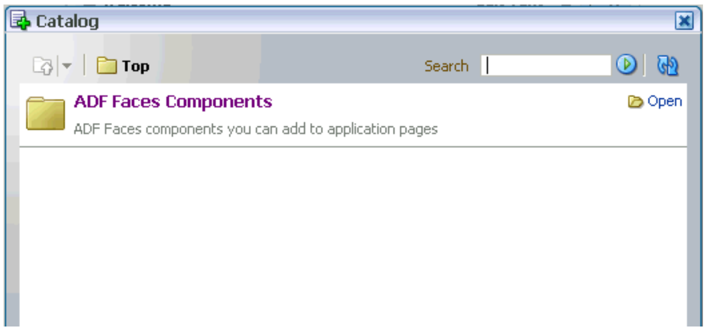
4.1.3.1 Add Content

Users can add content only in Design view. The **Catalog** dialog, shown in Figure 4–12, contains resources users can add to a page. These include documents, Oracle ADF components, portlets, and task flows. This dialog is invoked by clicking the **Add Content** button in any of the `Panel Customizable` components on the page.

Notes:

- Users can add content only inside a container component, such as a Panel Customizable or Box, which is represented by the dotted lines around a group of components.
- If you restrict customization on a container component, then users cannot add components inside it at runtime. See Section 5.8.1, "Applying Component-Level Restrictions by Defining Customization Policies" for more information.
- Task flows are exposed in the Catalog only if the catalog definition file has entries for the task flows. See Section 6.2.1.3, "Enabling Task Flows in the Resource Catalog" for more information.

Figure 4–12 Catalog Dialog



ADF Faces Components

ADF Faces components you can add to application pages

The Catalog dialog contains folders and components. An **Open** link next to an item in the Catalog indicates that the item is a folder. Users can drill down into the folder by clicking this link. An **Add** link next to an item indicates that the item can be added to the page. An Add link can appear next to a component or a folder. Users can navigate to a component or folder and add it to the page by clicking the Add link next to it. The component is added as the first child in the `Panel Customizable` component on which the Add Content link was clicked.

Depending on its configuration, the Catalog exposes all or a subset of the following components to users:

> **Note:** For information about Resource Catalog configurations, see Chapter 6, "Configuring the Resource Catalog for Oracle Composer."

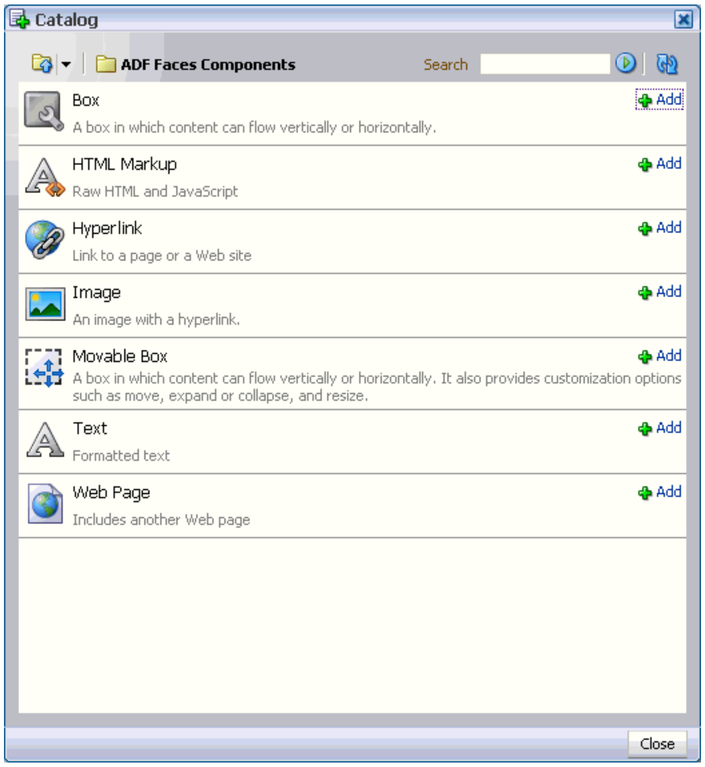
- ADF Faces Components

  The ADF Faces Components folder provides `Box`, `Movable Box`, and `Image` components that are analogous to the JDeveloper design time components `Panel Customizable`, `Show Detail Frame`, and `Image Link` respectively. In JDeveloper, these components are available in the Oracle Composer tag library.

  This folder also provides `HTML Markup`, `Hyperlink`, `Text`, and `Web Page` components that are analogous to the JDeveloper design time components `Output Text`, `Go Link`, `Rich Text Editor`, and `Inline Frame` respectively. In JDeveloper, these components are available in the ADF Faces tag library.

  The **ADF Faces Components** folder in the Catalog dialog contains these components, as shown in Figure 4–13.

Figure 4–13 ADF Faces Components You Can Add to the Page



Box
A box in which content can flow vertically or horizontally.

HTML Markup
Raw HTML and JavaScript

Hyperlink
Link to a page or a Web site

Image
An image with a hyperlink.

Movable Box
A box in which content can flow vertically or horizontally. It also provides customization options such as move, expand or collapse, and resize.

Text
Formatted text

Web Page
Includes another Web page

- Task Flows

  Users can add custom task flows and out-of-the-box service task flows only if you have configured them in the catalog definition file. See Section 6.2.1.3, "Enabling Task Flows in the Resource Catalog" for details.

A task flow added to a page is automatically enclosed in a `Movable Box` component. As a result, the task flow displays a header with options to move or delete the component.

> **Note:** The **Top** icon in the Catalog dialog enables users to return to a top-level folder.

- Portlets

  The **Portlets** folder lists all the registered producers. Users can navigate to the required portlet and add it to the page. The Portlets folder exposes portlets from any Java-PDK or WSRP producer that was registered in Oracle JDeveloper. For information about registering portlet producers, see Chapter 9, "Consuming Portlets."

- Documents

  The **Documents** folder lists documents and folders from the Documents service that users can add to the page.

  To display documents, ensure that you have performed the steps in Section 6.2.1.4, "Enabling Documents in the Resource Catalog."

See "Adding Content to a Page" in *Oracle Fusion Middleware User's Guide for Oracle WebCenter* for detailed information.

4.1.3.2 Rearrange Page Content

In Design view, users can rearrange content on the page in the following ways:

- Drag and drop a component within a `Panel Customizable` component or across `Panel Customizable` (or `Box`) components on the page.
- Use the Move actions on the Actions menu of a `Show Detail Frame` or portlet to move it within the parent `Panel Customizable` component. Depending on the number of child components in the `Panel Customizable` and how these components are oriented, a component can be moved to the left and right, or up and down.

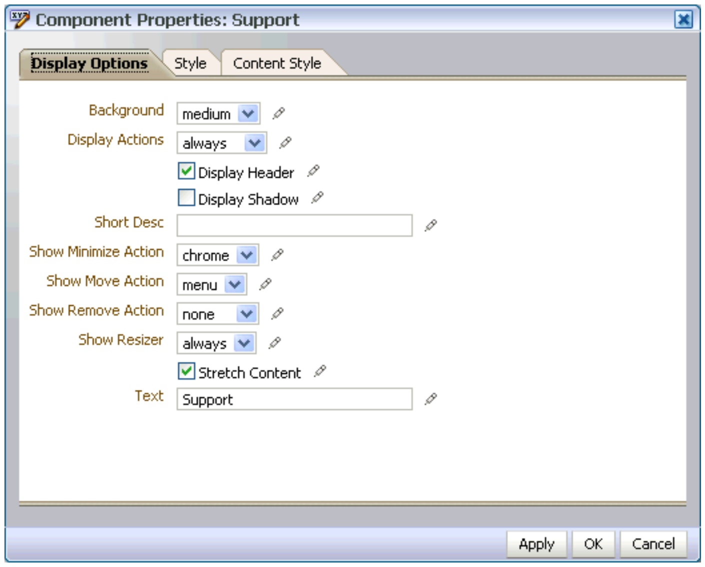
4.1.3.3 Edit Component Properties

The Component Properties dialog enables users to edit component properties and the parameters associated with components, such as portlets and task flows. The Component Properties dialog can be accessed in both Design view and Source view.

> **Note:** A component cannot be edited if:
>
> - the `Id` attribute was not set for the component at design time
> - the component or any of its attributes have been restricted
>
>   For information about restricting customization, see Section 5.8, "Overriding Default Security Behavior of Oracle Composer Components."
> - it is an ADF Faces component and Source view has been disabled for the application
>
>   For information about disabling Source view, see Section 5.9.1, "How to Disable Source View."

In Design view, clicking the Edit icon on a component displays the Component Properties dialog, shown in Figure 4–14.

Figure 4–14 Component Properties Dialog



Background: medium
Display Actions: always
[✓] Display Header
[ ] Display Shadow
Short Desc:
Show Minimize Action: chrome
Show Move Action: menu
Show Remove Action: none
Show Resizer: always
[✓] Stretch Content
Text: Support

In Source view, users can select a component from the hierarchy and click the **show properties of *component_name*** icon on the Source View panel header.

4.1.3.4 Delete Components

Users can delete a component from the page in the following ways:

- In Design view, by clicking the **Delete** icon on its header.
- In Source view, by selecting a component in the hierarchy and clicking the **Delete** icon on the header of the Source View panel.

  Alternatively, by right-clicking a component and selecting **Delete** from the context menu.

A Delete dialog prompts users to confirm deletion.

Notes:

- Of all components on the page, only direct child elements inside `Panel Customizable` (or `Box`) components can be deleted.
- If you defined type-level restrictions on a component, then the Delete icon is disabled for such a component at runtime. For more information, see Section 5.8.1.1, "How to Define Type-Level Customization Policies."

4.1.3.5 Change the Layout

In Design view, the **Change Layout** icon enables users to select a layout from a set of eight predefined layouts. The default page layout is `threeColumn`.

In Source view, users can select the `Layout Customizable` component and view its properties. The layout options are displayed in the Component Properties dialog. Users can select any predefined layout and click Apply or OK.

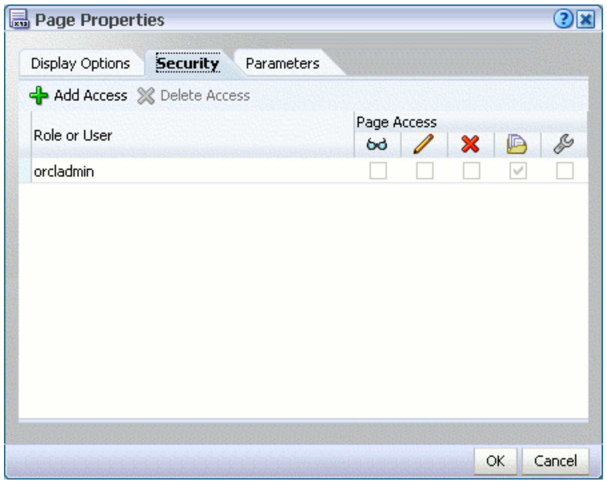
Note: Predefined layout options are available to users only if you have added a `Layout Customizable` component to the page at design time. For more information, see Section 4.1.4.3, "Layout Customizable."

4.1.3.6 Edit Page Properties

Users can access the **Page Properties** dialog from both Design view and Source view. In this dialog, users can create or edit a page's parameters and display properties. In addition, a Security tab is provided when you implement security in your application.

Figure 4–15 shows the Security tab in the Page Properties dialog.

Figure 4–15 Page Properties Dialog



Role or User	Page Access				
orcladmin	☐	☐	☐	✓	☐

On the Parameters tab, users can create page parameters that can be linked to component properties, thereby enabling any component on the page to adapt to the page context. For example, a page has a parameter called `SYMBOL` with the default value `ORCL`. This page contains the Stock Chart, Stock Price, and Company Info task flows. The task flows can be linked with the page parameter such that all the task flows respond to a change in the value of `SYMBOL` and show information pertaining to the chosen symbol.

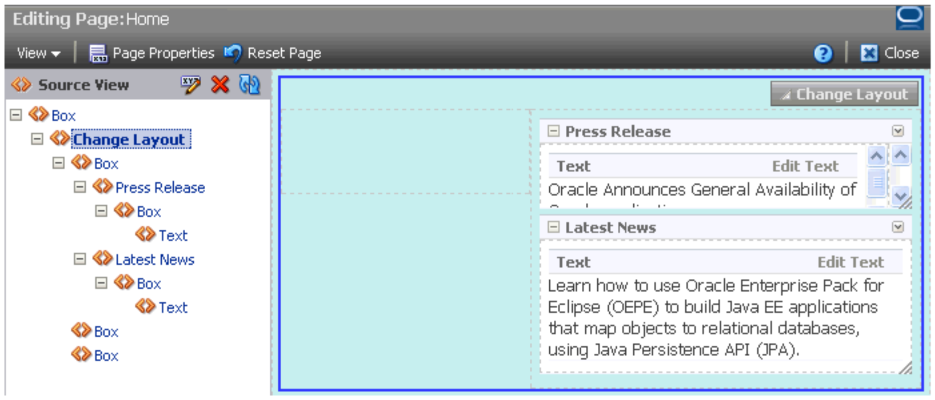
The Add a page parameter link enables users to add a new page parameter. A value can then be specified for this parameter on the Parameters tab.

See "Renaming Pages and Revising Page Keywords and Descriptions" and "Wiring Components and Pages" in *Oracle Fusion Middleware User's Guide for Oracle WebCenter* for detailed information about editing page settings.

4.1.3.7 View Component Hierarchy

Switching to Source view displays the hierarchy of page components in a tree structure. The root of the tree is the component that you added as the direct child of the `Page Customizable` component at design time (Figure 4–16).

Figure 4–16 Component Hierarchy in Source View



Oracle Announces General Availability of

Learn how to use Oracle Enterprise Pack for Eclipse (OEPE) to build Java EE applications that map objects to relational databases, using Java Persistence API (JPA).

When a node is selected in the tree structure, the corresponding component is selected on the page. Similarly, when a component is selected directly on the page, the corresponding node is selected in the tree structure.

4.1.3.8 Wire Components to Page Parameters

Components, such as portlets and task flows, may use parameters to link related components such that their contents are synchronized based upon the context. Oracle Composer enables users to link portlets and task flows with page parameters so that these components can read the page parameters and change their behavior accordingly.

Users can wire portlet and task flow parameters to page parameters using the Component Properties dialog. For more information, see "Wiring Pages, Task Flows, Portlets, and UI Components Together" in *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

In addition to wiring components and page parameters, Oracle Composer also enables users to pass values for display options and parameters through page URLs. For more information, see "Passing Parameter Values Through the Page URL" in *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

4.1.3.9 Reset Page

The Reset Page button enables users to remove all edits, regardless of when they were made, and reset the page to its original, out-of-the-box state. This button is available on the page in both Design view and Source view.

Note: Users cannot reset page customizations on the following:

- Components that are part of the page template
- Components located on pages within task flows

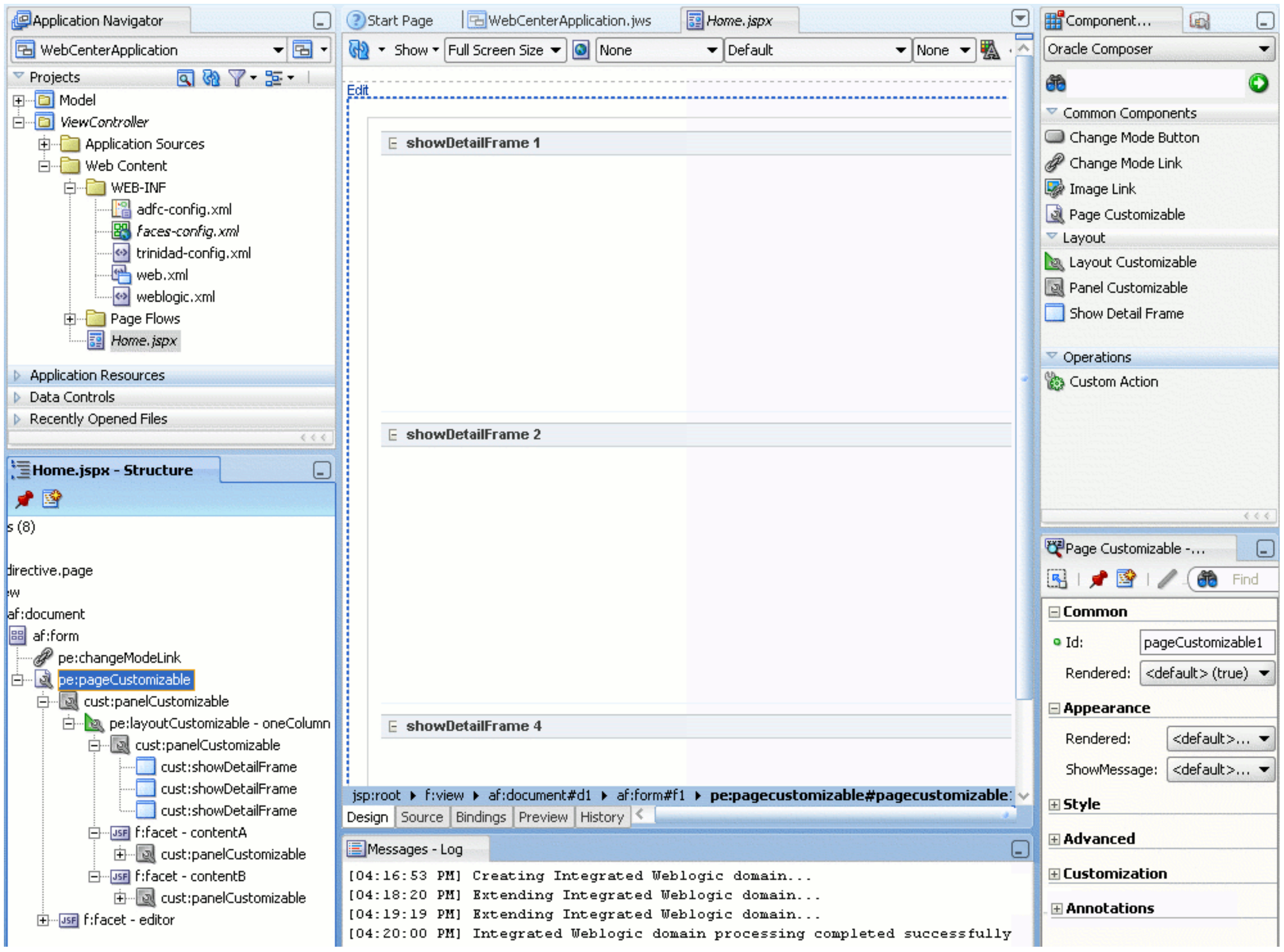
4.1.4 Oracle Composer Components

To make any JSPX document (`*.jspx`) editable at runtime, you must add Oracle Composer components to your page in Oracle JDeveloper at application design time.

Note: Oracle Composer works only with JSPX pages. You cannot add these components to JSP pages. For information about adding Oracle Composer components to your page, see Section 4.2, "Designing Editable Pages Using Oracle Composer Components."

The Oracle Composer tag library (Figure 4–17) available from the Component Palette provides components that you can add to make a page editable. By adding these components you can revise the layout and content of application pages.

Figure 4–17 Oracle Composer Tag Library in the Component Palette



```
[04:16:53 PM] Creating Integrated Weblogic domain...
[04:18:20 PM] Extending Integrated Weblogic domain...
[04:19:19 PM] Extending Integrated Weblogic domain...
[04:20:00 PM] Integrated Weblogic domain processing completed successfully
```

This section contains overviews of the Oracle Composer components that are used to enable page editing. It contains the following subsections:

- Section 4.1.4.1, "Page Customizable"
- Section 4.1.4.2, "Change Mode Link and Change Mode Button"
- Section 4.1.4.3, "Layout Customizable"
- Section 4.1.4.4, "Panel Customizable"
- Section 4.1.4.5, "Show Detail Frame"
- Section 4.1.4.6, "Custom Action"

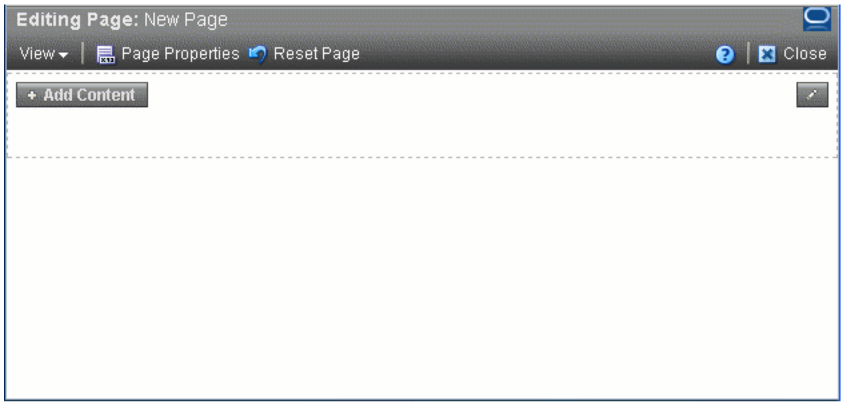
For more information about these components, see Appendix B, "Oracle Composer Component Properties and Files."

4.1.4.1 Page Customizable

The `Page Customizable` component defines the editable area of a page. Within this area, you can edit component properties, add content to the page, arrange content, and so on.

Adding a `Page Customizable` component enables the runtime inclusion of Oracle Composer on the page. End users can edit pages in Oracle Composer using page-related controls available across the top of the page, Add Content buttons on each content region, and Edit buttons on each component, as shown in Figure 4–18.

Figure 4–18 Page Customizable Component

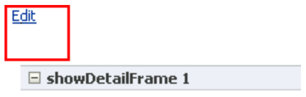


To enable runtime editing for multiple application pages in one operation, add a `Page Customizable` component to the ADF page template used for those pages. This avoids the need to manually add a `Page Customizable` to each page. For more information about adding the `Page Customizable` component, see Section 4.2.2, "How to Enable Runtime Customization Using a Page Customizable."

4.1.4.2 Change Mode Link and Change Mode Button

The `Change Mode Link` or `Change Mode Button` component provides an easy way to switch from View mode of the page to Edit mode. Figure 4–19 shows a `Change Mode Link` component on the page.

Figure 4–19 Change Mode Link Component



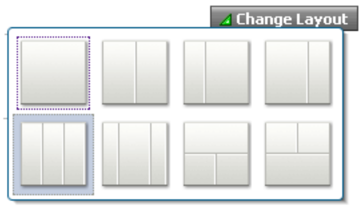
For more information about using `Change Mode Link` or `Change Mode Button`, see Section 4.2.3, "How to Enable Switching Between Page Modes Using a Change Mode Link or Change Mode Button."

4.1.4.3 Layout Customizable

The `Layout Customizable` component is a container that enables end users to lay out its child components in several predefined ways (for example, two column, three column, and so on). You can design your page such that all components on the page are enclosed in a `Layout Customizable` component. In such a case, the layout is applied to the entire page.

Access predefined layouts using the layout changer. By default, the layout changer displays as a small green icon both in page View mode and page Edit mode. By using the `Layout Customizable` component attributes you can choose to display the layout changer as an icon, text, or icon and text. In addition, you can decide whether to show or hide the layout changer in View mode. Figure 4–20 shows a Layout Customizable component and the predefined layouts that display when users click the layout changer.

Figure 4–20 Layout Customizable Component



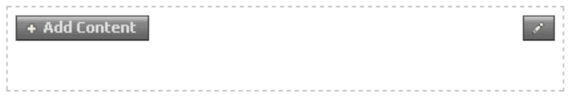
For more information, see Section 4.2.5, "How to Enable Layout Customization for a Page Using a Layout Customizable."

4.1.4.4 Panel Customizable

A `Panel Customizable` defines an area of the page onto which users can add components at runtime. Users can move or minimize `Show Detail Frame` components and portlets that are added as child components of a `Panel Customizable`. Users can also move these components across `Panel Customizable` components.

In Edit mode, the `Panel Customizable` component is rendered as a box with dotted lines. In fact, a `Panel Customizable` is referred to as a `Box` in the runtime Catalog. An Add Content button appears on each `Panel Customizable` component on the page, as shown in Figure 4–21. You can use this button to open the Resource Catalog Viewer and add components within the `Panel Customizable`.

Figure 4–21 Panel Customizable Component

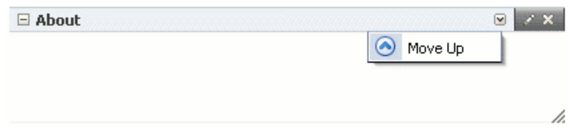


For more information, see Section 4.2.4, "How to Define Editable Areas of a Page Using Panel Customizable Components."

4.1.4.5 Show Detail Frame

A `Show Detail Frame` component renders a border or chrome around its child component along with a header that contains an Actions menu. The actions available on this menu enable users to move the component, along with its content, to new positions on the page (Figure 4–22). Note that a `Show Detail Frame` must be included inside a `Panel Customizable` component for it to be movable.

Figure 4–22 Show Detail Frame Component



A `Show Detail Frame` component enables the following types of actions:

- Collapse and expand the child component
- Move content to different positions on the page
- If you add a task flow as a child component, then it enables task flow navigation using the Actions menu.
- If you add an ADF Faces `Rich Text Editor` as a child component, then it enables end users to edit and save text in the `Rich Text Editor`.

You can add your own UI controls to further customize the display of content by using facets of the `Show Detail Frame`. For information about using these facets, see Section 4.2.11, "How to Enable Custom Actions on Show Detail Frame Components by Using Facets."

For information about adding this component to a page, see Section 4.2.6, "How to Enable Component Customization Using Show Detail Frame Components."

4.1.4.6 Custom Action

Use `Custom Action` components to trigger navigational flow in a task flow when a region is included inside a `Show Detail Frame` component. You can define custom actions corresponding to the ADFc outcomes of the task flows. At runtime, these custom actions are displayed on the `Show Detail Frame` header as icons, menu options, or both.

A `Custom Action` must always be defined as a child component of a `Show Detail Frame` and is useful only when the `Show Detail Frame` also contains a task flow as its child component.

You can define custom actions for `Show Detail Frame` components at the global and instance levels. For more information, see Section 4.2.13, "How to Enable Custom Actions on a Show Detail Frame Enclosing a Task Flow."

4.1.5 Security and Oracle Composer

You can define application security on many levels, including on a page, an operation, a component, or a component attribute. Oracle Composer provides a means of addressing the various levels of application security. For example, before users can customize components, the components’ sources of security restrictions are queried to determine if customization is allowed. If a component does not permit customization, Oracle Composer considers it restricted; that is, Oracle Composer treats the component as if it were explicitly secured. Table 4–1 describes Oracle Composer behavior to reflect customization restrictions.

Table 4–1 Customization Restrictions on Oracle Composer Components

Restriction	Behavior
The page (and its contents) is open for all customizations	Oracle Composer allows editing of components without restrictions.

Table 4–1 (Cont.) Customization Restrictions on Oracle Composer Components

Restriction	Behavior
The page is restricted	Oracle Composer displays but none of the options are accessible.
Some operations are restricted	Oracle Composer displays but the options corresponding to the restricted operations are disabled.
A component is restricted	The component is selectable. Oracle Composer shows all its properties as read-only. Oracle Composer's API enables the developer to determine if a component is restricted or not.
A component's attributes are restricted	The component is selectable. Oracle Composer shows its restricted properties as read-only. Oracle Composer's API enables the developer to determine if a component's attributes are restricted or not.

Depending on the privileges granted while implementing security, users can perform different personalization and editing tasks when they log in to the application at runtime.

This section describes the default security behavior of Oracle Composer components. It contains the following subsections:

- Section 4.1.5.1, "Page Security"
- Section 4.1.5.2, "MDS Customization Restrictions"
- Section 4.1.5.3, "Component Action-Level Security"

For information about overriding default security behavior, see Section 5.8, "Overriding Default Security Behavior of Oracle Composer Components."

4.1.5.1 Page Security

The page definition file is the respository for page-level security information. Oracle Composer enables editing capabilities based on a user's privileges:

- A user with the Personalize privilege on a page can perform only the personalizations described in Section 4.1.2, "Personalizing Capabilities in View Mode."
- A user with the Edit or Customize privilege can perform all runtime editing tasks, such as adding content, editing component properties, and deleting components. Oracle Composer and WebCenter Customizable Components do not differentiate between Edit and Customize privileges.

Notes: Oracle Composer and WebCenter Customizable Components support cascading of page privileges with Grant being a super set of all privileges. A user with Grant privilege on a page is considered to have Edit, Personalize, and View privileges. A user with Personalize privilege is considered to additionally have the View privilege.

Table 4–2 explains Oracle Composer behavior based on page-level privileges. Only those page privileges that are relevant to Oracle Composer and WebCenter Customizable Components privileges are listed in this table. The Grant privilege is not listed as it is a super set of all privileges. Users with the Grant privilege can perform all editing tasks. The table also does not list the View privilege because users with the View privilege cannot personalize or edit a page.

Table 4–2 Mapping of Page Privileges to Oracle Composer Behavior

Privilege	**Oracle Composer Behavior**
Edit or Customize	Users can switch to Edit mode, where Oracle Composer is invoked, and edit the page. Oracle Composer does not differentiate between the Edit and Customize privilege. That is, users with either the Edit or Customize privilege can perform all runtime editing tasks. With the Edit or Customize privilege, users can: ■ Add content ■ Edit component properties ■ Rearrange content ■ Delete components in Oracle Composer ■ Personalize or customize a portlet ■ Move a portlet or task flow in View mode. This is persisted as a personalization. ■ Expand and collapse a task flow or portlet in View mode. This is persisted as a personalization. ■ Resize a column of a table in a task flow. This is persisted as a personalization. ■ Reset the page to its original state ■ Delete components in View mode If users without Edit or Customize permission try to edit a page, a message appears stating that they do not have permission to do so.
Personalize	In View mode users can: ■ Rearrange content ■ Personalize a portlet ■ Move a portlet or task flow in View mode. This is persisted as a personalization. ■ Expand and collapse a task flow or portlet in View mode. This is persisted as a personalization. ■ Resize a column of a table in a task flow. This is persisted as a personalization. ■ Delete components in View mode **Note**: Having Personalize permission does not enable users to perform portlet customizations. If users without Personalize, Edit, or Customize permission try to edit a page, a message appears stating that they do not have permission to do so.

4.1.5.2 MDS Customization Restrictions

WebCenter applications have a default MDS configuration that restricts customization on all application objects. This default restriction must be addressed if you intend to enable runtime page editing. You can address default restrictions by adding a `Page Customizable` component to the page. Default customization restrictions do not affect the `Page Customizable` and all its child components.

You can further enable customization on a set of attributes for the component using MDS type-level restrictions or instance-level restrictions. Type-level restrictions are applicable to a specified component type across instances. At runtime, attributes on which you have enabled customization are shown as editable properties in Oracle Composer, and restricted attributes are shown as read-only properties. For information, see Section 5.8.1, "Applying Component-Level Restrictions by Defining Customization Policies."

4.1.5.3 Component Action-Level Security

`Show Detail Frame` components enable the placement of restrictions on individual supported actions. For example, one can specify a restriction on whether the current user is allowed to minimize the `Show Detail Frame`.

It is left to components to enforce restrictions on their supported actions and to provide well-defined APIs to check the availability of the action. You can specify restriction on `Show Detail Frame` components actions in `adf-config.xml`. If a restriction is specified and applicable to the current user, the `Show Detail Frame` does not render the action.

For information about applying action-level restrictions, see Section 5.8.4, "Applying Action-Level Restrictions on Show Detail Component Actions."

Similarly, you can define customization restrictions on a `Layout Customizable` component. Specifically, if you place a restriction on the `Show Layout Changer` attribute, then layout changer options are shown as disabled and users cannot use them to change the page layout.

4.2 Designing Editable Pages Using Oracle Composer Components

The Oracle Composer tag library provides design-time components that you can add to a page in Oracle JDeveloper to enable runtime page editing. When you create a page with Composer components at design time, at runtime Oracle Composer provides options for entering page edit mode and designing the page according to your requirements.

You can enable customizations in WebCenter and non-WebCenter applications. Within an application, you can enable customization of the following types of pages:

- A regular JSPX page, not based on a page template
- A JSPX template page
- A JSPX page based on a template

You must add Oracle Composer components to a page at design time to make it editable at runtime. The Oracle Composer tag library provides the components required for making the page editable. For more information, see Section 4.1.4, "Oracle Composer Components."

This section explains how to add Oracle Composer components to a page at design time to make it editable at runtime. It contains the following subsections:

- Section 4.2.1, "How to Create a Customizable Page"
- Section 4.2.2, "How to Enable Runtime Customization Using a Page Customizable"
- Section 4.2.3, "How to Enable Switching Between Page Modes Using a Change Mode Link or Change Mode Button"
- Section 4.2.4, "How to Define Editable Areas of a Page Using Panel Customizable Components"

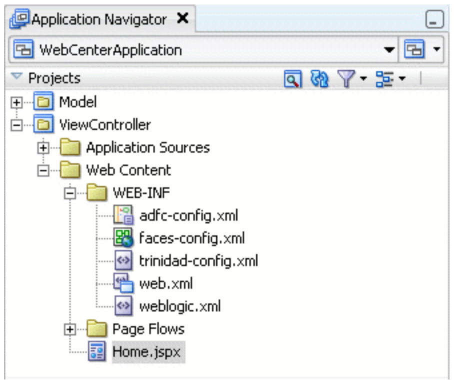
- Section 4.2.5, "How to Enable Layout Customization for a Page Using a Layout Customizable"
- Section 4.2.6, "How to Enable Component Customization Using Show Detail Frame Components"
- Section 4.2.8, "What Happens When You Add Oracle Composer Components"
- Section 4.2.9, "What Happens at Runtime"
- Section 4.2.10, "What You May Need to Know When Designing Editable Pages"
- Section 4.2.11, "How to Enable Custom Actions on Show Detail Frame Components by Using Facets"
- Section 4.2.12, "How to Enable Custom Actions on Show Detail Frame Components By Using Facets: Example"
- Section 4.2.13, "How to Enable Custom Actions on a Show Detail Frame Enclosing a Task Flow"
- Section 4.2.14, "How to Enable Custom Actions On a Show Detail Frame Enclosing a Task Flow: Example"
- Section 4.2.15, "How to Create Event-Enabled Task Flows"
- Section 4.2.16, "How to Apply Styles to Components"

4.2.1 How to Create a Customizable Page

The steps for creating a customizable JSPX page in your WebCenter application are available in Section 3.3, "Creating WebCenter Application-Enabled Pages."

When you create a new page in JDeveloper, it is listed in the Application Navigator under **Web Content** as shown in Figure 4–23. Additionally, the page is opened in the editor and becomes the active editor panel.

Figure 4–23 New Customizable Page (Home.jspx) Shown in the Application Navigator



Security Considerations

You can test how user privileges impact runtime customization capabilities by implementing security and configuring your application to authenticate users such that they have distinct identities. For the steps to implement a basic security model in your application, see Section 11.1.1.1, "Implementing Security for Services."

To enable users to edit a page in a secured application, you must ensure that you grant Edit or Customize privileges to users or roles.

4.2.2 How to Enable Runtime Customization Using a Page Customizable

Adding a `Page Customizable` component to the page ensures that Oracle Composer is invoked when users switch to Edit mode of the page. When you add a `Page Customizable` component, some configuration files are updated automatically with the default composer-specific settings. For more information, see Section 4.2.8, "What Happens When You Add Oracle Composer Components."

Note: For considerations you must make before adding a `Page Customizable` to the page, see Section 4.2.10, "What You May Need to Know When Designing Editable Pages."

To add a Page Customizable component to the page:

1. Open a customizable JSPX page.
2. In the Component Palette, select **ADF Faces** and drag a **Panel Stretch Layout** component onto the page.

Notes:

- The `Panel Stretch Layout` component stretches the child in the `center` facet to fill all of the available space. This holds true when users resize the browser.
- You can set the `styleClass` attribute to `AFVisualRoot` to ensure that the `Panel Stretch Layout` component occupies the complete width of the page at runtime.

---

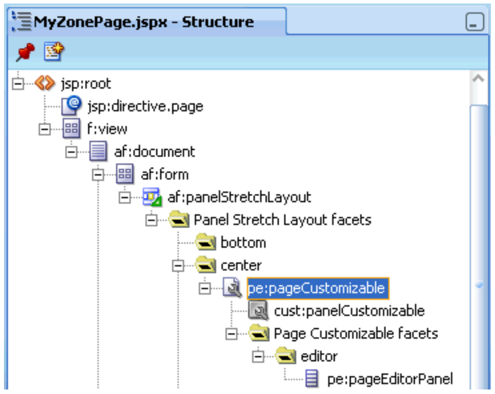
3. In the Component Palette, select **Oracle Composer**.
4. Add a **Page Customizable** component to the `center` facet of the `Panel Stretch Layout`.

   You must ensure that the `Page Customizable` is nested inside an `af:document` element. The `Page Customizable` is a rich client component that requires the rich client framework to function properly.
5. The required attributes for a `Page Customizable` component are populated with default values when you add the component to the page.

   You can define or modify attribute values by referring to Table B–1 in Section B.1, "Oracle Composer Component Properties."
6. Save the page.

   By default, a `Panel Customizable` component is added as a child component and the `Page Editor Panel` is added as a facet in the `Page Customizable` component, as shown in Figure 4–24.

Figure 4–24 Page Customizable Component



Example 4–1 shows the `pe:pageCustomizable` tag in the page source view.

Example 4–1 Page Customizable Component in the Page Source View

```
<pe:pageCustomizable id="pageCustomizable1"
  <cust:panelCustomizable id="panelcustomizable1"
                          layout="scroll"/>
  <f:facet name="editor">
    <pe:pageEditor id="pep1"/>
  </f:facet>
</pe:pageCustomizable>
```

4.2.3 How to Enable Switching Between Page Modes Using a Change Mode Link or Change Mode Button

To enable users to switch to Edit mode of a page easily, you must add a `Change Mode Link` or `Change Mode Button` component to the page.

Note: An alternative way to enable switching to Edit mode is by using the Change Mode API. For more information about this API, see *Oracle Fusion Middleware Composer Components Java API Reference for Oracle WebCenter*.

For things you should consider before adding a `Page Customizable` to the page, see Section 4.2.10, "What You May Need to Know When Designing Editable Pages."

To add a Change Mode Link or Change Mode Button component:

1. From the Component Palette, select **Oracle Composer**.
2. In the Structure window, within the `top` facet of the `Panel Stretch Layout` that you added in the previous section, drag a **Change Mode Link** or **Change Mode Button** component.

   You must ensure that the `Change Mode Link` or `Change Mode Button` is nested in an `af:document` element. The `Change Mode Link` or `Change Mode Button` component is a rich client component that requires the rich client framework to function properly.

Notes:

- If you have not used a `Panel Stretch Layout` on your page, then add the `Change Mode Link` or `Change Mode Button` component above the `Page Customizable` component in the Structure window. This ensures that the `Change Mode Link` or `Change Mode Button` is displayed properly at runtime.
- Use a `Change Mode Link` or `Change Mode Button` only when you have a `Page Customizable` on the page. You may have problems running a page that contains a `Change Mode Link` or `Change Mode Button` but no `Page Customizable` component.

3. The required attributes for a `Change Mode Link` or `Change Mode Button` component are set by default when you add the component to the page.

   Optionally, you can set any other attributes by referring to Table B–2 in Section B.1, "Oracle Composer Component Properties."

   The `pe:changeModeLink` or `pe:changeModeButton` tag displays within the `af:form` tag in the Structure window, as shown in Figure 4–25.

Figure 4–25 Change Mode Link

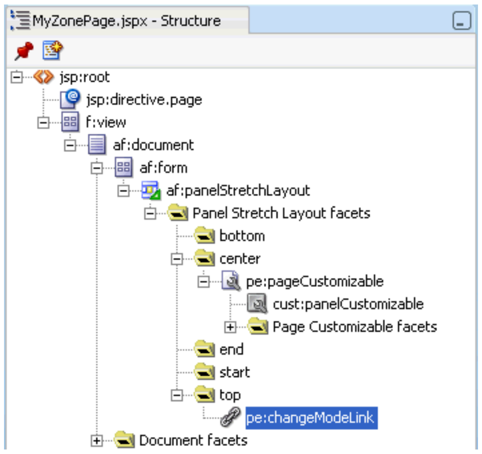
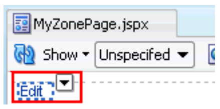


Figure 4–26 shows the `Change Mode Link` in the Design view of the page in JDeveloper.

Figure 4–26 Change Mode Link in Design View



4.2.4 How to Define Editable Areas of a Page Using Panel Customizable Components

The `Panel Customizable` component is required for page composition or content management tasks, such as adding, arranging, or removing portlets or regions. By default, one `Panel Customizable` component is automatically added as a direct child of the `Page Customizable` component. You can add more `Panel`

`Customizable` components within this `Panel Customizable` component according to your requirements.

All components within a `Panel Customizable` component are available for selection at runtime and can be edited. It is only within a `Panel Customizable` component that you can drag and drop components at runtime.

Note: For considerations you must make before adding a `Page Customizable` to the page, see Section 4.2.10, "What You May Need to Know When Designing Editable Pages."

To add a Panel Customizable component to the page:

1. From the Component Palette, select **Oracle Composer**.
2. Drag a **Panel Customizable** component to the Structure window and drop it at any suitable location within the form.

   You must ensure that the `Panel Customizable` is nested in an `af:document` element. The `Panel Customizable` component is a rich client component that requires the rich client framework to function properly.

   Notes:

   - A `Page Customizable` component contains one direct child `Panel Customizable` component by default. Do not add another `Panel Customizable` as a direct child component of the `Page Customizable`. If the `Page Customizable` has multiple child components, you may get errors while running the page.
   - Ensure that you do not delete the root `Panel Customizable` component on the page, because at runtime you can drop components *only* inside a `Panel Customizable` component.

   ---

3. The required attributes for a `Panel Customizable` component are set by default when you add the component to the page.

   Optionally, you can set any other attributes by referring to Table B–6 in Section B.1, "Oracle Composer Component Properties."

   Notes: If you select `stretch` layout for the `Panel Customizable`, then the first child component is stretched to fill up available space in the Panel `Customizable` component. Any other child components are ignored, though they are not removed from the page.

4.2.5 How to Enable Layout Customization for a Page Using a Layout Customizable

Use the `Layout Customizable` component to enable the runtime definition or modification of the layout of components on a page or an area of a page.

To add a Layout Customizable component:

1. From the Component Palette, select **Oracle Composer**.
2. Drag a **Layout Customizable** component to the Structure window and drop it inside the `Panel Customizable` component.

The target `Panel Customizable` component must be a child of the `Page Customizable` component.

Ensure that the `Layout Customizable` is nested in an `af:document` element. The `Layout Customizable` is a rich client component and requires the rich client framework to function properly.

Note: You can delete the direct child `Panel Customizable` in the `Page Customizable` and add the `Layout Customizable` as a direct child of the `Page Customizable`. However, you must ensure that the `Page Customizable` has only one direct child component.

---

3. The required attributes for a `Layout Customizable` component are set by default when you add the component to the page.

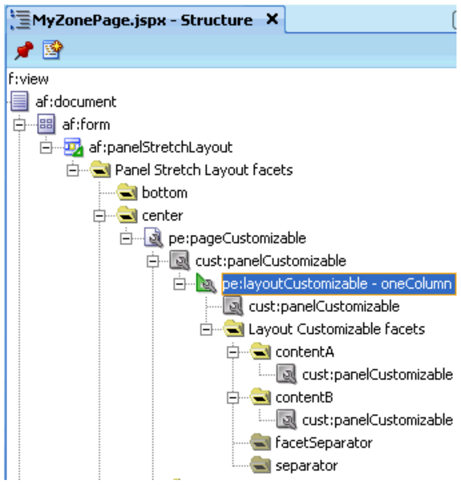
   Optionally, you can set any other attributes by referring to Table B–3 in Section B.1, "Oracle Composer Component Properties."

Note: To ensure that the `Layout Customizable` component is clearly visible on the page at runtime, provide a descriptive label for the component by using the `Text` attribute.

The `pe:layoutCustomizable` tag is located inside a `cust:panelCustomizable` tag in the Structure window as shown in Figure 4–27. A child `Panel Customizable` component is added by default in the `Layout Customizable` component. Additionally, a `Panel Customizable` component is added within each facet of the `Layout Customizable` component. These `Panel Customizable` components enable you to add content inside the `Layout Customizable` component at runtime.

The `Panel Customizable` added as a direct child provides the main area—the central area in a layout at runtime. The `Panel Customizable` components added within the two default `Layout Customizable` facets provide the two content areas, `A` and `B`. When you select a predefined layout at runtime, these three areas are arranged to display content in the selected pattern. See Predefined Layout Types for more information about how the content is laid out for each layout type.

Figure 4–27 Layout Customizable Component



```
f:view
  af:document
  af:form
    af:panelStretchLayout
      Panel Stretch Layout facets
        bottom
        center
          pe:pageCustomizable
            cust:panelCustomizable
              pe:layoutCustomizable - oneColumn
                cust:panelCustomizable
                Layout Customizable facets
                  contentA
                    cust:panelCustomizable
                  contentB
                    cust:panelCustomizable
                  facetSeparator
                  separator
```

4.2.6 How to Enable Component Customization Using Show Detail Frame Components

When you want to enable customizations, such as moving, minimizing, or removing components, you can drop a `Show Detail Frame` component inside a `Panel Customizable` component on the page. You can then add a component inside the `Show Detail Frame`.

Note: Each `Show Detail Frame` component should have only one direct child component. If you add multiple child components, then only the first one is rendered. The other direct child components are not rendered at design time or runtime.

If multiple components must be enclosed in a `Show Detail Frame`, then add a grouping component like `Panel Group Layout` or `Panel Customizable` to the `Show Detail Frame` component and then include the ADF Faces components or other content within this grouping component.

Use the `Show Detail Frame` component to enable customizations in View and Edit modes of the page. Changes made in View mode are available to that user only, and changes made in Edit mode are available to all application users.

To add a Show Detail Frame component to the page:

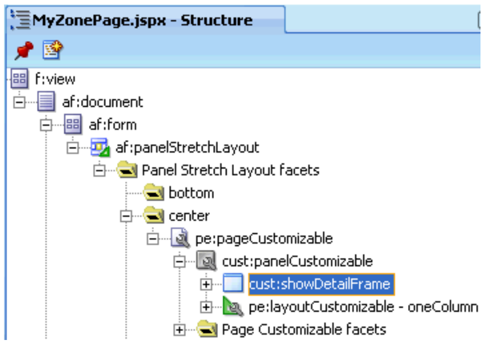
1. From the Component Palette, select **Oracle Composer**.
2. Drag a **Show Detail Frame** component to the Structure window and drop it inside a `Panel Customizable` component.

   The `Show Detail Frame` should be nested in a `Panel Customizable` component on the page.
3. The required attributes for a `Show Detail Frame` component are set by default when you add the component to the page.

   Optionally, you can set any other attributes by referring to Table B–7 in Section B.1, "Oracle Composer Component Properties."

   The `cust:showDetailFrame` tag is added inside the `cust:panelCustomizable` tag as show in Figure 4–28.

Figure 4–28 Show Detail Frame Component



For a better understanding of this type of task, see Section 4.3, "Designing Editable Pages Using Oracle Composer Components: Example."

4.2.7 How to Enable Customization in a Populated Page

When you have an existing ADF application with JSPX pages that are populated with content, and you want to enable customization, you can do so by moving all content inside a `Page Customizable` component.

You must first add the `Page Customizable`, then a `Layout Customizable`, and then the required hierarchy of `Panel Customizable` and `Show Detail Frame` components. Drag each of the existing components and drop them onto suitable locations inside the `Page Customizable`.

Note: For best results, move components using the Structure window and *not* by editing the source code.

4.2.8 What Happens When You Add Oracle Composer Components

When you add a `Page Customizable` component to the page, the following configurations are performed automatically:

- A default Resource Catalog definition file, `default-catalog.xml`, is configured in the application. The `default-catalog.xml` file is located in the `<`*`Application_Root`*`>\mds\oracle\adf\rc\metadata` directory.
- A Resource Catalog Viewer is configured for the application. At runtime, Oracle Composer provides users the option to add resources from this viewer to the page.
- When you create an application, a minimal `adf-config.xml` file is also created. When you add a `Page Customizable` to an application page, the required configuration is added to the `adf-config.xml` file.
- The `web.xml` file available in the `<`*`Application_Root`*`>\<`*`Project_Name`*`>\public_html\WEB-INF` directory is updated to:
  - Enable change persistence
  - Configure the MDS JSP provider
- The `DataBindings.cpx` file in the `<`*`Application_Root`*`>\<`*`Project_Name`*`>\public_html\WEB-INF\adfmsrc\<`*`Project_Name`*`>` directory, is updated to enable the presence of task flows on the page.
- The page definition file is updated with the binding for the Oracle Composer task flow, which is available as part of the WebCenter Framework extension JAR file.

Example 4–2 shows the code in the page definition file after a `Page Customizable` component is added to an application page.

Example 4–2 Page Definition File After Adding Page Customizable Component

```
<?xml version="1.0" encoding="UTF-8" ?>
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
                version="11.1.1.46.16" id="customizable_ComposerPagePageDef"
                Package="customization.pageDefs">
  <parameters/>
  <executables>
    <taskFlow id="pageeditor"
              taskFlowId="#{pageEditorBean.pageEditor}"
              xmlns="http://xmlns.oracle.com/adf/controller/binding"/>
  </executables>
  <bindings/>
</pageDefinition>
```

See Also: Section B.2, "Oracle Composer-Specific Files and Configurations" for information about files that are created or modified when you add Oracle Composer components.

4.2.9 What Happens at Runtime

At runtime, users can perform all the tasks described Section 4.1.2, "Personalizing Capabilities in View Mode" and Section 4.1.3, "Editing Capabilities in Edit Mode."

Each Oracle Composer component provides runtime capabilities that are described in Section 4.1.4, "Oracle Composer Components."

Note: Runtime customizations that you perform on a page are not carried over when you deploy the application to a target server.

4.2.10 What You May Need to Know When Designing Editable Pages

When adding Oracle Composer components to your customizable page, consider the following:

- To enable runtime customization of components, add only one `Page Customizable` to a page.

  Note: Do *not* add a second `Page Customizable` component to your page. This causes an error when you run the page.

- Ensure that the `Page Customizable` component has only one direct child component. Adding multiple direct child components may cause the application to be unstable at runtime.

  When you add multiple direct child components at design time, only the first child component is rendered at runtime. The first child component is stretched to fit the page. All other direct child components are ignored and not rendered on the page.

- Place all components you want to be customizable within the `Page Customizable` component.

- To enable runtime editing, you *must* ensure that the `ID` attribute is defined on all components on the page. Runtime editing of components that have no `ID` value is not supported in Oracle Composer.

  If your page includes components with no `ID` value, then you may encounter problems while editing the page in Oracle Composer.
- To ensure that the `Change Mode Link` or `Change Mode Button` is displayed properly at runtime, in the Structure window, add it above the `Page Customizable` component.
- To enable View mode personalization, place a `Show Detail Frame` component within a `Panel Customizable` component.
- Use a `Show Detail Frame` to enclose a single child only. If you must enclose multiple components in a `Show Detail Frame`, then place a grouping component, such as a `Panel Group Layout` or `Panel Customizable`, within the `Show Detail Frame` component and then place the ADF Faces components or other content within this grouping component.
- Portlets need not be placed within `Show Detail Frame` components. Portlets come equipped with a header and display options that are similar to `Show Detail Frame` components.

4.2.11 How to Enable Custom Actions on Show Detail Frame Components by Using Facets

You can use the `Show Detail Frame` facets to define and display custom actions on `Show Detail Frame` components. For example, if your `Show Detail Frame` contains a list of services provided in your application, you can add a custom action, `Show Detailed Information`, which opens up a task flow containing details about the various services.

For information about the facets supported by `Show Detail Frame` components, see Section B.1.5, "Show Detail Frame Component."

Oracle JDeveloper displays all facets available to the `Show Detail Frame` component in the Structure window, however, only those that contain UI components appear activated.

To add a Show Detail Frame facet, perform the following steps:

1. Right-click a `Show Detail Frame` component in the Structure window, and select **Facets - Show Detail Frame**.
2. Click the arrow to the right of this option.
3. From the list of supported facets, select the facet you want to add.

   Note: A check mark next to a facet name means the facet is already inserted in the `Show Detail Frame` component. Selecting that facet name again would result in the facet getting deleted from the page.

   The `f:facet` element for that facet is inserted in the page.
4. Add components in the facet according to your design requirements.

   For an end-to-end example of creating and using `Show Detail Frame` facets, see Section 4.2.12, "How to Enable Custom Actions on Show Detail Frame Components By Using Facets: Example."

4.2.12 How to Enable Custom Actions on Show Detail Frame Components By Using Facets: Example

Assume a JSPX page, `Page1`, with a `Panel Customizable` component. Inside the `Panel Customizable` is a `Show Detail Frame` component (`showDetailFrame1`). Inside the `Show Detail Frame` is an ADF task flow. The `Panel Customizable` has two other `Show Detail Frame` components, one above and the other below `showDetailFrame1`. The task flow displays two `Output Text` components on the page.

You can configure an `Additional Actions` facet on the `Show Detail Frame` component to display a **Customize** action on the Actions menu along with the **Move Up** and **Move Down** actions. At runtime, the **Customize** action enables users to customize the text in the `Output Text` components. This section describes the steps you take to achieve this effect. It contains the following subsections:

- Section 4.2.12.1, "How to Create an ADF Task Flow"
- Section 4.2.12.2, "How to Include an Additional Actions Facets"
- Section 4.2.12.3, "How to Create a Redirection Page"
- Section 4.2.12.4, "How to Create Navigation Rules Between Pages"
- Section 4.2.12.5, "What Happens at Runtime"

4.2.12.1 How to Create an ADF Task Flow

To create an ADF task flow:

1. From the **File** menu, select **New**.
2. In the **New** dialog, select **JSF** under Web Tier node, and select **ADF Task Flow** under Items.
3. Click **OK**.
4. In the Create Task Flow dialog, click **OK** to create a task flow definition file by accepting the default values.
5. From the **ADF Task Flow** tag library in the Component Palette, drop two view elements, **view1** and **view2**, onto the task flow definition file.
6. Drop a **Control Flow Case** from `view1` to `view2`.
7. Click the first view element and then click the second view element to draw the control flow line.

   Name this control flow case `next`.
8. Similarly, drop a **Control Flow Case** from `view2` back to `view1` and name it `prev`.
9. Create a backing bean called `BackingBean.java` to contain values for two variables `view1` and `view2`.

   `view1` and `view2` are initialized with `initialValue1` and `initialValue2` respectively. Ensure that the code of the bean is as show in the following example:

   ```
   package view;

   public class BackingBean {
       public BackingBean() {
       }
   ```

```
    private String view2 ="initial Value1";
    private String view1 = "initial Value2";

    public void setView2(String view2) {
        this.view2 = view2;
    }

    public String getView2() {
        return view2;
    }

    public void setView1(String view1) {
        this.view1 = view1;
    }

    public String getView1() {
        return view1;
    }
}
```

10. In the task flow definition file, double-click **view1** to create the page fragment (`view1.jsff`) for that element.
11. Add a **Panel Group Layout** and two **Output Text** components to `view1.jsff`.
12. Specify the `Value` for the first `Output Text` component to be `#{backingBean.view1}`, and specify the `Value` for the second `Output Text` component to be `#{backingBean.view2}`.
13. Save `view1.jsff`, and close it.
14. In the task flow definition file, double-click **view2** to create the page fragment (`view2.jsff`) for that element.
15. Add just one **Output Text** component to `view2.jsff`, and specify `Value` to be `#{backingBean.view2}`.
16. Save `view2.jsff`, and close it.

4.2.12.2 How to Include an Additional Actions Facets

To include an Additional Actions facet:

1. Create a JSPX page, `Page1.jspx`, with a `Panel Customizable` component and a nested `Show Detail Frame` component.
2. Add two more `Show Detail Frame` components, above and below the existing `Show Detail Frame` component.

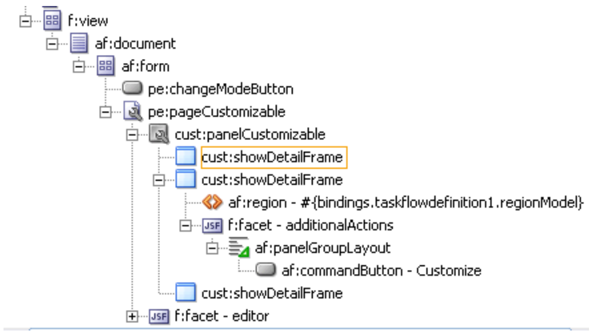
   The purpose of adding three `Show Detail Frame` components is to enable the display of `Move Up` and `Move Down` actions along with the additional action on the first `Show Detail Frame` component, `showDetailFrame1`.
3. Add the task flow definition file that you created in the previous step inside `showDetailFrame1`.
4. Right-click the first `Show Detail Frame` in the Structure window, and select **Facets - Show Detail Frame**.
5. Click the arrow to the right of this option, and from the list of supported facets, select **Additional Actions**.

   The `f:facet-additionalActions` element for that facet is inserted in the page.

6. Add a **Panel Group Layout** inside the `Additional Actions` facet, and add a **Button** component inside the `Panel Group Layout` component.
7. Set the `Text` attribute for the `Button` to `Customize`, and specify `customize` as the `Action` value.

   The page in the Structure Navigator should appear as shown in Figure 4–29.

Figure 4–29 Page1.jspx in Structure Navigator



```
f:view
  af:document
    af:form
      pe:changeModeButton
      pe:pageCustomizable
        cust:panelCustomizable
          cust:showDetailFrame
          cust:showDetailFrame
            af:region - #{bindings.taskflowdefinition1.regionModel}
            f:facet - additionalActions
              af:panelGroupLayout
                af:commandButton - Customize
          cust:showDetailFrame
        f:facet - editor
```

8. Save the page.

4.2.12.3 How to Create a Redirection Page

To create the redirection page:

1. Create a JSPX page called `Page2.jspx`, and add two **Input Text** components and a **Button** component.
2. Specify the `Value` for the first `Input Text` component to `#{backingBean.view2}`, and set the `Value` attribute for the second `Input Text` component to `#{backingBean.view1}`.

   The purpose of adding `Input Text` components with references to the backing bean is to enable the passing of a user's input to the bean so that it can be reflected in the `Output Text` components on `Page1.jspx`.
3. On the `Button` component, set the `Text` attribute to `OK` and specify `back` as the `Action` value.
4. Save the page.

   You can now enable switching between the two pages by defining navigation rules.

4.2.12.4 How to Create Navigation Rules Between Pages

To define navigation rules between the pages:

1. In the Application Navigator, under the project's WEB-INF folder, double-click the **faces-config.xml** file to open it.
2. Click the **Overview** tab to view the file in Overview mode.
3. Click the **Add** button in the Managed Bean section.
4. In the Create Managed Bean dialog, specify `backingBean` as the name.
5. For the Class Name field, click the **Browse** button adjacent to it and browse to the `BackingBean` Java class that you created earlier. Click **OK**.
6. Click **OK**.

7. From the Scope list, select **session** and click **OK**.
8. Select the **Navigation Rules** tab on the page.
9. In the From Views table, select **Page1.jspx**.
10. Under Navigation Cases, select **Page2.jspx** in the To View ID column, **customize** in the From Outcome column, and **true** in the Redirect column.
11. In the From Views table, select **Page2.jspx**.
12. Under Navigation Cases, select **Page1.jspx** in the To View ID column, **back** in the From Outcome column, and **true** in the Redirect column.

    In Source view, these entries appear as follows:

```
<managed-bean>
    <managed-bean-name>backingBean</managed-bean-name>
    <managed-bean-class>view.BackingBean</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
  </managed-bean>
  <navigation-rule>
  <from-view-id>/Page1.jspx</from-view-id>
  <navigation-case>
    <from-outcome>customize</from-outcome>
    <to-view-id>/Page2.jspx</to-view-id>
      <redirect/>
    </navigation-case>
  </navigation-rule>
  <navigation-rule>
    <from-view-id>/Page2.jspx</from-view-id>
  <navigation-case>
    <from-outcome>back</from-outcome>
    <to-view-id>/Page1.jspx</to-view-id>
      <redirect/>
    </navigation-case>
</navigation-rule>
```

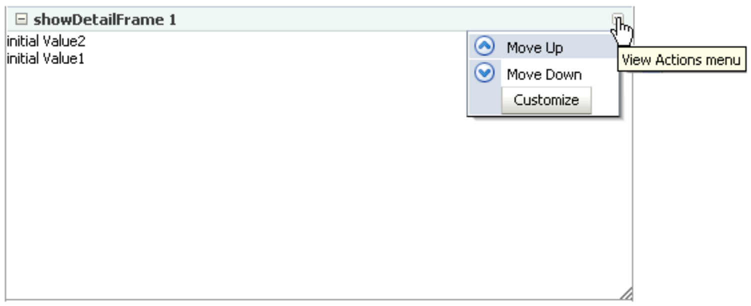
13. Save the file.

4.2.12.5 What Happens at Runtime

What Happens at Runtime

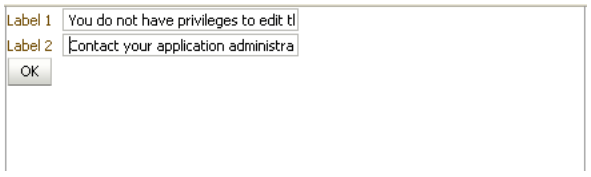
In JDeveloper, run `Page1.jspx`. The Actions menu on the `showDetailFrame1` component displays the Customize action, as shown in Figure 4–30.

Figure 4–30 Customize Action on the Actions Menu



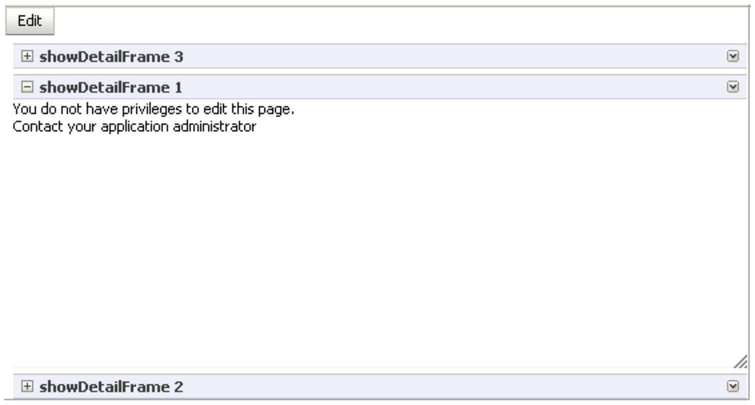
Clicking Customize takes you to `Page2.jspx` (Figure 4–32), where you can update the values for `Label1` and `Label2`.

Figure 4–31   Page with Option to Edit Text


Label 1
You do not have privileges to edit tl
Label 2
Contact your application administra
OK

Clicking **OK** takes you back to `Page1.jspx`, which reflects the recent text changes, as shown in Figure 4–32.

Figure 4–32   Page1 with Updated text



You do not have privileges to edit this page.
Contact your application administrator

4.2.13 How to Enable Custom Actions on a Show Detail Frame Enclosing a Task Flow

You can customize a task flow by including it in a `Show Detail Frame` component on which you define custom actions. Further, you can define custom actions that, when invoked at runtime, trigger the desired navigational flow. For example, you can define a custom action on a `Show Detail Frame` that specifies that the target task flow fragment opens in a separate browser window rather than inside the `Show Detail Frame`.

On a `Show Detail Frame` component, you can define custom actions at both the global and instance levels.

This section provides information about defining custom actions on a `Show Detail Frame`. It includes the following subsections:

- Section 4.2.13.1, "Defining Custom Actions at the Instance Level"
- Section 4.2.13.2, "Defining Custom Actions at the Global Level"
- Section 4.2.13.3, "Defining Custom Actions that Display Task Flow Views in a Separate Browser Window"

4.2.13.1 Defining Custom Actions at the Instance Level

Define custom actions on a particular `Show Detail Frame` component instance using the `Custom Action` component. You can find the `Custom Action` component in the Oracle Composer tag library. Custom actions are stored in the JSPX page definition file of the page that contains the `Show Detail Frame`.

To define custom actions at the instance level:

1. From the Component Palette, select **Oracle Composer**.
2. Drag a **Custom Action** and drop it on the page inside the `Show Detail Frame` component, below the `af:region` element.

   Add one `Custom Action` component for each internal task flow action you want to display as a custom action on the `Show Detail Frame` header.

   > **Note:** Add `Custom Action` components below the `af:region` element. Otherwise, you might face problems if the region is not the first child component of the `Show Detail Frame`.

3. Define attributes for the `Custom Action` by referring to Table B–9 in Section B.1, "Oracle Composer Component Properties."

   Ensure that you populate the `Action` attribute of each `Custom Action` with the correct ADFc outcomes of the associated task flow. The code should be similar to the one shown in Example 4–3.

Example 4–3 Custom Action Code

```
<cust:showDetailFrame text="showDetailFrame 1" id="sdf1">
  <af:region value="#{bindings.taskflowdefinition1.regionModel}"
             id="r1"/>
  <cust:customAction action="navigatefromview1" id="ca1"
                     location="both" icon="/Logo1.JPG"
                     text="View1 Action"
                     shortDesc="Custom View1 Action"/>
  <cust:customAction action="navigatefromview2"
                     location="both" id="ca2"
                     icon="/Logo2.JPG" text="View2 Action"
                     shortDesc="Custom View2 Action"/>
</cust:showDetailFrame>
```

> **Note:** If you define a custom action without a corresponding task flow action, then the custom action is not rendered on the `Show Detail Frame` header at runtime.

> **See Also:** "Creating ADF Task Flows" in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*

4. Save and run the page.

At runtime, when you select an action from the `Show Detail Frame`'s Actions menu, the associated control flow rule is triggered and the target task flow fragment is rendered.

4.2.13.2 Defining Custom Actions at the Global Level

Defining custom actions at the global level means making those custom actions available for all `Show Detail Frame` instances in an application. Though global-level custom actions are available for all `Show Detail Frame` components in an application, at runtime the header of any `Show Detail Frame` displays only those custom actions that correspond to the ADFc outcomes of the current view of the task flow.

Define global-level custom actions in your application's `adf-config.xml` file.

To define custom actions at the global level:

1. Open the application's `adf-config.xml` file, located in the `ADF META-INF` folder under Descriptors in the Application Resources panel.
2. Define custom actions using the `<customActions>` element with nested `<customAction>` tags for each action, as shown in Example 4–4:

Example 4–4 Custom Actions Definitions in the adf-config.xml File

```
<cust:adf-config-child
xmlns="http://xmlns.oracle.com/adf/faces/customizable/config">
  <enableSecurity value="true" />
  <customActions>
    <cust:customAction action="forward" displayName="Move Forward"
                       location="menu" rendered="true"
                       icon="/move_forward.png"/>
    <cust:customAction action="backward" tooltip="Move Backward"
                       location="chrome" rendered="true"
                       icon="/move_backward.png"/>
  </customActions>
</cust:adf-config-child>
```

Notes:

- If you implemented restrictions on the `Show Detail Frame` component's actions by performing the steps in Section 5.8.4, "Applying Action-Level Restrictions on Show Detail Component Actions," you already have a `cust:customizableComponentsSecurity` section in the `adf-config.xml` file. You can define custom actions in that section itself instead of the `cust:adf-config-child` section shown in Example 4–4.
- If you are defining an icon for the custom action, you must ensure that the image you specify is available in the project root folder.
- You can define custom actions for the internal actions defined in all task flows on your page; however, at runtime, the header of any `Show Detail Frame` displays only those custom actions that correspond to the ADFc outcomes of the current view of the task flow.

3. Save the `adf-config.xml` file.

For additional information about defining custom actions, see Section 4.2.14, "How to Enable Custom Actions On a Show Detail Frame Enclosing a Task Flow: Example."

Resolving Conflicts Between Global-Level and Instance-Level Custom Actions

Each custom action is uniquely identified by the value of its `action` attribute. If you have defined custom actions with the same `action` attribute value at the global and instance levels, then there may be conflicts in how these custom actions are invoked at runtime, depending on other attribute values. At such times, the `inheritGlobalActions` attribute of the `Show Detail Frame` defines the behavior of other custom action attributes (other than the `action` attribute) as follows:

Note: Regardless of the `inheritGlobalActions` setting (`true` or `false`) on the `Show Detail Frame` component:

- The `rendered` attribute is not inherited even if it is not specified at the instance level.
- The `location` attribute at the global or instance level should be set to the same value at both the global and instance levels.

- If `inheritGlobalActions=true`, or you have not specified a value for `inheritGlobalActions` (defaults to `false`), the behavior of custom action attributes is as follows:
  - If you defined a custom action attribute at the global and instance levels, then the attribute value specified at the instance level is used.
  - If you defined a custom action attribute only at the instance level, then that attribute value is used.
  - If you defined a custom action attribute only at the global level, then that value is ignored and the default value is used.
- If `inheritGlobalActions=true`, the behavior of custom action attributes is as follows:
  - If you defined a custom action attribute at the instance level, then its value is used regardless of whether the same attribute is specified at the global level.
  - If you defined a custom action attribute only at the global level, then that value is used.
  - If you have not defined a custom action attribute at the global or instance levels, then the attribute's default value is used.

After you have designed your application pages, you must deploy the application to the production environment. For more information, see Chapter 25, "Testing and Deploying Your WebCenter Application."

Note: Runtime customizations that you perform on the page in the development environment are not carried over when you deploy the application to a target server.

4.2.13.3 Defining Custom Actions that Display Task Flow Views in a Separate Browser Window

Custom actions typically display the target task flow views in place, inside the `Show Detail Frame` component. However, if you want to display a task flow's view in a dialog, you can define a custom action to display that task flow view in a separate browser window.

To display a task flow view in a separate browser window, the control flow rule to that view must be prefixed with `dialog:` in the task flow definition file and in the `action` attribute of the custom action corresponding to that view. The following example shows an `action` attribute definition:

```
<cust:customAction action="dialog:Next" id="ca1"
                    location="both" icon="/move_forward.png"
                    text="Next Action"
                    shortDesc="Next Action"/>
```

4.2.14 How to Enable Custom Actions On a Show Detail Frame Enclosing a Task Flow: Example

In this example, assume that your application contains a task flow, `customactions`, residing inside a `Show Detail Frame`. The task flow includes three view elements, `view_gadget`, `edit_settings`, and `about_gadget`, and three associated control flow rules, `ViewGadget`, `EditSettings`, and `AboutGadget`. Your object is to define custom actions such that the control flow rules are available as actions on the Actions menu of the `Show Detail Frame` component.

In this example, the control flow rules are added such that users can navigate back and forth between the three views. Each view element has an associated page fragment of the same name:

- The `view_gadget.jsff` fragment has a `Panel Stretch Layout` component. The center facet of this component is populated with an `Active Output Text` component whose `Value` attribute is set to `View Gadget`.
- The `edit_settings.jsff` fragment has a `Panel Stretch Layout` component. The center facet of this component is populated with an `Active Output Text` component whose `Value` attribute is set to `Edit Gadget Settings`.
- The `about_gadget.jsff` fragment has a `Panel Stretch Layout` component. The center facet of this component is populated with an `Active Output Text` component whose `Value` attribute is set to `About This Gadget`.

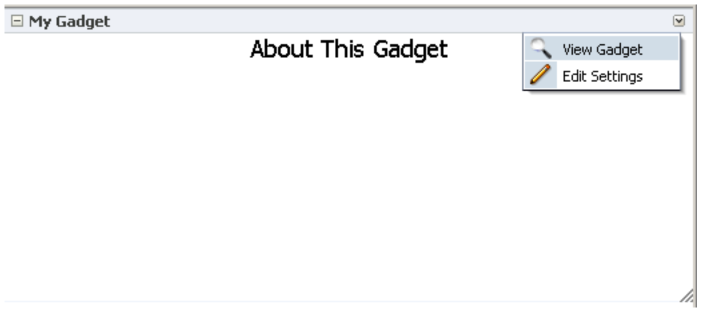
To enable custom actions on the task flow:

1. Place the `customactions` task flow inside a `Show Detail Frame` component on a customizable page, `MyPage.jspx`.

   For information about creating a customizable page, see Section 4.2.1, "How to Create a Customizable Page."
2. Add a `Custom Action` component from the Oracle Composer tag library as a child of the `Show Detail Frame` component, and set the `Action` and `Text` attributes to `ViewGadget` and `View Gadget` respectively.
3. Add two more `Custom Action` components to the `Show Detail Frame`:
   - Set the `Action` and `Text` attributes for the first component to `EditSettings` and `Edit Settings` respectively.
   - Set the `Action` and `Text` attributes for the second component to `AboutGadget` and `About Gadget` respectively.
4. Save and run `MyPage.jspx`.

   The `view_gadget` page fragment is rendered in the `Show Detail Frame` component (named **My Gadget**) on the page. The Actions menu displays the **About Gadget** and **Edit Settings** options. Click **About Gadget** to navigate to the `about_gadget` fragment. Note that the Actions menu now displays the navigation rules for the other two fragments (Figure 4–33).

Figure 4–33 Custom Actions on a Show Detail Frame Enclosing a Task Flow



To see this in action, look at the sample application, `CustomActions.jws`, on the Oracle WebCenter Suite 11g Demonstrations and Samples page on the Oracle Technology Network (OTN):

`http://webcenter.oracle.com`

4.2.15 How to Create Event-Enabled Task Flows

Oracle Composer provides a means of contextually wiring task flow events. You can wire a contextual event to an action handler to enable the passing of values from a producer component to a consumer component when the event is triggered on the producer.

For events to be available at runtime, event capability must be included in the task flow at design time. When you add event-enabled task flows to your customizable page, each task flow's Component Properties dialog includes an Events tab, where much of the wiring activity takes place. For information about including event capabilities, see "Creating Contextual Events" in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

4.2.16 How to Apply Styles to Components

You can change the look and feel of Oracle Composer components by applying different styles to the component header and content using one of the following methods:

- Build a skin using style selectors, and apply the skin to a WebCenter application. For more information about style selectors and skins, see Chapter 19 "Customizing the Appearance Using Styles and Skins" in *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*.
- Use JDeveloper style properties to specify style information through the Property Inspector. For more information, see Understanding contentStyle and inlineStyle Properties.

Notes: Using JDeveloper style properties overrides the style information from the skin CSS. However, when you define a style property using JDeveloper, this style overrides styles for the selected component only—child components continue to use the styles specified in the skin.

You can adjust the look and feel of `Page Customizable`, `Panel Customizable`, `Layout Customizable`, and `Show Detail Frame` components at design time by changing the style-related properties `inlineStyle` and `styleClass`.

`Show Detail Frame` components have another associated style property, `contentStyle`, which defines the style for the content inside the component. The `styleClass` property enables you to select an extra style from the skin, whereas the `inlineStyle` and `contentStyle` properties override the comparable styles specified in the application skin for that particular instance of the component.

The `inlineStyle` property overrides `styleClass`. Additionally, properties set on a component instance affect only that instance of the component. Other component instances in the application are not affected.

Note: The `background` property is also useful in adjusting the look and feel of `Show Detail Frame` components. Unlike `inlineStyle`, `contentStyle`, and `styleClass` properties, the `background` property works with skins.

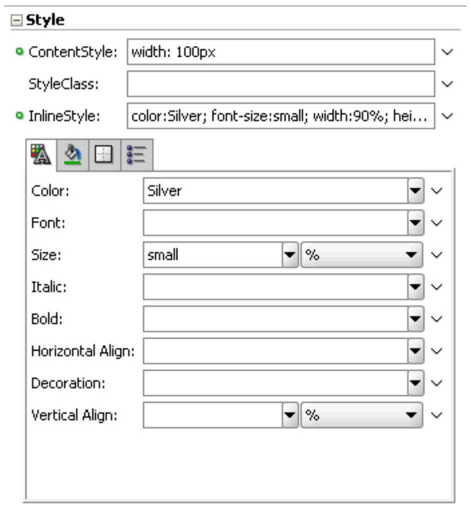
Understanding contentStyle and inlineStyle Properties

The style properties `inlineStyle` and `contentStyle` are alike in the types of attributes they support. They differ in their range of influence. While `inlineStyle` provides style information for the *entire component*, `contentStyle` provides style information only for *component content*. The `contentStyle` property is available to `Show Detail Frame` components but not to `Panel Customizable` components.

The `inlineStyle` property applies CSS to the root of the component, that is, the topmost DOM element. It does not override styles on child elements that are picking up color, font, and so on from a skin. For example, if a component header is skinned, then setting `inlineStyle` does not affect the component header. The `contentStyle` applies CSS to the DOM element that surrounds the *content* part of the component. In a `Show Detail Frame`, content refers to the area below the header.

On component content, the value specified for `contentStyle` takes precedence over the value specified for `inlineStyle`. Additionally, `contentStyle` on a component instance takes precedence over both `inlineStyle` and the `contentStyle` values of a parent component (such as a portlet nested in a `Panel Customizable` component).

Figure 4–34 Defining Styles for contentStyle and inlineStyle in the Property Inspector



ContentStyle:	width: 100px
StyleClass:	
InlineStyle:	color:Silver; font-size:small; width:90%; hei...
Color:	Silver
Font:	
Size:	small %
Italic:	
Bold:	
Horizontal Align:	
Decoration:	
Vertical Align:	%

4.3 Designing Editable Pages Using Oracle Composer Components: Example

In this example, assume you want to create a page that is customizable at runtime. The page is named `MyPage.jspx` in a WebCenter application named `MyWebCenterApp`.

To create a customizable page:

1. Create a WebCenter application named `MyWebCenterApp` by performing the steps in Section 3.2, "Creating a WebCenter Application."
2. Create a JSPX page named `MyPage.jspx` by performing the steps in Section 3.3, "Creating WebCenter Application-Enabled Pages."
3. Add a `Panel Stretch Layout` to `MyPage.jspx`.

   Note: The `Panel Stretch Layout` component stretches the child in the center facet to fill all the available space in the browser. This is true even if you resize the browser. Therefore, by placing your components within this child component, you can ensure that the customizable portion of your page occupies the complete browser area.

4. Add a `Page Customizable` to the center facet by following the steps in Section 4.2.2, "How to Enable Runtime Customization Using a Page Customizable."
5. Set the border color of the `Page Customizable` to `blue` to differentiate the editable area from other noneditable areas.

   Under the Style category in the Property Inspector for the `Page Customizable`, click the Box tab and set the `Border Color` attribute to `Blue`.
6. Set the border color of child `Panel Customizable` to `red`.

   Under the Style category in the Property Inspector for the `Panel Customizable`, click the Box tab and set the `Border Color` attribute to `Red`.

7. Add a `Change Mode Link` to the top facet of the `Panel Stretch Layout` by performing the steps in Section 4.2.3, "How to Enable Switching Between Page Modes Using a Change Mode Link or Change Mode Button."

8. Inside the `Panel Customizable` that is a direct child of the `Page Customizable`, add a `Layout Customizable` component by following the steps in Section 4.2.5, "How to Enable Layout Customization for a Page Using a Layout Customizable."

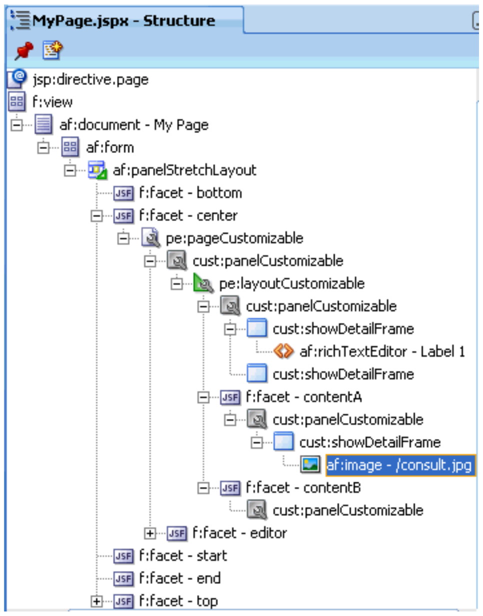
9. Add `Show Detail Frame` components inside each `Panel Customizable` on the page by performing the steps in Section 4.2.6, "How to Enable Component Customization Using Show Detail Frame Components."

10. For the sake of this example, add a `Rich Text Editor` and an `Image` component inside two of the `Show Detail Frame` components nested in the `Layout Customizable`.

    Drag and drop each of these components from the **ADF Faces** tag library to the required location on the page.

    The hierarchy of components on the page is as shown in Figure 4–35.

Figure 4–35 Component Hierarchy in MyPage.JSPX



MyPage.jspx - Structure

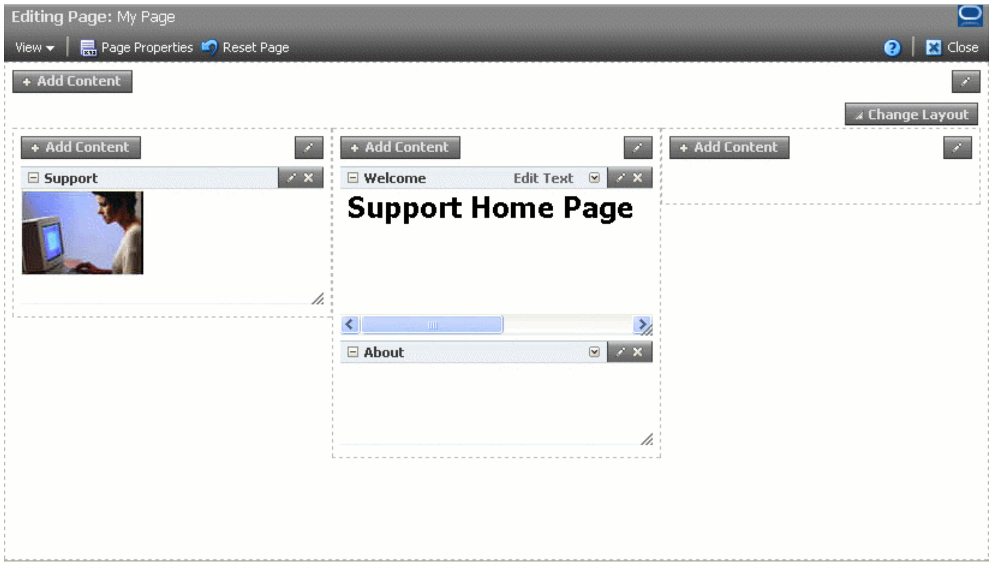
```
jsp:directive.page
f:view
  af:document - My Page
    af:form
      af:panelStretchLayout
        f:facet - bottom
        f:facet - center
          pe:pageCustomizable
            cust:panelCustomizable
              pe:layoutCustomizable
                cust:panelCustomizable
                  cust:showDetailFrame
                    af:richTextEditor - Label 1
                  cust:showDetailFrame
                f:facet - contentA
                  cust:panelCustomizable
                    cust:showDetailFrame
                      af:image - /consult.jpg
                f:facet - contentB
                  cust:panelCustomizable
            f:facet - editor
        f:facet - start
        f:facet - end
        f:facet - top
```

11. Run `MyPage.jspx`.

    The page opens in View mode. Click the **Edit** link on the page to enter Edit mode. The page opens in Oracle Composer. In Oracle Composer, you can perform all editing tasks described in Section 4.1.3, "Editing Capabilities in Edit Mode."

    Figure 4–36 shows how the page appears in Edit mode at runtime.

Figure 4–36 MyPage Opened in Oracle Composer



To use this sample page in other examples in this guide, configure ADF security on the application.

To configure security in your application:

1. Configure ADF Security with form-based authentication, generating a default login page.

   For more information, see Section 3.5.1, "How to Configure ADF Security."

2. Create three users `ahunold`, `sking`, and `ngreenbe`.

   For the detailed procedure, see Section 3.5.2, "How to Create an Application Role."

When you run `MyPage.jspx`, a login screen is displayed. You can log in with any of the three user names that you created.

4.4 Populating Pages with Content

After you create an editable page with the required Oracle Composer components, you can populate the page with content just like a regular JSPX page. However, there are certain limitations and recommendations that you must be aware of when adding content to your Oracle Composer-enabled page.

Populating editable pages at design time is like populating any other ADF Faces page. You can drag and drop components from different areas of the IDE onto the page. You can add components like portlets, task flows, and ADF Faces components.

When you drag and drop a component anywhere inside a `Page Customizable` component, the `Id` attribute is set to a unique value. The `Id` attribute is required for editing a component and persisting its changed state. When you add a component to a page at runtime, the `Id` attribute is set automatically.

Consider the following when adding content to your editable page:

- Components placed inside a `Panel Customizable` component that is nested in a `Page Customizable` component can be edited at runtime.

- When adding an ADF Faces component within the `Page Customizable` component, ensure that you wrap it in a `Show Detail Frame` component. You can then perform customizations on the component at runtime.
- You can include any Oracle ADF Faces component or task flow as child component of a `Show Detail Frame` component. However, portlets contain headers similar to those provided by `Show Detail Frame` components and can be added to `Panel Customizable` components directly. There are no additional benefits to including portlets in `Show Detail Frame` components.
- The `clientComponent` attribute is not applicable to `Panel Customizable` and `Show Detail Frame` components within the `Page Customizable` component. In these components, the `clientComponent` attribute is inherently defined.
- If your page has the ADF Faces components `Output Text` and `Output Formatted` nested inside a `Page Customizable` component, then ensure that you set the `clientComponent` attribute value. If this attribute value is not set, then you may encounter errors while trying to move or rearrange components on the page at runtime.
- To consume portlets in your editable page, you must first register the portlet producers with your application. See Chapter 9, "Consuming Portlets" for details.

5 Extending Runtime Editing Capabilities Using Oracle Composer

Oracle Composer provides an easy-to-use, declarative, and programmable extensibility mechanism for customizing runtime editing to fit your application requirements. This chapter describes how to extend Oracle Composer runtime editing capabilities to suit your business needs and enhance the end-user customization experience. It contains the following sections:

- Section 5.1, "Overview of Extensibility Options"
- Section 5.2, "Creating Oracle Composer Add-Ons"
- Section 5.3, "Creating Custom Property Panels"
- Section 5.4, "Configuring Event Handlers for Oracle Composer UI Events"
- Section 5.5, "Defining Property Filters"
- Section 5.6, "Performing MDS-Specific Configurations"
- Section 5.7, "Using Oracle Composer Sandbox"
- Section 5.8, "Overriding Default Security Behavior of Oracle Composer Components"
- Section 5.9, "Disabling Source View for the Application"
- Section 5.10, "Troubleshooting Oracle Composer Problems"

5.1 Overview of Extensibility Options

Oracle Composer provides a framework on which to build customizable application pages. In addition to its default capabilities, you can extend the Oracle Composer framework to augment the runtime capabilities available to end users.

This section describes the options available for extending Oracle Composer runtime capabilities declaratively. It contains the following subsections:

- Section 5.1.1, "Oracle Composer Add-Ons"
- Section 5.1.2, "Oracle Composer Custom Property Panels"
- Section 5.1.3, "Oracle Composer UI Events"
- Section 5.1.4, "Property Filters"
- Section 5.1.5, "MDS Customization Classes"
- Section 5.1.6, "Oracle Composer Sandbox"

- Section 5.1.7, "Configuration Files"

5.1.1 Oracle Composer Add-Ons

Typically, add-ons are custom task flows that are rendered as buttons on the Oracle Composer toolbar. The Page Properties and Reset Page buttons are examples of add-ons on the Oracle Composer toolbar. Click these buttons to display panels for editing page properties and resetting page customizations. You can create add-ons that appear along with or in place of the Page Properties and Reset Page add-ons. For example, you can create a Revision History add-on, which displays a Revisions button on the toolbar. Clicking this button displays the page revision history.

The process of creating Oracle Composer add-ons includes creating them as task flows, packaging them into JAR files, and defining them in the Oracle Composer extension file. For more information, see Section 5.2, "Creating Oracle Composer Add-Ons."

5.1.2 Oracle Composer Custom Property Panels

You can create and register custom property panels with Oracle Composer and include them as additional tabs in the Component Properties dialog box. The default Component Properties dialog in Oracle Composer is analogous to the Oracle JDeveloper Property Inspector. The Component Properties dialog displays categories of attributes on different tabs. You can create and register custom property panels for a component, populate them with categories of component properties, and render them as tabs along with the default tabs in the Component Properties dialog.

The process of creating custom property panels includes creating them as task flows, packaging them into JAR files, and defining them in the Oracle Composer extension file. For more information, see Section 5.3, "Creating Custom Property Panels."

5.1.3 Oracle Composer UI Events

Oracle Composer provides an intuitive user interface for editing pages at runtime. This includes such UI components as the Save, Close, and Delete buttons. Sometimes it is necessary to provide additional user interactions that augment Oracle Composer's innate capabilities. For example, when a user clicks Save, in addition to the save operation that Oracle Composer provides by default, you might want to configure the application to perform additional workflows. You can accomplish this with event handlers. An event handler is the code that is called back by Oracle Composer when a particular composer event is invoked. For more information, see Section 5.4, "Configuring Event Handlers for Oracle Composer UI Events."

5.1.4 Property Filters

The Component Properties dialog displays properties with editable values. Depending on what properties you want to expose, you can define filters declaratively to hide certain properties, show hidden properties, or filter properties dynamically using Expression Language (EL). For more information, see Section 5.5, "Defining Property Filters."

5.1.5 MDS Customization Classes

You can define criteria for applying customizations to your application's metadata objects. For example, you can define customizations that come into play depending on a user's assigned permissions, an application's deployment location (also known as *localization*), or a specific industry domain. Each criterion denotes a customization layer and is depicted using a `CustomizationClass`. A `CustomizationClass` is the interface MDS uses to identify the customization layer to be overlaid on the base definition. For an example, see Section 5.6.1.3, "How to Create a Custom UserCC Tip Layer."

When you implement a `CustomizationClass`, you must register it with the MDS. The MDS provides a means of associating a list of `CustomizationClass` types with a single `MetadataObject`. This is called the *fine-grained* association. The MDS also provides the means of associating a list of `CustomizationClass` types with a set of `MetadataObjects`. This is called the *coarse-grained* association. For information about the MDS-specific configurations, Section 5.6, "Performing MDS-Specific Configurations."

5.1.6 Oracle Composer Sandbox

Typically, in a custom WebCenter application that uses a file system to store metadata, runtime customizations are saved immediately in the *`application-root`*`/mds` directory. Changes made in both View and Edit modes are saved in this way. However, in certain circumstances, you might first want to apply the customizations in your view and evaluate whether to keep or cancel the changes before actually saving them to the back end.

Customizations can be stored in a file system repository or database repository. If you are using a database repository to store customizations, then you can configure Oracle Composer to create a *sandbox*. A sandbox is a temporary storage area for saving runtime page customizations before they are either saved to the back end or canceled.

Customizations made in View mode are saved to the back end immediately. Because such changes are available only to the user performing the customizations, there is no particular value in reviewing such changes before saving.

Because Edit mode customizations are available to all users who access the page, you might want to enable a sandbox so that you can experiment with page customizations and assess them before committing them. If you enable a sandbox for the application, a Save button is displayed on the Oracle Composer toolbar in both Design view and Source view of the page.

For information about enabling and using the sandbox, see Section 5.7, "Using Oracle Composer Sandbox."

5.1.7 Configuration Files

Before you start with the extensibility configurations described in this chapter, there are two important configuration files that you must know about. Most of the extensions discussed in this chapter are defined in these files:

- Oracle Composer extension file (`pe_xet.xml`)

  The `pe_ext.xml` file is not available in your application by default. You must create it the first time you perform such tasks as including add-ons, property panels, or event handlers. Create this file in the `META-INF` directory under the project's Web context root. In addition to your application having a `pe_ext.xml` file, each JAR that your application includes on its classpath can also have a `pe_ext.xml` file. All of these JARs are searched on application startup. Every JAR with a `pe_ext.xml` in its `META-INF` folder is processed, and the Composer extensions are loaded and combined. For information about the different elements you can define in `pe_ext.xml` to extend Oracle Composer capabilities, see Section B.2.1, "pe_ext.xml."

- Application's `adf-config.xml` file

  When you create your application, the `adf-config.xml` file is created automatically. The adf-config.xml file is located in the `ADF META-INF` folder under `Descriptors` in the Application Resources panel. When you perform such tasks as registering new add-ons and custom property panels, or creating customization layers, you must include the required entries in the `adf-config.xml` file.

  For information about the Oracle Composer-specific configurations you can make in `adf-config.xml`, see Section B.2.2, "adf-config.xml."

5.2 Creating Oracle Composer Add-Ons

Oracle Composer provides the following default capabilities for runtime editing:

- Page properties editor

  This dialog opens when users click the Page Properties button on the Oracle Composer toolbar. The Page Properties dialog displays the current page's properties and enables users to modify property values.

- Reset page option

  The Reset Page dialog box opens when users click the Reset Page button on the Oracle Composer toolbar. The Reset Page dialog enables users to remove all edits, regardless of when they were made, and reset the page to its original, out-of-the-box state.

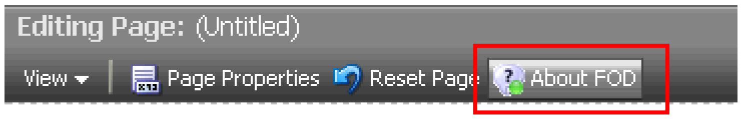
In addition to these, you can register new add-ons in Oracle Composer. For example, you can create a custom task flow with information about the application and register it as an add-on with the title `About`. Your add-on displays as an About button in the Oracle Composer toolbar. Clicking the About button opens a panel that contains the information you specified.

5.2.1 How to Create and Register Add-Ons

You can create and register custom task flows that can be invoked from buttons on the Oracle Composer toolbar. All registered add-ons have an associated button that displays on the toolbar.

This section steps through the procedure of creating an add-on and registering it with Oracle Composer. It includes an example that demonstrates how to create an add-on that displays information about the Fusion Order Demo (FOD) application. The example add-on renders an **About** button on the Oracle Composer toolbar (Figure 5–1) that users can click to invoke a task flow that contains information about FOD.

Figure 5–1 About FOD Button in Oracle Composer Toolbar



This section contains the following subsections:

- Section 5.2.1.1, "Creating an Add-On"
- Section 5.2.1.2, "Registering Add-Ons with Oracle Composer"
- Section 5.2.1.3, "Registering Add-Ons in adf-config.xml"

5.2.1.1 Creating an Add-On

Oracle Composer add-ons are task flows you create using JSPX pages or page fragments. To complete the process, you package the task flows into JAR files and register them with Oracle Composer and the `adf-config.xml` file.

To create an add-on:

1. In your WebCenter application project, create a JSFF file called `custompanelview.jsff`:
   a. From the **File** menu, select **New**.
   b. In the New Gallery dialog box, expand **Web Tier**, select **JSF**, then **JSF Page** or **JSF Fragment**.
   c. Click **OK**.
2. Design the fragment by adding the code shown in Example 5–1.

Example 5–1 Sample code in the JSFF Fragment

```
<?xml version='1.0' encoding='UTF-8'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich">
  <af:panelGroupLayout id="pnlgrp1" layout="vertical" halign="center">
    <af:spacer id="sp1" width="10" height="20" />
    <af:image id="customimage"
              source="/images/FusionOrderDemoLogo.jpg"/>
    <af:spacer id="sp2" width="10" height="20"/>
    <af:outputText id="actoutput1" value="Fusion Order Demo (FOD) is a sample
shopping cart application based on Oracle ADF Framework.
It also uses the Webcenter Framework to enable Collaboration,Customization and
Personalization features."
                   inlineStyle="font-size:small;"/>
    <af:spacer id="sp3" width="10" height="20"/>
    <af:outputText id="actoutput2" value="Build : 11.1.1.1"
                   inlineStyle="font-size:medium; font-weight:bolder;
                   text-decoration:none; font-style:normal;
                   background-color:White; color:Blue; font-family:garamond;"/>
  </af:panelGroupLayout>
</jsp:root>
```

3. Create a task flow definition called `custom-panel-task-flow`:
   a. From the **File** menu, choose **New**.
   b. In the New Gallery dialog box, expand **Web Tier**, select **JSF**, then **ADF Task Flow**.
   c. Click **OK**.
4. Drop the `custompanelview.jsff` fragment that you created onto the task flow definition.

   See Also: "Getting Started with ADF Task Flows" in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework* for information about creating task flows.

5. Save the task flow definition file.

To package the task flow in an ADF library:

1. Create a deployment profile for the task flow:

   a. Right-click **ViewController** and choose **New**.
   b. In the New Gallery, expand **General**, select **Deployment Profile**, and then **ADF Library JAR File**, and click **OK**.
   c. In the Create Deployment Profile -- ADF Library JAR File dialog, enter a name for your deployment profile and click **OK**.
   d. In the ADF Library JAR Deployment Profile Properties dialog, click **OK**.
   e. In the Project Properties dialog, click **OK**.
2. In the Application Navigator, right-click the project folder, choose **Deploy**, *deployment profile name*, **to**, and then choose **to ADF Library JAR file**. This creates a **deploy** folder including the JAR file, in the project folder located at `<Application_Root>\ViewController\deploy\`.

5.2.1.2 Registering Add-Ons with Oracle Composer

After you create the task flows, you must register them with Oracle Composer so that they are displayed on the Oracle Composer toolbar along with the default options.

To register an add-on with Oracle Composer:

1. If it does not already exist, create the Oracle Composer extension file, `pe_ext.xml` in the `META-INF` directory under the project's Web context root:
   a. From the **File** menu, select **New**.
   b. In the New Gallery dialog box, expand **General**, select **XML**, then **XML Document**.
   c. Click **OK**.

   Name the file `pe_ext.xml`.
2. Add one `<panel>` element for each task flow that you want to register as an add-on.

   Any number of panels can be declared under the `<panels>` element in the extension file.

   Example 5–2 shows the code of the extension file with a `<panel>` entry.

Example 5–2 Oracle Composer Extension File

```
<?xml version="1.0" encoding="US-ASCII" ?>
<pe-extension xmlns="http://xmlns.oracle.com/adf/pageeditor/extension">
  <addon-config>
    <panels>
      <panel name="oracle.fod.custom.panel" title="About FOD"

icon="http://myforums.oracle.com/jive3/images/question-pts-available-16x16.gif"

taskflow-id="/WEB-INF/custom-panel-task-flow.xml#custom-panel-task-flow" />
    </panels>
  </addon-config>
</pe-extension>
```

For more information about the `addon-config` and other nested elements, see Section B.2.1.1, "addon-config."

5.2.1.3 Registering Add-Ons in adf-config.xml

To register an add-on, you must add a reference to it in the application's `adf-config.xml` file. Add the `addon-panels` entry to define new add-ons.

To register add-ons in adf-config.xml:

1. Open the application's `adf-config.xml` file, located in the `ADF META-INF` folder under `Descriptors` in the Application Resources panel.
2. Add the following namespace within the `adf-config` element in the file:

   ```
   xmlns:pe="http://xmlns.oracle.com/adf/pageeditor/config"
   ```

3. Add a `<page-editor-config>` entry with an `<addon-panels>` entry.

   Within `page-editor-config`, add `<addon-panel>` entries for the default options and the new panels, as shown in Example 5–3.

   You must include entries for the default add-ons along with those for new panels. Without these entries, the default add-ons are not displayed in Oracle Composer.

   The `name` attribute must contain the name you used to register the panel in the Oracle Composer extension file.

Example 5–3 New Add-Ons Referenced in adf-config.xml

```
<pe:page-editor-config>
  <addon-panels>
    <!-- Page Properties add-on -->
    <addon-panel name="oracle.adf.pageeditor.addonpanels.page-settings" />

    <!-- Page Reset add-on -->
    <addon-panel name="oracle.adf.pageeditor.addonpanels.page-reset" />

    <addon-panel name="oracle.fod.custom.panel" />

  </addon-panels>
</pe:page-editor-config>
```

Note: If you do not specify any `<addon-panel>` entries under `<addon-panels>`, then only the default options are displayed in Oracle Composer.

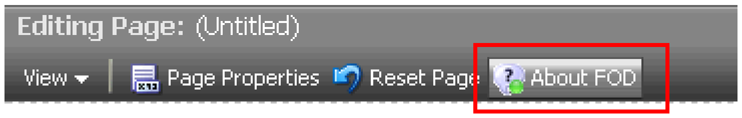
For information about the Oracle Composer-specific configurations you can make in `adf-config.xml`, see Section B.2.2, "adf-config.xml."

5.2.2 What Happens at Runtime

Custom add-ons that you register with Oracle Composer are rendered on the Oracle Composer toolbar along with the default add-ons.

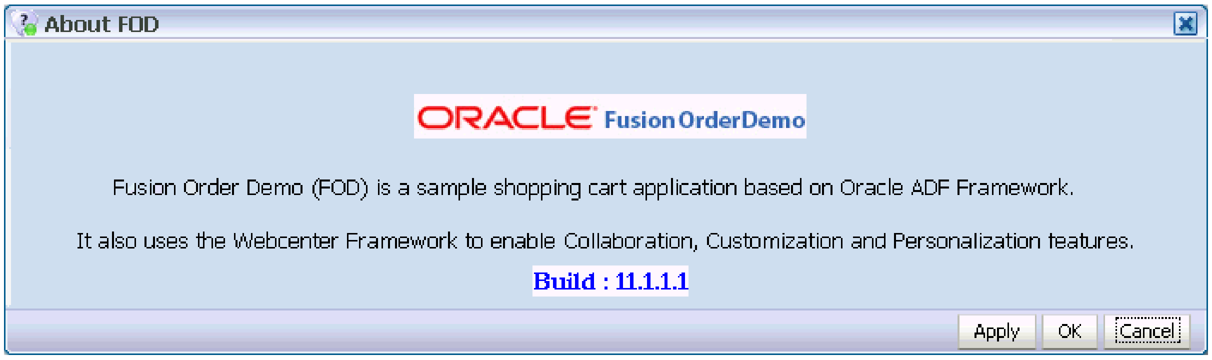
In the example, an About FOD button is rendered on the Oracle Composer toolbar (Figure 5–2).

Figure 5–2 About FOD Button in Oracle Composer Toolbar



Clicking this button displays the About FOD task flow (Figure 5–3).

Figure 5–3 About FOD Task Flow



Fusion Order Demo (FOD) is a sample shopping cart application based on Oracle ADF Framework.

It also uses the Webcenter Framework to enable Collaboration, Customization and Personalization features.

Build : 11.1.1.1

You can look at these catalog definitions in the `StoreFrontModule` in the Fusion Order Demo application. For information about this application, see Chapter 2, "Introduction to the WebCenter Sample Application."

5.2.3 How to Exclude Oracle Composer Default Add-Ons

You can choose to show or hide the Page Properties and Reset Page buttons in Oracle Composer. Not all Oracle Composer components have the same flexibility. For example, you cannot select to hide the Catalog or the Component Properties dialogs. To display only the Page Properties or Reset Page button, you can selectively include the button's associated add-on in the `adf-config.xml` file. Example 5–4 shows the entries you can add for the two built-in add-ons.

Example 5–4 Default Built-In Options

```
<!-- Page Properties add-on -->
<addon-panel name="oracle.adf.pageeditor.addonpanels.page-settings" />

<!-- Page Reset add-on -->
<addon-panel name="oracle.adf.pageeditor.addonpanels.page-reset" />
```

In Example 5–5, the Page Properties add-on entry is removed and a new add-on is included.

Example 5–5 The adf-config.xml File with Only One Built-In Option

```
<addon-panels>
  <!-- Page Reset add-on -->
  <addon-panel name="oracle.adf.pageeditor.addonpanels.page-reset" />

  <addon-panel name="oracle.fod.custom.panel" />

</addon-panels>
```

The excluded default add-on (Page Properties) is not rendered on the Oracle Composer toolbar. The included add-ons are rendered on the toolbar in the order in which they are defined in the `adf-config.xml` file.

5.2.4 How to Selectively Render Add-Ons

Depending on your business requirement, you may need to selectively hide the add-ons available to different users in Oracle Composer. To hide an add-on, you must set the `rendered` attribute for the panel to `false` using a constant value or an EL expression, as shown in Example 5–6.

Example 5–6 rendered Attribute Setting in the adf-config.xml File

```
<addon-panels>
  <addon-panel name="oracle.fod.custom.panel"
              rendered="#{bean.showNotification}" />
  . . .
</addon-panels>
```

In this example, at runtime, a backing bean's `isShowNotification()` method is called. It returns either `true` or `false`, depending on whether the panel is to be rendered.

If the returned value is `false`, Oracle Composer does not display the add-on. If the returned value is `true`, Oracle Composer displays a button for the panel.

Note: The default value for `rendered` is `true`. That is, if you do not specify a `rendered` attribute for a panel, the panel is always rendered.

5.3 Creating Custom Property Panels

When a user clicks the Edit icon on a component, its properties are shown in the Component Properties dialog. The Component Properties dialog provides a series of tabs that group related attributes together. The attributes have associated, typically editable values that control a component's visual and style properties. Parameters and Events tabs provide a means of creating or consuming component parameters and events.

Similarly, when a user clicks the Page Properties button, a Page Properties dialog opens with its own series of tabs. These contain display-related properties, page parameters, and security settings.

You can create and register custom property panels to render in place of or in addition to the panels displayed in the Component Properties or Page Properties dialog. For example, you can develop a custom property panel for an `Image` component to expose a picker for an image's `Source` property.

This section describes how to create custom property panels and how to exclude, override, and selectively render default property panels. It contains the following subsections:

- Section 5.3.1, "How to Create and Register Custom Property Panels"
- Section 5.3.2, "What Happens at Runtime"
- Section 5.3.3, "How to Exclude Default Property Panels"
- Section 5.3.4, "How to Override Default Property Panels"

- Section 5.3.5, "How to Selectively Render Property Panels"

5.3.1 How to Create and Register Custom Property Panels

Creating a custom property panel is similar to creating an add-on. That is, you create custom property panels as task flows and register them in the Oracle Composer extension file. You can configure a custom property panel to render in the dialog at all times or only when a particular component or task flow is selected for editing.

This section describes how to create and register a custom property panel. It contains the following subsections:

- Section 5.3.1.1, "Creating a Custom Property Panel"
- Section 5.3.1.2, "Registering a Custom Property Panel for a Component"
- Section 5.3.1.3, "Registering a Custom Property Panel for a Task Flow"

5.3.1.1 Creating a Custom Property Panel

Property panels provide a means of editing page or component properties. For example, the end user may click the Edit icon on a selected portlet and then modify its parameter values and change its visual attributes in the resulting properties dialog.

Oracle Composer enables you to associate property panels with components and task flows. When a user clicks the Edit icon on the component or task flow, Oracle Composer opens the Component Properties dialog and displays the custom property panels you associated with the object along with the default property panels. If you do not associate a property panel with a particular component or task flow, then that panel is rendered for all pages, components and task flows in the Component Properties and Page Properties dialogs.

The steps for creating a custom property panel and declaring it in the Oracle Composer extension file are similar to those for creating and declaring Oracle Composer add-ons. For information about these tasks, see Section 5.2.1.1, "Creating an Add-On" and Section 5.2.1.2, "Registering Add-Ons with Oracle Composer."

5.3.1.2 Registering a Custom Property Panel for a Component

After declaring a custom property panel in the extension file, you must also register it in the extension file and associate it with a component, if required. Registering a custom property panel ensures that the panel displays automatically in the Oracle Composer Component Properties and Page Properties dialog.

Use the `property-panels` element to register the custom property panel in the `pe_ext.xml` file and to associate the custom panel with a component.

To register a property panel for a component:

1. Add a `<property-panels>` element in the `pe_ext.xml` file.

   For information about creating the extension file, see Section 5.2.1.2, "Registering Add-Ons with Oracle Composer."

2. Add a `property-panel` declaration within the `<property-panels>` element.

   You can have multiple `property-panel` entries.

3. Within the `property-panel` element, add a `component` element to specify the runtime class name of the component (optional) and a `panel` element to specify the name you used to declare the panel in the `addon-config` section of the file.

Example 5–7 shows a custom property panel that is associated with a `Command Button` component by specifying the component's fully qualified class name. For information about Oracle ADF components and their runtime classes, see *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

Example 5–7 Code to Register a Property Panel for a Component

```
<property-panels>
  <property-panel name="cmdbtn">
    <component>oracle.rich.CommandButton</component>
    <panel name="prop.panel.cmdbtn" />
    <panel name="prop.panel.generic" />
  </property-panel>
</property-panels>
```

Note: When registering a property panel, if you do not associate it with a component or task flow, then the registered panel is rendered for all pages, task flows, and components in the Component Properties and Page Properties dialogs.

You can register multiple `panel` elements within a `property-panel` element. For more information about the `property-panels` element and its nested elements, see Section B.2.1.2, "property-panels."

5.3.1.3 Registering a Custom Property Panel for a Task Flow

You can define custom property panels for a task flow by registering it with the Oracle Composer extension file. Example 5–8 shows the sample code used to register a custom property panel for a task flow.

A custom property panel registered for a specific task flow appears only when its associated task flow is selected. Otherwise, default property panels appear.

Note: Use task flow-specific custom property panels only to customize task flow parameters or any other aspect of the task flow's working. Note that a custom property panel does not function if its associated task flow is rendered using the Oracle JSF Portlet Bridge.

To register a property panel for a task flow:

1. Add a `<property-panels>` element in the `pe_ext.xml` file.

   For information about creating the extension file, see Section 5.2.1.2, "Registering Add-Ons with Oracle Composer."

2. Add a `property-panel` declaration within this.

   You can have multiple `property-panel` entries.

3. Add`taskflow-id` and `panel` elements within the `property-panel` element.

   Add the `taskflow-id` element to specify the task flow name. Add the `panel` element to specify the name you used to declare the property panel in the `addon-config` section of the file.

   Example 5–8 shows a custom property panel associated with a `dashboard` task flow.

Example 5–8 Code to Register a Property Panel for a Task Flow Instance

```
<property-panels>
  <property-panel name="dashboard">
    <taskflow-id>/WEB-INF/dashboard-taskflow#prop-panel</taskflow-id>
    <panel name="dashboard.prop-panel" />
  </property-panel>
</property-panels>
```

Note: When registering a property panel, if you do not associate it with a component or task flow, then the registered panel is rendered for all pages, task flows, and components in the Component Properties and Page Properties dialogs.

You can register multiple `panel` elements within a `property-panel` element. For more information about the `property-panels` element and its nested elements, see Section B.2.1.2, "property-panels."

5.3.2 What Happens at Runtime

If you associated the custom property panel with a specific component or task flow, at runtime the panel renders as a tab in the Component Properties dialog invoked from the specified component or task flow. If you did not associate the custom property panel with a specific component or task flow, at runtime the custom property panel renders as a tab in both the Page Properties and Component Properties dialogs for all pages, components, and task flows.

5.3.3 How to Exclude Default Property Panels

You can use the `rendered` attribute to show or hide a default property panel in the Page Properties and Component Properties dialogs. Set the attribute value to `true` to show the default property panel; set it to `false` to hide a default property panel.

Example 5–9 shows the `rendered` attribute set to `false` for the Content Style property tab of a `Command Button` component. For information about default property panels, see Section B.3, "Oracle Composer Default Add-Ons and Property Panels."

Example 5–9 rendered Attribute for a Property Panel

```
<property-panels>
  <property-panel name="cmdbtn">
    <component>oracle.rich.CommandButton</component>
    <panel name="prop.panel.cmdbtn" />
    <panel name="oracle.adf.pageeditor.pane.content-style-editor" rendered="false"
/>
  </property-panel>
</property-panels>
```

5.3.4 How to Override Default Property Panels

You can override a default property panel with a custom panel you provide by registering your custom panel using the same name as the default panel. When you register a custom panel, provide the name of the default panel as the value for the `name` attribute within the `panel` element. Your custom property panel is then rendered on the default tab in lieu of the tab's default properties in the Component Properties or Page Properties dialog.

For a list of default property panels that you can override, see Section B.3, "Oracle Composer Default Add-Ons and Property Panels."

5.3.5 How to Selectively Render Property Panels

Custom property panels registered in the application are rendered in the Component Properties dialog when users click the Edit icon on a specified component. To display property panels selectively, you can use an EL value in the `property-panel`'s `rendered` attribute as follows:

```
<property-panel name="global-but-just" rendered="#{bean.showProperty}">
```

5.4 Configuring Event Handlers for Oracle Composer UI Events

When a user performs an action, such as clicking a Save button, Oracle Composer calls back into the application code to give the application a chance to respond to the action—or *event*. This is useful in performing application-specific tasks on such events. Moreover, this is the recommended approach to, for example, saving the changes on a custom property panel.

Oracle Composer provides a means of registering event handlers for different events. This section describes how. It contains the following subsections:

- Section 5.4.1, "How to Create and Register Handlers for Composer UI Events"
- Section 5.4.2, "What Happens When You Create and Register Event Handlers"

5.4.1 How to Create and Register Handlers for Composer UI Events

When you register an event handler with Oracle Composer, it is called when the corresponding event is fired in the Composer UI. This section describes how to create an event handler and register it with Oracle Composer. It contains the following subsections:

- Section 5.4.1.1, "UI Events that Support Event Handler Registration"
- Section 5.4.1.2, "Creating a Save Event Handler: Example"
- Section 5.4.1.3, "Registering an Event Handler with Oracle Composer"

5.4.1.1 UI Events that Support Event Handler Registration

Table 5–1 lists the UI events for which Oracle Composer currently supports registering of handlers.

Table 5–1 Events for Which Registering Handlers is Supported

Event	Cause	Event Type	Listener Interface to Be Implemented	Method to Be Implemented	Event Parameters
Save	Invoked when a user clicks the Save button on the Oracle Composer toolbar, or the Apply or OK button in the Component Properties dialog or Page Properties dialog.	`save`	`oracle.adf.view.page.event.SaveListener`	`processSave`	`SaveEvent`
Close	Invoked when a user clicks the Close button on the Oracle Composer toolbar.	`close`	`oracle.adf.view.page.event.CloseListener`	`processClose`	`CloseEvent`
Deletion	Invoked when a user deletes the component.	`delete`	`oracle.adf.view.page.event.DeletionListener`	`processDeletion`	`DeletionEvent` Get the deleted component using `DeletionEvent's getComponent` method.
Addition	Invoked when a user adds a component to the page from the catalog, by clicking the **Add** button against an item in the Catalog dialog.	`add`	`oracle.adf.view.page.event.AdditionListener`	`processAddition`	`AdditionEvent`
Selection	Invoked when a user selects: ■ The Edit icon on a Panel Customizable or Show Detail Frame component in Design view ■ A component on the page or in the hierarchy in Source view	`select`	`oracle.adf.view.page.event.SelectionListener`	`processSelection`	`SelectionEvent`

5.4.1.2 Creating a Save Event Handler: Example

To register an event with Oracle Composer, you must first create a Java class and implement the appropriate listener for the event handler. This section describes the steps to create a Save event handler. You can perform similar steps to create event handlers for all the supported events listed in Table 5–1.

A Save event handler is called when a user clicks the Save button on the Oracle Composer toolbar or the Apply or OK button in the Component Properties or Page Properties dialog. A Save handler must implement `oracle.adf.view.page.editor.event.SaveListener`.

To create a Save event handler:

1. In JDeveloper, select the **ViewController** project, and from the **File** menu, choose **New**.
2. In the New Gallery dialog, expand **General**, select **Java**, then **Java Class**, and click **OK**.
3. In the Create Java Class dialog, specify a name for the class, for example, `SaveHandler`.
4. Under the Optional Attributes section, add the `oracle.adf.view.page.editor.event.SaveListener` interface.
5. Click **OK**.

   The Java class source looks like the following:

   ```
   package view;

   import javax.faces.event.AbortProcessingException;

   import oracle.adf.view.page.editor.event.SaveEvent;
   import oracle.adf.view.page.editor.event.SaveListener;

   public class Class1 implements SaveListener {
     public Class1() {
       super();
     }
     public void processSave(SaveEvent saveEvent) throws AbortProcessingException
   {
   // Your implementation goes here
     }
   }
   ```

   You must declare the `processSave` method as `throws AbortProcessingException` because the method may throw this exception if the event must be canceled. You can include the reason for cancelling this event in the Exception object when you create it.

   On throwing this exception, further processing of this event is canceled and the listeners that are in the queue are skipped.

You can create event handlers for all supported events by performing steps similar to these.

5.4.1.3 Registering an Event Handler with Oracle Composer

After creating and implementing an event handler, you must register it with Oracle Composer. Registration is necessary for ensuring that the handler is called back by Composer when the corresponding event occurs in the UI.

Register event handlers in the Oracle Composer extension file, `/META-INF/pe_ext.xml`. For more information about creating this file, see Section 5.2.1.2, "Registering Add-Ons with Oracle Composer."

To register an event handler:

1. In the `pe_ext.xml` file, add the following entries:

   ```
   <event-handlers>
     <event-handler event="save">view.SaveHandler</event-handler>
   </event-handlers>
   ```

The values you provide for the `event` attribute and between the `event-handler` tags are unique to the type of event being entered and the name you specified for the event class.

2. Save the file.

5.4.2 What Happens When You Create and Register Event Handlers

At runtime, registered event handlers are called according to the sequence in the extension file and according to the order in which they were found on the class path. Oracle Composer's native event handler is called last.

On invocation of an event handler's `process`*`EventName`* method, if an event handler throws `AbortProcessingException`, then the event is canceled and no further event handlers are called, including Composer's native event handlers.

If, however, an error occurs while instantiating an event handler, then Oracle Composer continues with the next event handler. A warning message is logged.

5.5 Defining Property Filters

Some component properties are not displayed in Oracle Composer's Component Properties dialog box because they are filtered out by default. The default filters are defined in the `<filter-config>` section of the Composer's extension file (/META-INF/pe_ext.xml).

Global filters filter attributes for all components. They are defined using the `<global-attribute-filter>` tag. Tag-level filters filter attributes for a specified component only. They are defined using the `<taglib-filter>` tag.

Note: In an extension file, you can have any number of `<taglib-filter>` tags under `<filter-config>`, but you can have only one `<global-attribute-filter>` tag to define all global attribute filters.

You can define additional filters to hide more properties in the Component Properties dialog. This section describes how. It contains the following subsections:

- Section 5.5.1, "How to Define Property Filters"
- Section 5.5.2, "What Happens at Runtime"
- Section 5.5.3, "How to Remove Property Filters"

5.5.1 How to Define Property Filters

Oracle Composer defines a built-in filter configuration. You can use the extension file, `pe_ext.xml`, to define additional property filters and to delete filters. You can define any number of filters, even for a single tag, in different extension files. Oracle Composer merges the filtering information from all the extension files.

To define property filters in an extension file:

1. In the `pe_ext.xml` file, add the `filter-config` element as shown in the following example:

Note: For information about creating the `pe_ext.xml` file, see Section 5.2.1.2, "Registering Add-Ons with Oracle Composer."

```
<filter-config>
  <global-attribute-filter>
    <attribute name="accessKey" />
    <attribute name="attributeChangeListener" />
    <attribute name="autoSubmit" />
    <attribute name="binding" />
  </global-attribute-filter>
  <taglib-filter namespace="http://xmlns.oracle.com/adf/faces/rich">
    <tag name="commandButton">
      <attribute name="text" />
      <attribute name="icon" />
    </tag>
  </taglib-filter>
</filter-config>
```

2. Save the file.

5.5.2 What Happens at Runtime

At runtime, when you edit a component's properties, the properties that were filtered out are not rendered in the Component Properties dialog.

5.5.3 How to Remove Property Filters

You can remove global and tag-level filters so that previously filtered properties are now rendered in the Component Properties dialog. This is useful for displaying properties that are filtered out by Oracle Composer's built-in filters or by another extension file defined elsewhere the application.

Note: After you remove a property filter, it is rendered in the Component Properties dialog even if a filter is defined for that property in another extension file.

To remove a property filter:

1. Edit the Oracle Composer extension file, `pe_ext.xml`, available in the `META-INF` directory.

   You can remove property filters by editing entries in this file.

2. Search for the attribute from which to remove the filter, and set `filtered` to `false` in the `<attribute>` tag as shown in the following example:

```
<pe-extension xmlns="http://xmlns.oracle.com/adf/pageeditor/extension">
. . .
<filter-config>
  <global-attribute-filter>
    <attribute name="accessKey" filtered="false" />
    <attribute name="attributeChangeListener" />
    . . .
  </global-attribute-filter>
  <taglib-filter namespace="http://xmlns.oracle.com/adf/faces/rich">
    <tag name="activeCommandToolbarButton">
      . . .
```

```
        <attribute name="windowWidth" filtered="false"/>
      </tag>
    </taglib-filter>
  </filter-config>
</pe-extension>
```

3. Save the file.

5.6 Performing MDS-Specific Configurations

Most industries customize their enterprise applications to serve different audiences and domains. Problems can arise when an application is modified at the site level. For example, upgrading an application with site-level customizations may lead to data loss or data-merge errors. Consequently, a new version of the application cannot be deployed until all merge conflicts are reconciled.

In the metadata domain, MDS provides the customization feature to address such problems. The customization feature allows for the creation of nonintrusive customization layers that are applied on top of the base application definitions. Customization layers, or layered changes, are described in their own documents and are stored separately from the base application definition. At runtime, applicable customizations are loaded from the metadata store and layered over the base metadata definition to produce the desired effect. Product upgrades and patches affect only the base metadata definition, so customizations continue to function properly.

You can apply component-level restrictions to the JSF pages that use Oracle Composer components, add customization layers to WebCenter applications that provide different user- and site-level privileges, and set the location of a customization store for all MDS customizations.

The MDS enables clients to specify multiple customization types. For example, you can add customizations to runtime modes, application or user roles, application states, or any client specified criteria. Each such customization type is called a *customization layer*. Customization layers are applied in order of precedence, that is, if the same change is made in two different layers that apply to the given user and session, the change defined in the higher precedence layer is applied first.

This section provides an example of adding of a customization layer. Additionally, it provides a usage example of the `customizationAllowed` attribute. It contains the following subsections:

- Section 5.6.1, "Adding Customization Layers to View and Edit Modes: Example"
- Section 5.6.2, "Applying Tag-Level Security Using the customizationAllowed Attribute"

5.6.1 Adding Customization Layers to View and Edit Modes: Example

You can apply customizations to a metadata object based on client-defined criteria. For example, you can customize an application and the metadata objects that it uses based on an end-user's permissions, an application's deployment location (also called *localization*), or a specific industry domain. Each such category—permissions, localization, and domain—denotes a customization layer, and each is depicted using a `CustomizationClass`. A `CustomizationClass` is the interface MDS uses to identify the customization layer to be overlaid on the base definition. See Section 5.6.1.3, "How to Create a Custom UserCC Tip Layer" for an example.

When you implement a `CustomizationClass`, you must also register it with the MDS. The MDS provides the ability to associate a list of `CustomizationClass` types with a single `MetadataObject`. This is called the *fine-grained* association. The MDS also provides the ability to associate a list of `CustomizationClass` types with a set of `MetadataObjects`. This is called the *coarse-grained* association.

This section explains through example how adding customization tip layers to View and Edit runtime modes provides personalization capabilities to all users and customization capabilities to selective users. To enable customizations in the Edit mode, the `site` tip layer is added. To enable personalization in the View mode, the `user` tip layer is added. By default, the `user` tip layer is applied on top of the `site` tip layer. The `user` tip layer stores all customizations made in the View mode in a specific location created for the user who made them. Such changes are visible only to that user. The `site` tip layer stores all customizations made in the Edit mode and are visible to all users.

To enable tip layers at runtime, Oracle Composer provides the `WebCenterComposerFilter` filter and supplies a means of defining an abstract factory for creating the MDS `SessionOptions` object. This object provides applicable customization layers at runtime and enables users to perform personalizations (View mode) or customizations (Edit mode) based on their role. When creating a new MDS session, the `MDSSession.createSession` method of this object is used to specify the session options.

This section provides an example exercise for creating, implementing, and registering customization layers, configuring WebCenterComposerFilter, and switching between MDS customization layers. It contains the following subsections:

- Section 5.6.1.1, "How to Add Oracle Composer to a JSF Page"
- Section 5.6.1.2, "How to Create a SiteCC Tip Layer"
- Section 5.6.1.3, "How to Create a Custom UserCC Tip Layer"
- Section 5.6.1.4, "How to Implement the ComposerSessionOptionsFactory Class"
- Section 5.6.1.5, "How to Register the Implementation with Oracle Composer"
- Section 5.6.1.6, "How to Configure WebCenterComposerFilter"
- Section 5.6.1.7, "How to Redirect the Servlet to Enable Switch Between MDS Customization Layers"
- Section 5.6.1.8, "What Happens at Runtime"

5.6.1.1 How to Add Oracle Composer to a JSF Page

This section describes how to add the Page Customizable component to a JSF page. The purpose of this exercise is to provide Oracle Composer in the Edit mode at runtime so that the `admin` user can perform customizations at the site level. This section includes the addition of a Change Mode Link to enable switching from View mode to Edit mode at runtime.

To add Oracle Composer to a JSF Page:

1. Open the JSPX page and select **Oracle Composer** from the Component Palette.
2. Select **Change Mode Link** and drop it onto the page.
3. Select **Page Customizable** and drop it onto the page.

   The Source view should like this:

   ```
   <af:form id="f1">
     <pe:changeModeLink id="cml1"/>
     <pe:pageCustomizable id="pageCustomizable1">
   ```

```
    <cust:panelCustomizable id="panelCustomizable1" layout="scroll"/>
      <f:facet name="editor">
        <pe:pageEditor id="pep1"/>
      </f:facet>
  </pe:pageCustomizable>
</af:form>
```

When you drop these components onto the page, the default `UserCC` tip layer is extended by the ADF. Consequently, the `adf-config.xml` file is updated with this customization class:

```
<cust-config>
  <match>
    <customization-class name="oracle.adf.share.config.UserCC" />
  </match>
</cust-config>
```

5.6.1.2 How to Create a SiteCC Tip Layer

This section describes how to create a `SiteCC` tip layer, `site`, in which all site-level customizations performed in Edit mode are stored. In this sample application, site-level customizations are stored in the `/mds/mdssys/cust/site/webcenter/`*`pagename`*`.jspx.xml`directory.

To create the `site` tip layer:

1. From the File menu, choose **New**.
2. In the New Gallery dialog, expand **General**, select **Java**, then **Java Class**, and click **OK**.

   The Create Java Class dialog box opens.
3. In the **Name** field, enter `SiteCC`.
4. In the **Extends** field, enter `CustomizationClass`.

   This imports `oracle.mds.cust.CustomizationClass`.
5. Click **OK.**

   The `SiteCC.java` file is rendered in the Source view.
6. To implement the `getCacheHint`, `getName`, and `getValue` methods, double-click the bulb on the left corner and select **Implement Methods**.
7. In the Implement Methods dialog, click **OK**.
8. In the Source view, press `Enter` after `public class SiteCC extends CustomizationClass`, and add the following `string`:

   ```
   {
     private static final String DEFAULT_LAYER_NAME = "site";
     private String mLayerName = DEFAULT_LAYER_NAME;
     private String mLayerValue = "webcenter";
   ```

   The following libraries are imported:

   ```
   import oracle.mds.core.MetadataObject;
   import oracle.mds.core.RestrictedSession;
   import oracle.mds.cust.CacheHint;
   ```
9. Update the code as shown in **bold** here:

   ```
   public class SiteCC extends CustomizationClass
   ```

```
{
  private static final String DEFAULT_LAYER_NAME = "site";
  private String mLayerName = DEFAULT_LAYER_NAME;
  private String mLayerValue = "webcenter";
  Note: You can provide any site name.

  public CacheHint getCacheHint()
  {
    return CacheHint.ALL_USERS;
  }

  public String getName()
  {
    return mLayerName;
  }

  public String[] getValue(RestrictedSession mdsSession, MetadataObject mo)
  {
    return new String[] { mLayerValue };
  }
}
```

10. Save the `SiteCC.java` file.

5.6.1.3 How to Create a Custom UserCC Tip Layer

The ADF extends a default UserCC (`user`) tip layer when you drop Oracle Composer components onto a JSF page. Subsequently, the `adf-config.xml` file is updated to include the UserCC customization class.

This section describes how to create a custom `user` tip layer for a user, `scott`. This layer is applied on top of the `site` layer. The personalizations that a user performs in View mode are saved in this `user` tip layer in a folder created specifically for the logged-in user. In this example, the personalizations performed in View mode are saved in the `/mds/mdssys/cust/user/scott/`*`pagename`*`.jspx.xml`directory.

To create the user tip layer:

1. From the File menu, choose **New**.
2. In the New Gallery dialog, expand **General**, select **Java**, then **Java Class**, and click **OK**.

   The Create Java Class dialog box opens.
3. In the **Name** field, enter `UserCC`.
4. In the **Extends** field, enter `CustomizationClass`.

   This imports the `oracle.mds.cust.CustomizationClass`.
5. Click **OK.**

   The `UserCC.java` file displays in the Source view.
6. To implement the `getCacheHint`, `getName`, and `getValue` methods, double-click the bulb on the left corner and select **Implement Methods**.
7. In the Implement Methods dialog box, click **OK**.
8. In the Source view, press `Enter` after `public class SiteCC extends CustomizationClass`, and add the following `string`:

```
{
  private static final String DEFAULT_LAYER_NAME = "user";
  private String mLayerName = DEFAULT_LAYER_NAME;
  private String mLayerValue = "scott"; //The name of the logged-in user
in this example. The logged-in user name can be acquired dynamically.
```

The following libraries are imported:

```
import oracle.mds.core.MetadataObject;
import oracle.mds.core.RestrictedSession;
import oracle.mds.cust.CacheHint;
```

9. Update the code as shown in **bold** here:

```
public class UserCC extends CustomizationClass
{
  private static final String DEFAULT_LAYER_NAME = "user";
  private String mLayerName = DEFAULT_LAYER_NAME;
  private String mLayerValue = "scott";

  public CacheHint getCacheHint()
  {
    return CacheHint.USER;
  }

  public String getName()
  {
    return mLayerName;
  }

  public String[] getValue(RestrictedSession mdsSession, MetadataObject mo)
  {
    return new String[]{mLayerValue};
  }
}
```

10. Save the `UserCC.java` file.

5.6.1.4 How to Implement the ComposerSessionOptionsFactory Class

In this section, the `ComposerSessionOptionsFactory` class provided by the ADF is implemented in order to supply MDS `SessionOptions` for each HTTP request.

To implement the `ComposerSessionOptionsFactory` class:

1. From the **File** menu, choose **New**.
2. In the New Gallery dialog, expand **General**, select **Java**, then **Java Class**, and click **OK**.

   The Create Java Class dialog box opens.
3. In the **Name** field, enter `AppsSessionOptionsFactoryImpl`.
4. Click **OK.**

   The `AppsSessionOptionsFactoryImpl.java` file is rendered in the Source view.
5. Import the following libraries:

```
import oracle.adf.view.page.editor.mds.ComposerSessionOptionsFactory;
import oracle.adf.view.page.editor.mode.ModeContext;
import oracle.mds.config.CustClassListMapping;
import oracle.mds.config.CustConfig;
import oracle.mds.core.SessionOptions;
import oracle.mds.cust.CustClassList;
import oracle.mds.cust.CustomizationClass;
```

6. Add the following code to implement the `ComposerSessionOptionsFactory` class and provide `SessionOptions`:

```
public class AppsSessionOptionsFactoryImpl
  implements ComposerSessionOptionsFactory
{

  public SessionOptions createSessionOptions(SessionOptions
defaultSessionOptions, String mode)
  {
    CustomizationClass[] custLayer;
    CustConfig custConfig = null;

    if (ModeContext.EDIT_MODE.equals(mode))
    {
      //Mode is Edit, change to SiteCC
      custLayer = EDIT_LAYER;
    }
    else
    {
      //Mode is View, change to UserCC + SiteCC
      custLayer = VIEW_LAYER;
    }

    try
    {
      CustClassList custClassList = new CustClassList(custLayer);
      CustClassListMapping custClassListMapping =
        new CustClassListMapping("/", null, null, custClassList);
      custConfig = new CustConfig(new CustClassListMapping[]
        { custClassListMapping });
    }
    catch (Exception e)
    {
      e.printStackTrace();

    }
if(defaultOptions.getServletContextAsObject() != null)
{
    return new SessionOptions(defaultSessionOptions.getIsolationLevel(),
                              defaultSessionOptions.getLocale(), custConfig,
                              defaultSessionOptions.getVersionContext(),
                              defaultSessionOptions.getVersionCreatorName(),
                              defaultSessionOptions.getCustomizationPolicy());
                              sessionOptions.getServletContextAsObject());
  }
    else
    {
      return new SessionOptions(sessionOptions.getIsolationLevel(),
                              sessionOptions.getLocale(), custConfig,
                              sessionOptions.getVersionContext(),
                              sessionOptions.getVersionCreatorName(),
                              sessionOptions.getCustomizationPolicy());
    }
  }

  //Edit mode SiteCC
  private static final CustomizationClass[] EDIT_LAYER =
    new CustomizationClass[]
    { new SiteCC() };

  //View mode SiteCC + USerCC
  private static final CustomizationClass[] VIEW_LAYER =
    new CustomizationClass[]
    { new SiteCC(), new UserCC() };

}
```

5.6.1.5 How to Register the Implementation with Oracle Composer

For the `site` and `user` customization layers to function, you must register the `ComposerSessionOptionsFactory` class with Oracle Composer. For example, if the concrete class is `view.AppsSessionOptionsFactoryImpl`, the following snippet must be added to the `adf-config.xml` file located in the `\.adf\META-INF` folder in your application directory:

```
<page-editor-config xmlns="http://xmlns.oracle.com/adf/pageeditor/config">

<session-options-factory>view.AppsSessionOptionsFactoryImpl</session-options-facto
ry>
</page-editor-config>
```

5.6.1.6 How to Configure WebCenterComposerFilter

You must configure the `WebCenterComposerFilter` filter in the `web.xml` file located in the `ViewController\public_html\WEB-INF` folder in your application directory. This filter registers Oracle Composer's concrete `SessionOptionsFactory` with the ADF for every HTTP request. When the filter receives a call from the ADF, it forwards the request to the WebCenter application and gets the `SessionOptions` with the new customized layer. If you have not set the `Sandbox` or `VersionContext` in the `SessionOptions`, then Oracle Composer sets its own Sandbox and returns it to the ADF. For more information on Sandbox, see Section 5.7.1, "How to Enable Oracle Composer Sandbox Creation." The `composerFilter` and its filter mapping must be configured after `ServletADFFilter` and before `ADFBindingFilter`. For example, see the following `web.xml` file:

```
....

  <filter>
    <filter-name>ServletADFFilter</filter-name>
    <filter-class>oracle.adf.share.http.ServletADFFilter</filter-class>
  </filter>
  <!-- WebCenterComposerFilter goes here -->
  <filter>
    <filter-name>composerFilter</filter-name>

<filter-class>oracle.adf.view.page.editor.webapp.WebCenterComposerFilter</filter-c
lass>
  </filter>
  <filter>
    <filter-name>adfBindings</filter-name>
    <filter-class>oracle.adf.model.servlet.ADFBindingFilter</filter-class>
  </filter>
.....
  <filter-mapping>
    <filter-name>ServletADFFilter</filter-name>
    <servlet-name>Faces Servlet</servlet-name>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>REQUEST</dispatcher>
  </filter-mapping>

  <!-- WebCenterComposerFilter mapping goes here -->
  <filter-mapping>
    <filter-name>composerFilter</filter-name>
    <servlet-name>Faces Servlet</servlet-name>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>REQUEST</dispatcher>
  </filter-mapping>
  <filter-mapping>
    <filter-name>adfBindings</filter-name>
    <servlet-name>Faces Servlet</servlet-name>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>REQUEST</dispatcher>
  </filter-mapping>
....
```

For information about the Oracle Composer-specific configurations you can make in `web.xml`, see Section B.2.3, "web.xml."

5.6.1.7 How to Redirect the Servlet to Enable Switch Between MDS Customization Layers

To redirect the servlet, that is, to refresh the full page at runtime, you must create:

- The `AppNavigationUtils` class, which calls the `AppNavigationUtils.redirectToSamePage()` method
- The `AppCloseHandler` CloseListener, which uses the `AppNavigationUtils` class
- The `AppModeBean`, which displays Edit mode

This section describes how to create these objects. It contains the following subsections:

- Section 5.6.1.7.1, "How to Create the AppNavigationUtils Class"
- Section 5.6.1.7.2, "How to Create AppCloseHandler"
- Section 5.6.1.7.3, "How to Register the AppCloseHandler"
- Section 5.6.1.7.4, "How to Create AppModeBean"

5.6.1.7.1 How to Create the AppNavigationUtils Class To create the `AppNavigationUtils` class:

1. From the File menu, choose **New**.
2. In the New Gallery dialog, expand **General**, select **Java**, then **Java Class**, and click **OK**.

   The Create Java Class dialog box opens.
3. In the **Name** field, enter `AppNavigationUtils` and click **OK**.

   The `AppNavigationUtils.java` file is rendered in the Source view.
4. Import the following libraries:

   ```
   import javax.faces.context.FacesContext;
   import javax.servlet.http.HttpServletRequest;
   ```

5. Add the following code:

```
public class AppNavigationUtils
{

  public static void redirectToSamePage()
  {
    HttpServletRequest request =
(HttpServletRequest)FacesContext.getCurrentInstance().getExternalContext().getR
equest();
    String url = request.getRequestURL().toString();
    String _adfCtrlState = request.getParameter("_adf.ctrl-state");
    url = url + "?_adf.ctrl-state="+ _adfCtrlState;
    System.out.println(url);
    try
    {
      FacesContext.getCurrentInstance().getExternalContext().redirect(url);
      FacesContext.getCurrentInstance().responseComplete();
    }
    catch(Exception e)
    {
      e.printStackTrace();
    }
  }
}
```

5.6.1.7.2 How to Create AppCloseHandler To create `AppCloseHandler`:

1. From the File menu, choose **New**.
2. In the New Gallery dialog, expand **General**, select **Java**, then **Java Class**, and click **OK**.

   The Create Java Class dialog box opens.
3. In the **Name** field, enter `AppCloseHandler` and click **OK**.

   The `AppCloseHandler.java` file displays in the Source view.
4. Import the following libraries:

```
import oracle.adf.view.page.editor.event.CloseEvent;
import oracle.adf.view.page.editor.event.CloseListener;
```

5. Add the following code:

```
public class AppCloseHandler
  implements CloseListener
{

  public void processClose(CloseEvent closeEvent)
  {
    AppNavigationUtils.redirectToSamePage();
  }
}
```

5.6.1.7.3 How to Register the AppCloseHandler After creating `AppCloseHandler`, you must register it in the Oracle Composer extension file, `pe_ext.xml`.

To register the event handler:

1. In the `pe_ext.xml` file, add the following entries:

```
<event-handlers>
  <event-handler event="close">view.AppCloseHandler</event-handler>
</event-handlers>
```

2. Save the file.

For more information about event handlers, see Section 5.4, "Configuring Event Handlers for Oracle Composer UI Events."

5.6.1.7.4 How to Create AppModeBean

To create `AppModeBean`:

1. From the **File** menu, choose **New**.
2. In the New Gallery dialog, expand **General**, select **Java**, then **Java Class**, and click **OK**.

   The Create Java Class dialog box opens.
3. In the **Name** field, enter `AppModeBean` and click **OK**.

   The `AppModeBean.java` file displays in the Source view.
4. Import the following libraries:

```
import javax.faces.event.ActionEvent;
import oracle.adf.view.page.editor.mode.ModeContext;
```

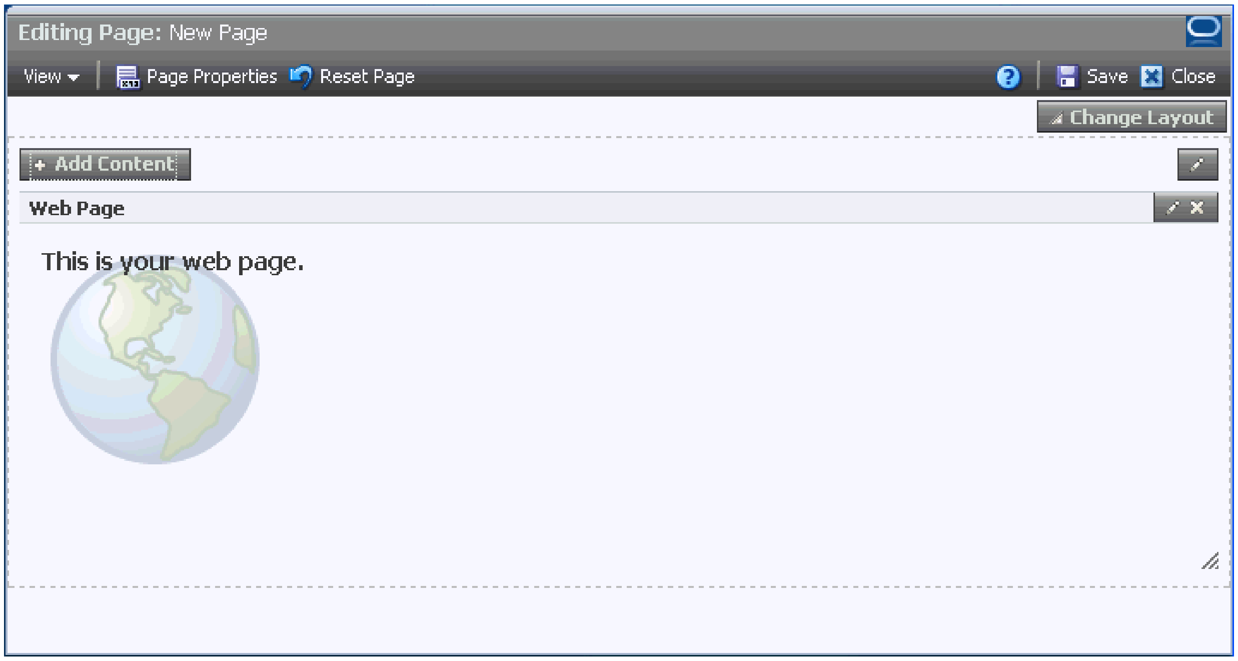
5. Add the following code:

```
public class AppModeBean
{

  public void edit(ActionEvent actionEvent)
  {
    ModeContext.getCurrent().setEditMode();
    AppNavigationUtils.redirectToSamePage();
  }
}
```

5.6.1.8 What Happens at Runtime

To see how the `site` customization layer functions, first run the JSF page in a browser. Then log in to the page as `admin`, and click the **Edit** link to switch to Edit mode. At runtime, customize the page. For example, drop a movable box, a hyperlink, and an image onto the page. The page should look like Figure 5–4.

Figure 5–4 Customized Page



Go to `\mds\mdssys\cust\site\webcenter` in your application directory, and open the *`pagename`*`.jspx.xml` file. This is where the site-level customizations that you made to the page are stored.

To use the `user` customization layer, log in to the page as `scott` and personalize the page in View mode. Then go to `\mds\mdssys\cust\user\scott` in your application directory, and open the *`pagename`*`.jspx.xml` file. This is where the user-level customizations (personalizations) that you made to the page are stored.

5.6.2 Applying Tag-Level Security Using the customizationAllowed Attribute

By default, customization is enabled on all components inside a `Page Customizable` component. You may want to change this to restrict customization on some components. MDS provides the `customizationAllowed` and `customizationAllowedBy` attributes to enable you to restrict customization of specific component instances.

The `customizationAllowed` attribute controls whether the component can be customized at runtime. If you set this attribute to `false`, then the component cannot be customized at runtime. That is, in the Component Properties dialog the properties are grayed out and do not allow editing.

The `customizationAllowedBy` attribute specifies the roles for which customization is enabled.

This section provides examples of enabling and restricting customization on a component. It contains the following subsections:

- Section 5.6.2.1, "How to Enable Customization on an Image Component"
- Section 5.6.2.2, "How to Restrict Customization on an Image Component"

For more information, see Extended Metadata and Annotation Properties in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

5.6.2.1 How to Enable Customization on an Image Component

To enable customization on an Image component:

1. In the `MyPage.jspx` page that you created in Section 4.3, "Designing Editable Pages Using Oracle Composer Components: Example," add an ADF Faces Image component, for example, `brandingImage.gif`, inside the `Panel Customizable` called `panelCustomizable1`.

   Customization is enabled automatically on the `Image` component since it is nested inside a `Page Customizable` component.

2. Run `MyPage.jspx`.
3. Switch to Edit mode of the page.
4. From the **View** menu, select **Source** to switch to Source view of the page.
5. Select the image in the component hierarchy in the Source View panel and click the **Show the properties of Image** icon.

   The Component Properties dialog displays the image properties. You can edit any available property in this dialog. Edit a property and click **OK**. The property's editable value demonstrates that customization is enabled on the component.

5.6.2.2 How to Restrict Customization on an Image Component

You can restrict customization of an image on the page by setting the `customizationAllowed` attribute to `false` on the image component.

To restrict customization on the Image Component:

1. In Oracle JDeveloper, select the `Image` component that you added in the previous section, and in the Property Inspector set the value for **customizationAllowed** to `false`.
2. Run `MyPage.jspx`.
3. Switch to page Edit mode.
4. From the **View** menu, select **Source** to switch to the Source view of the page.
5. Select the image in the component hierarchy in the Source View panel, and click the **Show the properties of Image** icon.

   In the Component Properties dialog, the properties are grayed out and cannot be edited.

5.7 Using Oracle Composer Sandbox

This section discusses the steps you can take to enable sandbox creation and describes runtime behavior of a sandbox-enabled application. It contains the following subsections:

- Section 5.7.1, "How to Enable Oracle Composer Sandbox Creation"
- Section 5.7.2, "What Happens at Runtime"
- Section 5.7.3, "How to Disable Sandbox for an Application"
- Section 5.7.4, "How to Destroy Stale Sandboxes"
- Section 5.7.5, "How to Enable Application Sandbox Creation"
- Section 5.7.6, "What Happens When You Enable Application Sandbox Creation"

For information about the sandbox, see Section 5.1.6, "Oracle Composer Sandbox."

5.7.1 How to Enable Oracle Composer Sandbox Creation

You can enable a sandbox only if your application uses a database store. Therefore, you must ensure that you have configured a database store before performing the steps in this section. For information about setting up a database store, see *Oracle Fusion Middleware Configuring and Managing JDBC for Oracle WebLogic Server*.

This section describes how to enable the creation of an Oracle Composer sandbox. It contains the following subsections:

- Section 5.7.1.1, "Updating Your Application's adf-config.xml File"
- Section 5.7.1.2, "Updating Your Application's web.xml File"

5.7.1.1 Updating Your Application's adf-config.xml File

You must update the `adf-config.xml` file to define and configure namespaces for all the metadata for which you want to support sandbox creation. This section describes how.

Note: For information about the Oracle Composer-specific configurations you can make in `adf-config.xml`, see Section B.2.2, "adf-config.xml."

To configure sandbox creation in adf-config.xml:

1. Open the `adf-config.xml` file located in the `ADF META-INF` folder under `Descriptors` in the Application Resources panel.
2. Under the `<metadata-namespaces>` element, ensure that `<namespace>` elements are defined for all metadata for which you want to enable sandbox creation, as shown in Example 5–10.

Example 5–10 Namespace Definitions in the adf-config.xml File

```
<!-- your jspx files -->
   <namespace path="/test" metadata-store-usage="webcenter_metadata_db_store"/>
<!-- your pagedef customizations alone go here -->
   <namespace path="/pageDefs" metadata-store-usage="webcenter_metadata_db_store">
     <namespace-restriction type="CUSTOMIZATIONS"/>
     </namespace>
```

3. Configure the database store, as shown in Example 5–11.

Note: Perform this step only if you plan to use WLS Administration Console to deploy your application EAR file. This configuration happens automatically if you deploy your application using Fusion Middleware Control or Oracle JDeveloper and select a database repository connection as the target MDS connection. For more information, see "Deploying WebCenter Applications" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

Example 5–11 Metadata Store Configuration in the adf-config.xml File

```
<metadata-store class-name="oracle.mds.persistence.stores.db.DBMetadataStore">
  <property name="jdbc-userid" value="userone"/>
  <property name="jdbc-password" value="userone123"/>
  <property name="jdbc-url" value="jdbc:oracle:thin:@host_name:1521:orcl"/>
  <property name="partition-name" value="partitionName"/>
</metadata-store>
<!-- <metadata-store
class-name="oracle.mds.persistence.stores.file.FileMetadataStore">
  <property name="metadata-path"
value="C:\JDeveloper\mywork\Application2\ViewController\public_html"/>
</metadata-store> -->
```

4. Add the `<sandbox-namespaces>` element and add individual `<namespace>` tags for all namespaces for which you want to enable sandbox creation, as show in Example 5–12.

Example 5–12 Configuring Namespaces for Sandbox Creation

```
<sandbox-namespaces>
  <namespace path="/test"/>
  <namespace path="/pageDefs"/>
</sandbox-namespaces>
```

5. Save the `adf-config.xml` file.

5.7.1.2 Updating Your Application's web.xml File

To ensure that a sandbox is created when you are in Edit mode of a page, you must create a filter in your application's `web.xml` file and set the appropriate filter mappings. All requests are then routed through this filter, and a sandbox is created for all Edit mode customizations. If you are using a file system metadata store, then there is no action performed on a filtered request.

Note: For information about the Oracle Composer-specific configurations you can make in `web.xml`, see Section B.2.3, "web.xml."

This section provides the example of defining a Composer-specific filter and its relevant filter mappings in your application’s `web.xml` file. It describes how to add the filter, `WebCenterComposerFilter`.

To define a Composer-specific filter and the filter mappings:

1. Open the `*application_root*/ViewController/public_html/WEB-INF/web.xml` file.
2. Add `ServletADFFilter` and `WebCenterComposerFilter` before the `adfBindings` filter so that all requests are routed through these filters first:

   ```
   <filter>
     <filter-name>ServletADFFilter</filter-name>
     <filter-class>oracle.adf.share.http.ServletADFFilter</filter-class>
   </filter>
   <filter>
     <filter-name>WebCenterComposerFilter</filter-name>

   <filter-class>oracle.adf.view.page.editor.webapp.WebCenterComposerFilter</filte
   r-class>
   </filter>
   ```

3. Add corresponding `<filter-mapping>` elements before the filter mapping for `adfBindings` filter as shown in Example 5–13.

Example 5–13 Filter Mappings for Composer-Specific Filter

```
<filter-mapping>
  <filter-name>ServletADFFilter</filter-name>
  <servlet-name>Faces Servlet</servlet-name>
  <dispatcher>FORWARD</dispatcher>
  <dispatcher>REQUEST</dispatcher>
</filter-mapping>
<filter-mapping>
  <filter-name>WebCenterComposerFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>WebCenterComposerFilter</filter-name>
  <url-pattern>/faces/*</url-pattern>
  <dispatcher>FORWARD</dispatcher>
  <dispatcher>REQUEST</dispatcher>
</filter-mapping>
```

The `<url-pattern>` tag is used to specify the pages that must pass through the `WebCenterComposerFilter` so that sandbox creation can be enabled when a page goes into Edit mode.

4. Save the `web.xml` file.

5.7.2 What Happens at Runtime

When a user switches to page Edit mode, the presence of a Save button on the Oracle Composer toolbar indicates that sandbox creation is enabled for the application. Changes made to the page are not saved until the user clicks Save. If the Save button is not rendered, the sandbox is not available and each change is committed immediately.

When editing component properties in the Component Properties dialog, clicking **Apply** results in the following:

- If the sandbox is not available, then the page is refreshed to display the changes made to component properties and changes are saved to the back end.
- If the sandbox is available, then the page is refreshed to display the changes made to component properties. To save changes, user must click **Save**.

Clicking **Save** or **Close** on the page results in the following events:

- On clicking **Save**, the sandbox is committed and a new sandbox is created. The page remains in Edit mode.
- On clicking **Close**, either of the following two events can occur:
  - If there are no changes since the last Save operation, then the sandbox is destroyed and Oracle Composer is closed, taking the user back to View mode.
  - If there are unsaved changes, then a Close Confirmation dialog displays, in which a user can select from the following options:
    - Click **Save** to commit the sandbox and close Oracle Composer.
    - Click **Don't Save** to destroy the sandbox and close Oracle Composer.
    - Click **Cancel** to close the dialog and return to Oracle Composer without saving the changes.

Notes: If a user navigates away from a page while editing it and then returns to the page, any unsaved changes are lost.

What Happens During Concurrent Edits

If two or more users are editing a page at the same time, then during the next server request, changes made by all users are visible on the page. However, if two or more users edit the same page using the same customization layer, then their changes conflict. If this happens, the page displays a message that another user is editing the page. Changes that are saved last overwrite prior changes. For example, if users A and B are editing a page simultaneously, then concurrency issues are handled as follows:

- If A saves the page first, then A's changes are committed to MDS. Later, when B saves the page, A's changes are overwritten with B's changes.
- If A deletes a component while B is trying to personalize (say move) that same component in View mode, a WebCenter error page is displayed to B. B has to simply navigate back to the original page. The deleted component does not appear, and B can continue working on other components.

5.7.3 How to Disable Sandbox for an Application

To ensure that changes are committed to the back end immediately, you can disable the sandbox.

To disable sandbox:

1. Open the `adf-config.xml` file located in the `ADF META-INF` folder under `Descriptors` in the Application Resources panel.
2. Under the `<page-editor-config>` element, add the `<disable-sandbox>` attribute and set it to `true`.
3. Save `adf-config.xml`.

5.7.4 How to Destroy Stale Sandboxes

When users edit a page at runtime, if the browser closes unexpectedly or the user navigates away from the page, the sandbox used in that session is still available. Any other user accessing the same page continues to see a concurrency message that another user is editing the same page. You can destroy such stale sandboxes to free some space in the database store and enhance performance.

You can ensure that a stale sandbox is destroyed when:

- A session times out.

  The `<session-timeout>` element in the application's `web.xml` file defines the time duration after which the sandbox is destroyed.
- The user logs in to Edit mode of the page again, with the same user name.

  In this case, a new sandbox is created when the same user switches to Edit mode. To ensure this happens, you must configure `WebCenterComposerSessionListener` in your application's `web.xml` file. This listener is called when a session times out. It creates a new sandbox when the same user renters page Edit mode.

To configure `WebCenterComposerSessionListener`:

1. Open the *application_root*/ViewController/public_html/WEB-INF/web.xml file.
2. Define a new listener, WebCenterComposerSessionListener, as shown in the following example:

```
<listener>
  <description>Oracle Composer Http Session Listener</description>
  <display-name>Oracle Composer Http Session Listener</display-name>
  <listener-class>
    oracle.adf.view.page.editor.webapp.WebCenterComposerSessionListener
  </listener-class>
</listener>
```

3. Save the web.xml file.

For information about the Oracle Composer-specific configurations you can make in web.xml, see Section B.2.3, "web.xml."

5.7.5 How to Enable Application Sandbox Creation

If you enable sandbox creation at the application level, then Oracle Composer does not create its own sandbox when page Edit mode is invoked. Instead, the application sandbox is used. That is, a user interface provided by the application is used for committing and destroying the sandbox. To enable the application to provide a sandbox for Oracle Composer, you must first perform a set of configurations.

To enable an application sandbox for use in Oracle Composer:

1. Implement a ComposerSessionOptionsFactory class to provide MDSSessionOptions for each request.

   For information about performing this task, see Section 5.6.1.4, "How to Implement the ComposerSessionOptionsFactory Class."
2. Register your implementation with Oracle Composer.

   For information about performing this task, see Section 5.6.1.5, "How to Register the Implementation with Oracle Composer."
3. Configure WebCenterComposerFilter.

   For information about performing this task, Section 5.6.1.6, "How to Configure WebCenterComposerFilter."

5.7.6 What Happens When You Enable Application Sandbox Creation

At runtime, the application-defined user interface enables users to save their changes to the back end or close Oracle Composer without saving the changes. In this case, Oracle Composer has no control over when customizations are canceled or committed to the back end.

5.8 Overriding Default Security Behavior of Oracle Composer Components

The ability to customize an artifact, such as the page, a component, or an attribute of a component, is inherited from security definitions at the tag, page, component, and attribute levels. You can override the default security definitions at various levels in keeping with your business requirement. This section describes how to override the default security behavior. It contains the following subsections:

- Section 5.8.1, "Applying Component-Level Restrictions by Defining Customization Policies"
- Section 5.8.2, "Applying a Component Instance-Level Customization Restriction By Using Security Roles: Example"
- Section 5.8.3, "Applying Attribute-Level Security"
- Section 5.8.4, "Applying Action-Level Restrictions on Show Detail Component Actions"

Note: For information about the default security behavior of Oracle Composer components, see Section 4.1.5, "Security and Oracle Composer."

5.8.1 Applying Component-Level Restrictions by Defining Customization Policies

By default, the MDS restricts all customizations. The `Page Customizable` component is available for lifting default customization restrictions on the `Page Customizable` and all its child components. Another way to configure customization on a set of component attributes is to use MDS type-level restrictions or instance-level restrictions. This section describes how. It contains the following subsections:

- Section 5.8.1.1, "How to Define Type-Level Customization Policies"
- Section 5.8.1.2, "How to Define Instance-Level Customization Policies"

5.8.1.1 How to Define Type-Level Customization Policies

You can enable type-level restrictions on components and their attributes in a standalone XML file and then register this file in the `mds-config` section of the `adf-config.xml` file. The standalone XML file contains annotations that match the types for which customization restrictions must be specified.

To enable customization of selected components or attributes at the type level:

1. Create an XML document, for example `standalone.xml`, add the following code, and replace the text in bold with appropriate values:

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammarMetadata xmlns="http://xmlns.oracle.com/bali/xml/metadata"
                 xmlns:mds="http://xmlns.oracle.com/mds"
                 namespace="http://java.sun.com/JSP/Page">
  <elementMetadata elementName="component_name">
    <mds:customizationAllowed>true</mds:customizationAllowed>
      <attributeMetadata attributeName="attribute_name">
        <mds:customizationAllowedBy>security_roles</mds:customizationAllowedBy>
      </attributeMetadata>
  </elementMetadata>
</grammarMetadata>
```

   The `<customizationAllowedBy>` tag can appear only once. Multiple values can be specified as a space-separated list of allowed policies as part of the declaration.

   The `<customizationAllowed>` tag takes a Boolean as its value. A value of `true` for this tag means that anyone can customize the specified component, so long as other inherited customization polices allow it.

   The `<customizationAllowedBy>` tags do not serve any purpose if the same object has been tagged with `<customizationAllowed>false</`

customizationAllowed>. Additionally, the customization must be allowed at the top level of the object tree. Otherwise the default behavior dictates that no customization can be permitted at a lower level.

2. Register this XML file in the `mds-config` section of the `adf-config.xml` file as follows:

```
<mds-config xmlns="http://xmlns.oracle.com/mds/config">
  <type-config>
    <standalone-definitions>
      <classpath>File_Path/standalone.xml</classpath>
    </standalone-definitions>
  </type-config>
</mds-config>
```

3. Make the standalone files and associated XSDs available in a shared library JAR file; otherwise MDS cannot load them due to class loader restrictions.

5.8.1.2 How to Define Instance-Level Customization Policies

You can enable customization on a component or any of its attributes at the instance level. To apply customization restriction at the component level, you can set the `customizationAllowed` and `customizationAllowedBy` attributes on the component in JDeveloper (see Section 5.6, "Performing MDS-Specific Configurations"). To apply restrictions on a component's attributes, perform the steps described in this section.

The code for instance-level customization policies is similar to that used in the type-level policy definitions. They have the same meaning at the instance level. However, instance level policies are subordinate to type restrictions. That is, all restrictions are inherited from the type, and no customization is allowed unless customization is enabled at the type level.

Instance-level policies for component attributes are defined in an RDF file. The RDF file is picked up by MDS, and the policies are implemented.

To define instance-level customization policies:

1. Create an RDF file with the same name as the JSPX file, but using the RDF extension, for example `main.jspx.rdf`.

   Creating the RDF file in this folder structure ensures that the file is picked up by MDS and the policies defined here are implemented.

2. Include tags in the following format to apply restrictions on a component's attribute:

```
tag_id(xmlns(tag_prefix=tag_namespace))/@tag_prefix:attribute_name
```

   The following example shows how to apply restriction on the `text` attribute of a `Command Button` component:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/">
<rdf:Description rdf:about="/"/>
<!-- Opens the customization on a tag, for example, an af:commandButton whose
ID is 'cb1'.  -->
<rdf:Description rdf:about="cb1">
  <customizationAllowed
xmlns="http://xmlns.oracle.com/mds">false</customizationAllowed>
</rdf:Description>
```

```
<!-- Restricts customizations on the commandButton's attribute 'text'. -->
<rdf:Description
rdf:about="cb1(xmlns(af=http://xmlns.oracle.com/adf/faces/rich))/@af:text">
  <customizationAllowed
xmlns="http://xmlns.oracle.com/mds">false</customizationAllowed>
</rdf:Description>
```

At runtime, these policies are read and the artifacts are enabled for customization accordingly.

5.8.2 Applying a Component Instance-Level Customization Restriction By Using Security Roles: Example

This section explains how you can use component-level restrictions to limit access to certain component functions, based on different user roles and responsibilities. Specifically, it describes how to apply an instance-level customization restriction to a `Panel Customizable` component to provide customization access to only the admin user. That is, when a user with `admin` privileges logs into the system at runtime, the user can customize the `Panel Customizable` on which the instance-level customization is applied.

In Oracle Composer's Property Inspector, the properties of the `Panel Customizable` appear gray to users who do not have `admin` privileges. These users cannot drop any `Movable Box` or `Show Detail Frame` into the `Panel Customizable` because of the applied restriction. Moreover, they cannot rearrange the `Show Detail Frames` inside this `Panel Customizable`.

To restrict a functionality based on user roles is one way to apply customization restrictions. You can also use expression language expressions (ELs), which provide roles-based results, to enable or restrict access to a component function.

Note: MDS customization restrictions can be applied only to `Panel Customizable`, `Show Detail Frame`, and `Layout Customizable` components. These restrictions cannot be applied to ADF Faces components, such as `Splitter` and `showDetailItem`.

This section contains the following subsections:

- Section 5.8.2.1, "How to Configure ADF Security"
- Section 5.8.2.2, "How to Define Users and Grant Roles in the jazn-data.xml File"
- Section 5.8.2.3, "How to Customize the SessionOptions Object to Include Customization Policy"
- Section 5.8.2.4, "How to Register the Implementation with Oracle Composer"
- Section 5.8.2.5, "How to Configure WebCenterComposerFilter"
- Section 5.8.2.6, "How to Apply an Instance-Level Customization Restriction"
- Section 5.8.2.7, "What Happens at Runtime"

5.8.2.1 How to Configure ADF Security

Before applying an instance-level customization restriction, you must secure your application using ADF security. For information, see Section 3.5.1, "How to Configure ADF Security."

5.8.2.2 How to Define Users and Grant Roles in the jazn-data.xml File

In a previous section, a customization restriction is applied to a `Panel Customizable` component so that, at runtime, only the user with `admin` privileges can access all the features of `Panel Customizable`, and therefore, can customize its attributes.

In this section, two users are created, `admin` and `customer`, and roles are granted. All permissions are granted to both users. The `admin` user can customize the `Panel Customizable` (`panelCustomizable1`) component. However, because of the customization restriction you applied in an earlier section, `customer` does not have the rights to customize `panelCustomizable1`, and this user cannot use all the features of `panelCustomizable1`.

To define users in the jazn-data.xml file:

1. In the Application Resources, right-click the **jazn-data.xml** file and select **Properties**.

   The Edit JPS Identity & Policy Store dialog box opens.
2. In Identity Store, under jazn.com, select **Users**.
3. Click **Add**.

   The **Add User** dialog box opens.
4. In the **Name** field, specify a user name, for example, `admin`.
5. In the **Credentials** field, specify the password, for example, `password`.
6. Click **OK**.
7. Repeat steps 3 through 6 to add another user, for example, `customer`.
8. To grant roles, double-click the **jazn-data.xml** file to open it.
9. Select the **Web Pages** tab.
10. Grant all the permissions (View, Customize, Edit, Grant, Personalize) to both, `admin` and `customer` users.

For more information about creating users, see the section on *enforcing ADF Security in a Fusion Web application* in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

5.8.2.3 How to Customize the SessionOptions Object to Include Customization Policy

To enable the expected runtime behavior of the `Panel Customizable` component, where only the user with `admin` role has access to all capabilities of the `Panel Customizable`, including the ability to customize it, the MDS session for this request must include the user roles in its customization policy. To achieve this, the following customization policy is included in the `SessionOptions` object:

```
SecurityContext stx = ADFContext.getCurrent().getSecurityContext();
  cPol = new CustomizationPolicy(stx.getUserRoles());
```

Then the MDS session is created with this customization policy.

To implement the ComposerSessionOptionsFactory class:

1. From the File menu, choose **New**.
2. In the New Gallery dialog, expand **General**, select **Java**, then **Java Class**, and click **OK**.

The Create Java Class dialog box opens.

3. In the **Name** field, enter `AppsSessionOptionsFactoryImpl`.
4. Click **OK.**

   The `AppsSessionOptionsFactoryImpl.java` file is rendered in the Source view.
5. Import the following libraries:

```
import oracle.adf.share.ADFContext;
import oracle.adf.share.security.SecurityContext;
import oracle.adf.view.page.editor.mds.ComposerSessionOptionsFactory;
import oracle.adf.view.page.editor.mode.ModeContext;
import oracle.mds.config.CustClassListMapping;
import oracle.mds.config.CustConfig;
import oracle.mds.core.SessionOptions;
import oracle.mds.cust.CustClassList;
import oracle.mds.cust.CustomizationClass;
import oracle.mds.cust.CustomizationPolicy;
```

6. Add the following code to implement the `ComposerSessionOptionsFactory` class and provide `SessionOptions`:

```
public class AppsSessionOptionsFactoryImpl
implements ComposerSessionOptionsFactory
{
  public AppsSessionOptionsFactoryImpl()
  {
  }
  public SessionOptions createSessionOptions(SessionOptions sessionOptions,
                                             String mode)
  {
    CustomizationClass[] custLayer;
    CustConfig custConfig = null;
    CustomizationPolicy cPol = null;

    if (ModeContext.EDIT_MODE.equals(mode))
    {
      //Mode is Edit, change to SiteCC
      custLayer = EDIT_LAYER;
    }
    else
    {
      //Mode is View, change to UserCC + SiteCC
      custLayer = VIEW_LAYER;
    }
    try
    {
      CustClassList custClassList = new CustClassList(custLayer);
      CustClassListMapping custClassListMapping =
        new CustClassListMapping("/", null, null, custClassList);
      custConfig = new CustConfig(new CustClassListMapping[]
            { custClassListMapping });
      SecurityContext stx = ADFContext.getCurrent().getSecurityContext();
      cPol = new CustomizationPolicy(stx.getUserRoles());
    }
    catch (Exception e)
    {
      e.printStackTrace();
    }
    if(sessionOptions.getServletContextAsObject() != null)
    {
      return new SessionOptions(sessionOptions.getIsolationLevel(),
                                sessionOptions.getLocale(), custConfig,
                                sessionOptions.getVersionContext(),
                                sessionOptions.getVersionCreatorName(),
                                cPol == null ?
sessionOptions.getCustomizationPolicy() : cPol,
                                sessionOptions.getServletContextAsObject());
    }
    else
    {
      return new SessionOptions(sessionOptions.getIsolationLevel(),
                                sessionOptions.getLocale(), custConfig,
                                sessionOptions.getVersionContext(),
                                sessionOptions.getVersionCreatorName(),
                                cPol == null ?
sessionOptions.getCustomizationPolicy() : cPol);
    }
  }
  //Edit mode SiteCC
  private static final CustomizationClass[] EDIT_LAYER =
    new CustomizationClass[]
    { new SiteCC() };
  //View mode SiteCC + USerCC
  private static final CustomizationClass[] VIEW_LAYER =
    new CustomizationClass[]
    { new SiteCC(), new UserCC() };
}
```

5.8.2.4 How to Register the Implementation with Oracle Composer

To register the implementation with Oracle Composer, perform the steps described in Section 5.6.1.5, "How to Register the Implementation with Oracle Composer."

5.8.2.5 How to Configure WebCenterComposerFilter

To configure `WebCenterComposerFilter`, perform the steps described in Section 5.6.1.6, "How to Configure WebCenterComposerFilter."

5.8.2.6 How to Apply an Instance-Level Customization Restriction

In this section, you apply a customization restriction to a `Panel Customizable` component. At runtime, this restriction ensures that only the user with `admin` privileges can access all the features of the `Panel Customizable`, including customizing its attributes. Users who do not have `admin` privileges can use only a limited set of component features.

To apply an instance-level customization restriction:

1. In the JSPX page, add a **Command Link**.
2. In the Command Link (`af:commandLink`) component, add the `#{appModeBean.edit}` action listener you created in Section 5.6.1.7, "How to Redirect the Servlet to Enable Switch Between MDS Customization Layers."
3. Add a **Page Customizable** component.
4. Apply an instance-level customization restriction on the component `panelCustomizable1` in the Page Customizable.

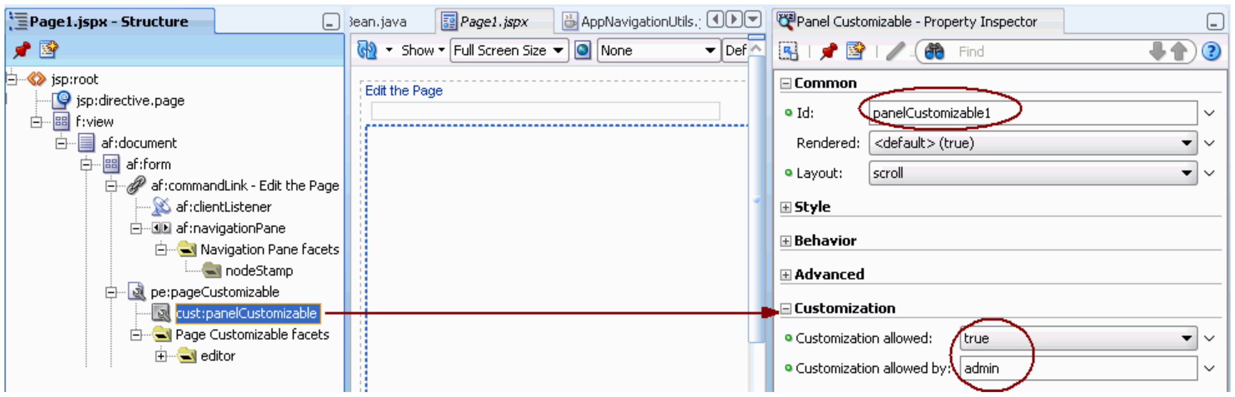
This restriction is stored in the following directory: `public_html/mdssys/mdx/<pagename>.jspx.rdf`.

The Source view should look like this:

```
<af:commandLink text="Edit the Page" id="cl1"
                actionListener="#{appModeBean.edit}">
  <af:clientListener method="togglePEPageMode" type="action"/>
  <af:navigationPane id="np1"/>
</af:commandLink>
<pe:pageCustomizable id="pageCustomizable1">
  <cust:panelCustomizable id="panelCustomizable1" layout="scroll"/>
    <f:facet name="editor">
      <pe:pageEditorPanel id="pep1"/>
    </f:facet>
</pe:pageCustomizable>
<trh:script text="
  function togglePEPageMode(event) { ComposerUtils.toggleMode(event); }
">
</trh:script>
```

The design view should look like Figure 5–5.

Figure 5–5 Design View: Customization Allowed



Id: panelCustomizable1
Rendered: <default> (true)
Layout: scroll
Customization allowed: true
Customization allowed by: admin

Note: For detailed information on how to add Oracle Composer components, see Section 3.3.2, "How to Create a Page in a Manually-Created WebCenter Application."

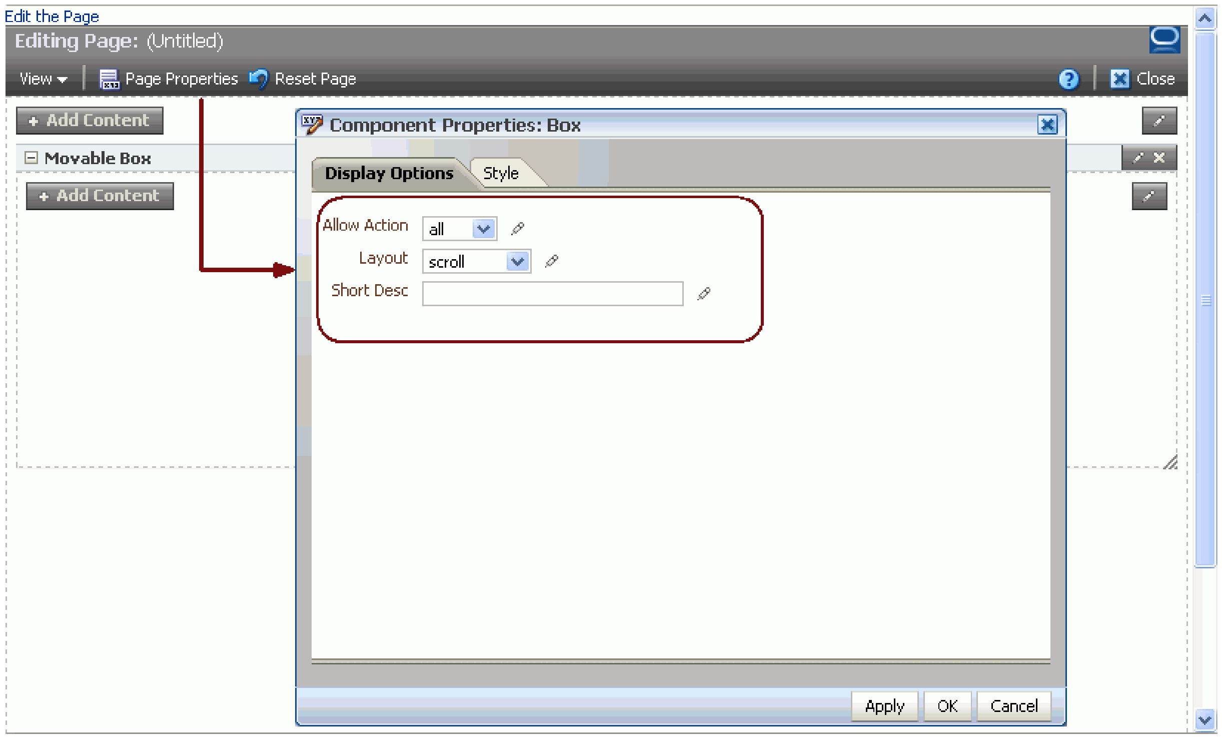
5.8.2.7 What Happens at Runtime

What happens at runtime is influenced by which user is logged on. To see the difference, log in as one user and then the other. For example:

In the Application Navigator, right-click the JSPX page and select **Run**.

Log in to the page as `admin`, and switch to Edit mode. The **Add Content** button and the *pencil* icon are displayed on the outer `panelCustomizable`. As `admin`, you can switch to Source view and access the properties of the outer `panelCustomizable`, as shown in Figure 5–6. That is, the user `admin` can edit this component.

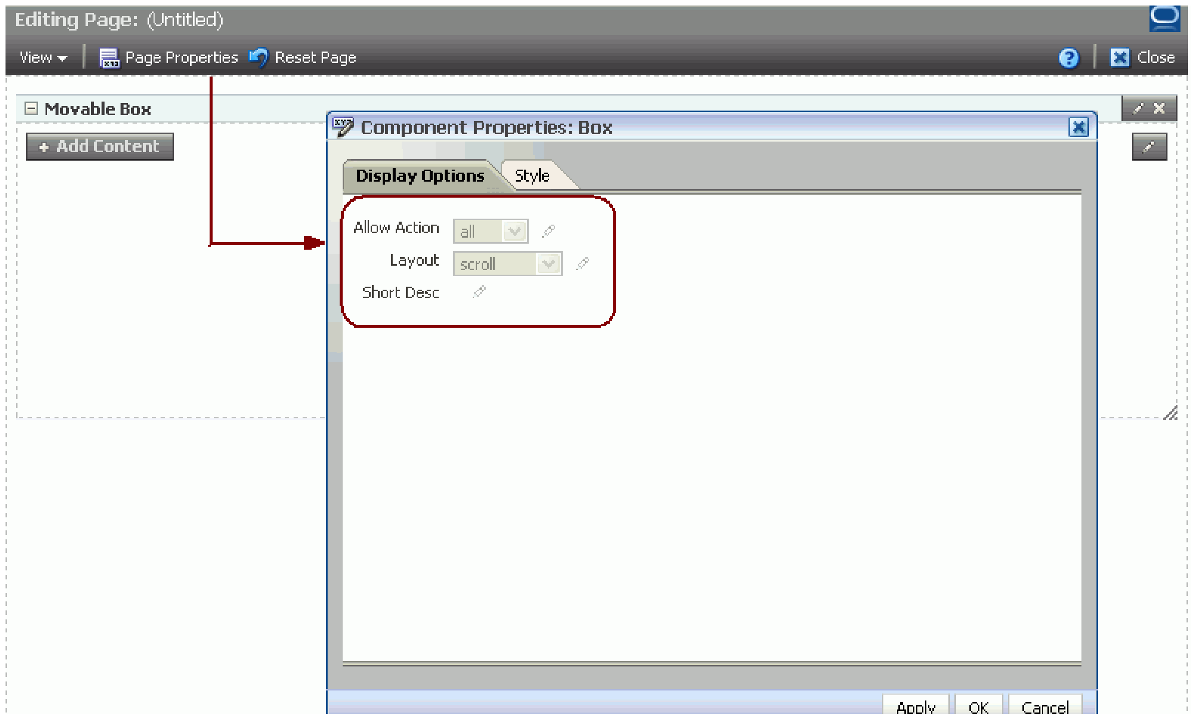
Figure 5–6 Edit Mode for admin



Allow Action: all
Layout: scroll
Short Desc:

Log in to the page as `customer`, and switch to Edit mode. The **Add Content** button and the *pencil* icon are not available on the outer `Panel Customizable`. That is, this component is not editable for `customer`, as shown in Figure 5–7.

Figure 5–7 Edit Mode for customer



Allow Action: all
Layout: scroll
Short Desc

5.8.3 Applying Attribute-Level Security

You can choose whether specific component attributes can be customized at runtime. Additionally, you can specify that the changes made to an attribute must be persisted to a persistence store, such as MDS. You can apply attribute-level security in the following ways:

- By defining property filters
- By persisting changes at the component level

This section describes how to persist changes at the component level. It contains teh following subsections:

- Section 5.8.3.1, "How to Define Change Persistence at the Component Level"
- Section 5.8.3.2, "What Happens at Runtime"

For information about applying property filters, see Section 5.5.1, "How to Define Property Filters."

5.8.3.1 How to Define Change Persistence at the Component Level

You can use the `persist-changes` attribute in the `adf-config.xml` file to specify whether changes to a specific attribute must be persisted.

> **Note:** For information about the Oracle Composer-specific configurations you can make in `adf-config.xml`, see Section B.2.2, "adf-config.xml."

Example 5–14 shows a sample `adf-config.xml` file with `persist-changes` attributes defined for some attributes.

Example 5–14 Using the persist-changes Attribute in the adf-config.xml File

```
<adf-config xmlns="http://xmlns.oracle.com/adf/config">
  <adf-faces-config xmlns="http://xmlns.oracle.com/adf/faces/config">
    <persistent-change-manager>
      <persistent-change-manager-class>
       oracle.adf.view.rich.change.MDSDocumentChangeManager
      </persistent-change-manager-class>
    </persistent-change-manager>
    <taglib-config>
      <taglib uri="http://xmlns.oracle.com/adf/pageeditor">
        <tag name="imageLink">
          <attribute name="destination">
            <persist-changes>true</persist-changes>
          </attribute>
        </tag>
        <tag name="customAction">
          <attribute name="text">
            <persist-changes>true</persist-changes>
          </attribute>
          <attribute name="icon">
            <persist-changes>true</persist-changes>
          </attribute>
        </tag>
      </taglib>
    </taglib-config>
  </adf-faces-config>
```

```
</adf-config>
```

5.8.3.2 What Happens at Runtime

If you set `persist-changes` to `true`, then a change made to the component attribute is persisted in any instance of that component. If you set `persist-changes` to `false`, then even if you change the attribute value in the Component Properties dialog at runtime, the change is not saved.

5.8.4 Applying Action-Level Restrictions on Show Detail Component Actions

The ability of a user to perform actions on `Show Detail Frame` components is inherited from page security. Inheritance is based on the value of the application-wide switch, `enableSecurity`, in the `adf-config.xml` file. If you select the WebCenter Application template when you create your application, then the `adf-config.xml` file is located in the `ADF META-INF` folder under `Descriptors` in the Application Resources panel.

This section describes how to apply action-level restrictions on `Show Detail Frame` component actions. It contains the following subsections:

- Section 5.8.4.1, "How to Add an enableSecurity Section to adf-config.xml"
- Section 5.8.4.2, "Defining Security at the Actions Category Level"
- Section 5.8.4.3, "Defining Security at the Actions Level"

5.8.4.1 How to Add an enableSecurity Section to adf-config.xml

The `enableSecurity` element is not available by default in `adf-config.xml`. To override or extend the page-level security inheritance for `Show Detail Frame` components, you must add the `customizableComponentsSecurity` section in the `adf-config.xml` file, as shown in Example 5–15, and set the `enableSecurity` element in that section to `true`.

Example 5–15 enableSecurity Element in the Customizable Components Security Section in adf-config.xml

```
<cust:customizableComponentsSecurity
xmlns="http://xmlns.oracle.com/adf/faces/customizable/config">
  <cust:enableSecurity value="true"/>
    <cust:actionsCategory>
       ..........................................
</cust:customizableComponentsSecurity>
```

Restrictions on actions on `Show Detail Frame` components can be implemented at the following levels:

- **Page level**: You can define security for customizable components such that page-level privileges are inherited by these components. This is the default behavior.

  By default, customizable components inherit allowable actions from the defined page-level permissions, such as edit, personalize, or customize. That is, a user who has edit, personalize, or customize privileges on a page can perform View mode personalizations on that page. The `enableSecurity` element enables you to override the security inheritance behavior. It can take either of the following values:

  – `true`: If set to `true` (the default when not specified), then the ability of a user to modify a component is first determined from the page permissions and then adjusted according to the current set of actions defined for that type of permission. For example, if a user has customize permission, then the actions that constitute the customize category (move, customize, and so on) are available to the user, but they are overridden by the actions that are defined in the `adf-config.xml` file.

– `false`: If set to `false`, then all actions are available to users. A user's page permissions and actions configured in `adf-config.xml` are ignored.

- **Actions category level**: You can define security on all actions for `Show Detail Frame` components.

  You can add an `actionsCategory` element in the `adf-config.xml` file to define security on multiple actions simultaneously. Depending on the `actionCategory` attributes that you enable, appropriate privileges are provided on the `Show Detail Frame` components.

- **Actions level**: You can define security on individual actions for `Show Detail Frame` components.

  You can use the `actions` element in the `adf-config.xml` file to enable or disable individual actions. Depending on the `actions` attributes that you enable, appropriate privileges are provided on the `Show Detail Frame` components.

Notes:

- Privileges can be inherited from the parent only. Inheritance from a component in any other position in the hierarchy is not supported
- Settings made at the actions category level or actions level are applicable to all instances of the `Show Detail Frame` component across the application. These settings cannot be made for a single instance of a `Show Detail Frame` component.

You can define security for actions on `Show Detail Frame` components at the application level if `enableSecurity` is set to `true` in the `adf-config.xml` file. A value of `true` implies that permission checks are made in addition to the `actionsCategory` and `actions` values specified in the `adf-config.xml`.

5.8.4.2 Defining Security at the Actions Category Level

You can add an `actionsCategory` element in the customizable components actions security section in the `adf-config.xml` file to define security on multiple `Show Detail Frame` components actions. Depending on the `actionsCategory` attributes that you enable, appropriate privileges are provided on the `Show Detail Frame` components.

The `actions` and `actionsCategory` elements in the `adf-config.xml` file have certain default mappings. Table 5–2 describes the different `actionsCategory` attributes and the actions they support by default.

Table 5–2 Actions Categories and Show Detail Frame Actions Mapping

actionsCategory	Actions Supported
`personalizeActionsCategory`	showMoveAction showRemoveAction showMinimizeAction showResizer allowAction
`customizeActionsCategory`	showEditAction showAddContentAction

Note: For information about the Oracle Composer-specific configurations you can make in `adf-config.xml`, see Section B.2.2, "adf-config.xml."

Example 5–16 shows the `actionsCategory` entry that you can add to the customizable components actions security section in the `adf-config.xml` file.

Example 5–16 actionsCategory Element in the Customizable Components Security Section

```
<cust:customizableComponentsSecurity
xmlns="http://xmlns.oracle.com/adf/faces/customizable/config">
  <cust:enableSecurity value="true"/>

  <cust:actionsCategory>
    <cust:actionCategory name="personalizeActionsCategory" value="false"/>
    <cust:actionCategory name="customizeActionsCategory" value="true"/>
  </cust:actionsCategory>

  <cust:actions>
  ..........................................
  </cust:actions>

</cust:customizableComponentsSecurity>
```

You can also use EL for these element values, as shown in Example 5–17.

Example 5–17 EL Used in Customizable Components an actionCategory Entry

```
<cust:actionsCategory>
  <cust:actionCategory name="personalizeActionsCategory"
value="#{appBusinessRules.InsideCorpNetwork}"/>
</cust:actionsCategory>
```

For reference, the managed bean, `appBusinessRules`, is defined as shown in Example 24–7 in Section 24.10, "Overriding Inherited Security on Portlets and Customizable Components."

5.8.4.3 Defining Security at the Actions Level

You can use the `actions` element in the customizable components actions security section of the `adf-config.xml` file to enable or disable individual `Show Detail`

`Frame` actions. Depending on the `actions` attributes that you enable, appropriate privileges are provided on the `Show Detail Frame` components.

> **Note:** For information about the Oracle Composer-specific configurations you can make in `adf-config.xml`, see Section B.2.2, "adf-config.xml."

Example 5–18 shows the `actions` entry that you can add to the customizable components actions section of the `adf-config.xml` file. You can use EL for element values.

Example 5–18 action Elements in the Customizable Components Security Section

```
<cust:customizableComponentsSecurity
xmlns="http://xmlns.oracle.com/adf/faces/customizable/config">
  <cust:enableSecurity value="true"/>

  <cust:actionsCategory>
    .........................................
  </cust:actionsCategory>

  <cust:actions>
    <cust:action name="showMinimizeAction" value="true"/>
    <cust:action name="showMoveAction" value="false"/>
  </cust:actions>

</cust:customizableComponentsSecurity>
```

5.9 Disabling Source View for the Application

By default, Source view is enabled in applications. You can disable Source view if you want to prevent users from being able to edit any page components other than task flows, portlets, and layout components. This section describes how. It contains the following subsections:

- Section 5.9.1, "How to Disable Source View"
- Section 5.9.2, "What Happens at Runtime"

> **Note:** For information about the editing capabilities in Source view, see Section 4.1.3, "Editing Capabilities in Edit Mode."

5.9.1 How to Disable Source View

You can disable Source view by setting the `enable-source-view` entry to `false` in the application's `adf-config.xml` file.

> **Note:** For information about the Oracle Composer-specific configurations you can make in `adf-config.xml`, see Section B.2.2, "adf-config.xml."

To disable Source view:

1. Open the application's `adf-config.xml` file, located in the `ADF META-INF` folder under `Descriptors` in the Application Resources panel.
2. Add the `enable-source-view` property and set its value to `false`, as shown in Example 5–19.

Example 5–19 Disabling Source View in adf-config.xml

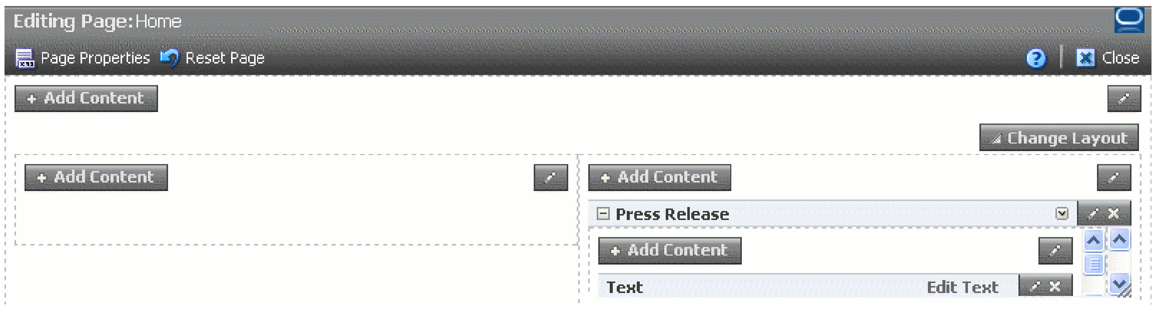
```
<page-editor-config xmlns="http://xmlns.oracle.com/adf/pageeditor/config">
  <enable-source-view>false</enable-source-view>
</page-editor-config>
```

3. Save the file.

5.9.2 What Happens at Runtime

At runtime, when you switch to page Edit mode, the page is rendered in Design view and the View menu is not displayed to the user, as shown in Figure 5–8.

Figure 5–8 Design View of Page without View Menu



5.10 Troubleshooting Oracle Composer Problems

This section provides information to assist you in troubleshooting problems you may encounter while using Oracle Composer.

Problem

Oracle Composer is not working on ADF application pages. Errors are reported in the logs when using Oracle Composer's Resource Catalog or Component Properties dialog. None of the customizations are saved. Objects are not getting added from the Resource Catalog.

Solution

Oracle Composer is compatible only with the change manager, `MDSDocumentChangeManager`. You must set the `CHANGE_PERSISTENCE` context parameter in the `web.xml` file to `MDSDocumentChangeManager`, as shown in the following example:

```
<context-param>
 <param-name>org.apache.myfaces.trinidad.CHANGE_PERSISTENCE</param-name>
 <param-value>oracle.adf.view.rich.change.MDSDocumentChangeManager</param-value>
</context-param>
```

If you use the `FilteredPersistenceChangeManage` change manager, then Oracle Composer cannot write any customizations.

If you configure Oracle Composer first, and then ADF Faces, then the value of `CHANGE_PERSISTENCE` is changed to `oracle.adf.view.rich.change.FilteredPersistenceChangeManager`.

You must change this value to `oracle.adf.view.rich.change.MDSDocumentChangeManager`.

If you configure ADF Faces first, and then Oracle Composer, the `CHANGE_PERSISTENCE` is automatically set to `MDSDocumentChangeManager`.

Problem

When you run a page, the following error displays:

```
java.lang.IllegalStateException: The expression
"#{bindings.pageeditorpanel.regionModel}" (that was specified for the RegionModel
"value" attribute of the region
component with id "pePanel") evaluated to null. This is typically due to an error
in the configuration of the
objects referenced by this expression. If it helps, the expression
"#{bindings.pageeditorpanel}" evaluates to "null".
If it helps, the expression "#{bindings}" evaluates to "view_untitled1PageDef".
Now using an empty RegionModel instead.
```

Solution

This error occurs if a page containing the `Page Customizable` component does not have the required task flow binding in its page definition file. Ensure that the page definition file contains the following valid entry under the `<executables>` node:

```
<taskFlow id="pageeditorpanel"
taskFlowId="#{pageEditorBean.pageEditorPanel}"xmlns="http://xmlns.oracle.com/adf/c
ontroller/binding"/>
```

This error may also occur if your page is based on a page template and that page template contains the `Page Customizable` component. In such a case, the `af:pageTemplate` tag does not contain the `value="#{bindings.pageTemplateBinding}"` attribute.

Ensure that the page definition file has the following entry under the `<executables>` node:

```
<page path="view.pageDefs.templateDef1PageDef"
id="pageTemplateBinding" Refresh="ifNeeded"/>
```

Problem

Users cannot switch to Edit mode. The Edit link (`Change Mode Link` or `Change Mode Button`) appears disabled.

Solution

The user may have only view privilege on the page. Ensure that the user has the edit privilege on the page. For the page template on which the page is based, it is sufficient to have only view privilege.

Problem

Resource Catalog is empty.

Solution

The default catalog is not available to MDS. Ensure that the deployment profile contains the necessary entries to copy the default catalog file. Also, ensure that in the MDS section of `adf-config.xml`, a namespace entry points to the default catalog file.

Problem

A project task flow does not appear in the Resource Catalog.

Solution

The task flow must be packaged as an ADF Library (JAR file) and added to the project. You must also ensure that the task flow ID is given in the following format:

```
path_to_jar/ADF_TaskFlow/task_flow_path
Ex. doclib-service-view.jar/ADF_
TaskFlow/oracle+webcenter+doclib+view+jsf+taskflows+mainView.xml#doclib-document-l
ibrary
```

Problem

When you view the properties of a component in the Component Properties dialog, properties appear disabled and you cannot edit property values.

Solution

Ensure that the component is not restricted by using MDS. For information, see Section 5.6.2, "Applying Tag-Level Security Using the customizationAllowed Attribute."

Problem

You added a global- or instance-level `Custom Action`. However, it neither displays on the chrome or in the **Action** menu on the `Show Detail Frame`.

Solution

Ensure the following:

- The first child of the `Show Detail Frame` must be `af:region`.
- The view currently displayed on the task flow must have an outcome in its task flow definition file that matches the action of `Custom Action`.
- If the action has the prefix `dialog:`, the outcome in the task flow definition must also have the same prefix.

For information, see Section 4.2.13, "How to Enable Custom Actions on a Show Detail Frame Enclosing a Task Flow."

Problem

Your application is configured to use MDS sandbox. When you run the application, the sandbox either does not work or generates exceptions.

Solution

Ensure the following:

- A database repository is used. Sandbox works only with a database-based repository, and not with a file-based repository.

- Ensure that the order of filters in `web.xml` is correct. The `<filter>` entries must be in the correct order for sandbox to work correctly.

For information, see Section 5.7, "Using Oracle Composer Sandbox."

Problem

You created an add-on, but it does not appear on the Oracle Composer toolbar.

Solution

Ensure the following:

- The `pe_ext.xml` file is in `/META-INF` folder in a JAR file or the application and is available in the class path.
- The task flow binding ID specified while registering the panel in `pe_ext.xml` is correct.
- An `addon-panel` entry exists under the `page-editor-config` section in the `adf-config.xml` file.
- If the add-on is in a JAR file, ensure that the JAR file is created as an ADF Library.

For information about add-ons, see Section 5.2, "Creating Oracle Composer Add-Ons."

Problem

You have registered a custom property panel. However, it does not appear in the Component Properties dialog when your select a component to display its properties.

Solution

Ensure the following:

- The `pe_ext.xml` file is in `/META-INF` folder in a JAR file or the application and is available in the class path.
- The task flow binding ID specified while registering the panel in `pe_ext.xml` is correct.
- The `property-panel` registration is correct and is specified against the component you want the panel to appear against. Further, the fully qualified class name of the component is correctly specified using the `component` node.
- A duplicate property panel registration is not overriding your panel entry.
- If the panel is configured to use the rendered attribute, ensure that the value or EL evaluates to `true`.
- If the add-on is in a JAR file, ensure that the JAR file is created as an ADF Library.

For information about custom property panels, see Section 5.3, "Creating Custom Property Panels."

Problem

The `Show Detail Frame` and `Panel Customizable` do not show the Edit icon, or the `Panel Customizable` does not show the **Add Content** button. Also, you are not able to move the `Show Detail Frame` out of a `Panel Customizable` component or drop it into another `Panel Customizable` component.

Solution

Ensure that the `Panel Customizable` is not restricted by using MDS and the component actions are not secured by using entries in `adf-config.xml`.For information, see Section 5.8, "Overriding Default Security Behavior of Oracle Composer Components."

6 Configuring the Resource Catalog for Oracle Composer

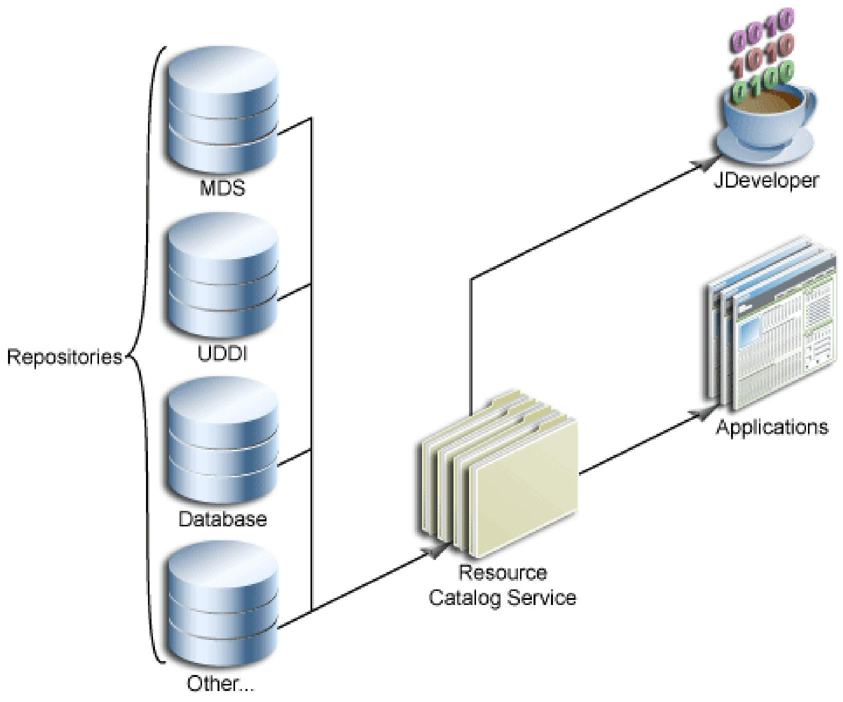
This chapter describes the default Resource Catalog in your application and explains how to customize the runtime Resource Catalog to display different content to different users.

This chapter contains the following sections:

6.1 Overview of Resource Catalog

The Resource Catalog provides a consolidated view of the contents of one or more otherwise unrelated repositories in a unified search and browse user interface. Resources originate in their source repository and are then exposed to the developer through the Resource Catalog as shown in Figure 6–1.

Figure 6–1 Resource Catalog Overview



When using JDeveloper, you see two types of Resource Catalogs—design-time catalogs and runtime catalogs.

Design time catalogs are created in the Resource Palette and are like favorites lists for developers. You can define connections and then create catalogs to organize resources exposed by those connections. Those resources can then be re-used in any application you are developing.

> **See Also:** Oracle JDeveloper online Help for more information about the Resource Palette and catalogs.
>
> To locate this information in Oracle JDeveloper, from the **Help** menu, select **Table of Contents**. In Help Center, expand **JDeveloper Basics** and select **Working with the Resource Palette**.

Runtime catalogs are artifacts of an application, like JSPX pages, that are used only at runtime. These catalogs determine which components, task flows, and portlets are available for placing on a page in Edit mode. For information about adding Oracle Composer components to a page, see Section 4.2, "Designing Editable Pages Using Oracle Composer Components."

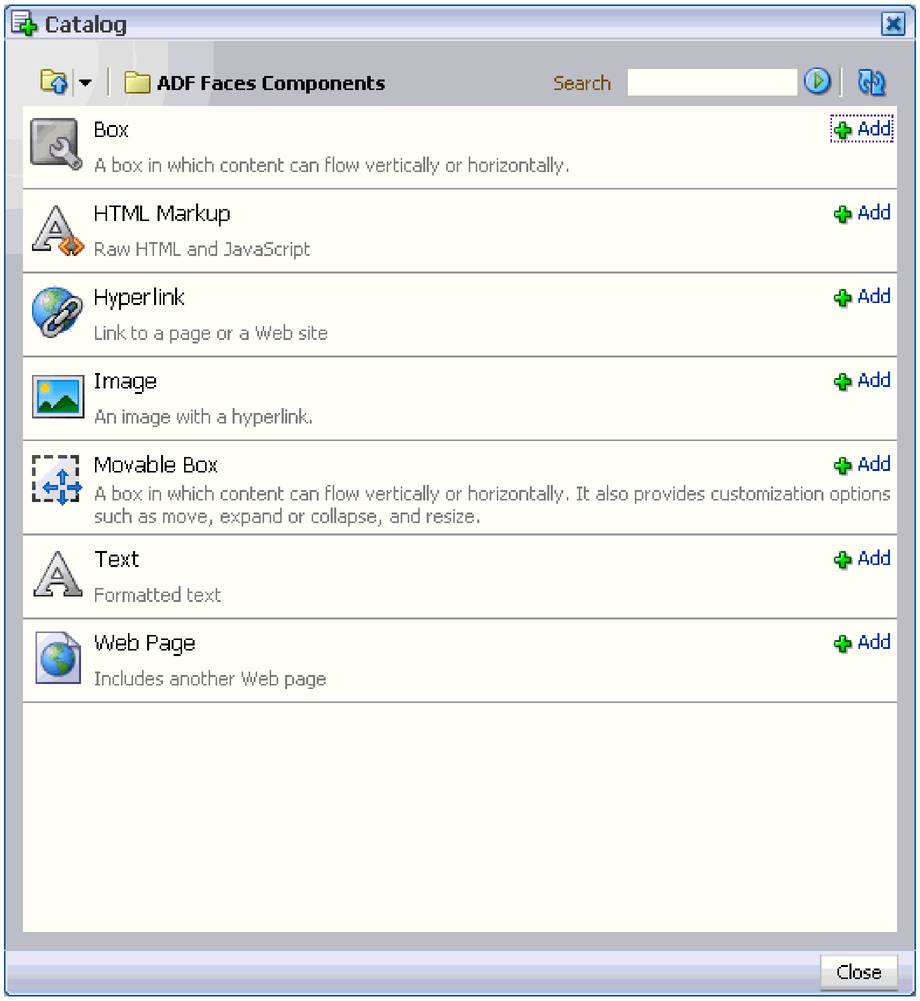
Default Resource Catalog Configuration

When you add a `Page Customizable` component to the page, the following configurations occur automatically:

- A default Resource Catalog definition file, `default-catalog.xml`, is configured in the application. The `default-catalog.xml` file is located in the `<`*`Application_Root`*`>\mds\oracle\adf\rc\metadata` directory. Catalog definitions are XML files that specify the contents of the catalog.
- A default Resource Catalog is configured for the application. At runtime, Oracle Composer provides an option to add content to the page from this Resource Catalog.

The default Resource Catalog contains the **ADF Faces Components** folder, which contains components that a user can add to the page (Figure 6–2).

Figure 6–2 Components in the Default Resource Catalog



Box
A box in which content can flow vertically or horizontally.
Add

HTML Markup
Raw HTML and JavaScript
Add

Hyperlink
Link to a page or a Web site
Add

Image
An image with a hyperlink.
Add

Movable Box
A box in which content can flow vertically or horizontally. It also provides customization options such as move, expand or collapse, and resize.
Add

Text
Formatted text
Add

Web Page
Includes another Web page
Add

If you have registered a portlet producer with your application, then that producer's portlets are displayed in a Portlets folder in the Resource Catalog. If you have not registered any producer, the Portlets folder is hidden.

You can determine the components that are visible to users by modifying the default Resource Catalog or creating one or more of your own Resource Catalogs. Resource Catalogs can contain the following components:

- **Oracle ADF Faces components**: The ADF Faces Components folder provides `Box`, `Movable Box`, and `Image` components that are analogous to the JDeveloper design time components `Panel Customizable`, `Show Detail Frame`, and `Image Link` respectively. In JDeveloper, these components are available in the Oracle Composer tag library.

  The ADF Faces Components folder also provides `HTML Markup`, `Hyperlink`, `Text`, and `Web Page` components that are analogous to the JDeveloper design time components `HTML`, `Go Link`, `Rich Text Editor`, and `Web Page` respectively. In JDeveloper, these components are available in the ADF Faces tag library.

- **Portlets**: You can add Java-PDK or WSRP portlets from any producer that was registered in Oracle JDeveloper. See Chapter 9, "Consuming Portlets" for detailed information about registering portlet producers and adding portlets to the page.
- **Task Flows**: If you have created task flows in Oracle JDeveloper, you can add those task flows from the JAR files onto your page at runtime.

6.2 Modifying the Content of the Default Resource Catalog

You can determine the contents of your Resource Catalog by modifying the default catalog definition file or creating a new catalog definition file and configuring Oracle Composer to point to it.

This section provides an example that illustrates how to include items of your choice in the default catalog definition file. It contains the following sections:

- Section 6.2.1, "How To Add Items to Your Resource Catalog"
- Section 6.2.2, "What Happens at Runtime"
- Section 6.2.3, "What You May Need to Know When Defining a Resource Catalog"
- Section 6.2.4, "How to Define Task Flow Parameters and Attributes of the Enclosing Show Detail Frames for Task Flows"

6.2.1 How To Add Items to Your Resource Catalog

If you want to modify the contents of the default Resource Catalog, you must edit the default catalog definition file, `default-catalog.xml` and include the components of your choice. This section describes how to add more items to the default Resource Catalog. It contains the following subsections:

- Section 6.2.1.1, "Adding Connections for Resources"
- Section 6.2.1.2, "Defining the Catalog Contents"
- Section 6.2.1.3, "Enabling Task Flows in the Resource Catalog"
- Section 6.2.1.4, "Enabling Documents in the Resource Catalog"

6.2.1.1 Adding Connections for Resources

Many of the resources included in catalogs, such as Web 2.0 Services and portlets, are accessed through connections defined in your application's `connections.xml` file.

You can create connections to resources in different ways. This section describes how to create a connection from the Resource Palette. For information about additional ways to create connections, see Section 3.9, "Accessing Connection Wizards."

To create a connection from the Resource Palette:

1. From the Resource Palette, click the **New** icon in the upper left corner.
2. Choose **New Connection** and then the type of connection you want to create.

   The wizard for your connection type appears.
3. Step through the wizard filling in the necessary information about the connection.

   When you click **Finish**, the connection should appear in the Resource Palette under **Connections**.
4. Right click your connection in the Resource Palette and choose **Add to Application** from the context menu.

Alternatively, you can drag and drop the connection from the Resource Palette to the Application Resources panel of the Application Navigator.

6.2.1.2 Defining the Catalog Contents

If you modify the default catalog definition file, your modifications appear in the catalog at runtime. The default catalog definition file, `default-catalog.xml`, is available in the `<Application_Root>\mds\oracle\adf\rc\metadata` directory.

You can modify the catalog content according to the rules of catalog definitions. Refer to the following for assistance in modifying the definition:

- Section C.2, "Default Catalog Definition"
- Section C.3, "XML Schema"
- Section C.4, "Catalog Definition Attributes"

The default catalog definition contains commented entries for displaying documents from the Documents service and out-of-the-box service task flows. If you want the Resource Catalog to display these components, you must enable relevant entries in the catalog definition file.

Example

This example describes the procedure for adding a `Discussions` folder to the Resource Catalog. The Discussions folder contains the Discussions service main view task flow as a resource.

To expose the new `Discussions` folder in the Resource Catalog, you must add the code shown in **bold** in Example 6–1 to the `default-catalog.xml` file.

Example 6–1 Modified Sample default-catalog.xml File

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<catalogDefinition xmlns="http://xmlns.oracle.com/adf/rcs/catalog"
                   id="catalogDefinition"
                   name="Default Resource Catalog"
                   description="Default resource catalog definition containing 
sample entries">
  <contents>

    <!-- ***********************************************************************
    * Custom folder exposing ADF Faces components                            *
    * Comment out this element if you want to remove it from the catalog.     *
    ******************************************************************** -->
    <customFolder id="facesComponents" name="ADF Faces Components"
                  description="ADF Faces components you can add to application 
pages"
factoryClass="oracle.adfinternal.view.page.editor.componentcatalog.adapter.Compone
ntObjectFactory"/>

    <!-- ***********************************************************************
    * Dynamically include the portlets custom folder into the catalog at      *
    * runtime. The custom folder will only be included if the portlet runtime *
    * jar files are included in the application classpath.                    *
    ******************************************************************** -->
    <customContent id="portletContent"
 contentProviderClass="oracle.adf.rc.webcenter.WebCenterContentProvider"/>

    <!-- ***********************************************************************
    * This entry is added to modify the default-catalog.xml                 *
    * In this entry  resource is publised within a folder.
*
    ******************************************************************* -->
   <folder name="Discussions" id="mydiscussions">
     <contents>
       <resource id="discussionsMainView"
               name="Discussions"
               description="Main view of the Discussions Service"
               repository="application.classpath"
               path="forum-view.jar/ADF_
TaskFlow/oracle+webcenter+collab+forum+view+taskflows+main-task-flow.xml#forum-mai
n"/>
     </contents>
 </folder>

    <!-- *******************************************************************
    * Uncomment the following <customFolder> element if you have configured   *
    * the Document Library service in your application and want to include    *
    * documents in the catalog.                                               *
    ******************************************************************* -->
<!--
    <customFolder id="doclibDocuments"
                  name="Documents"
                  description="Documents from the Document Library Service"
   factoryClass="oracle.webcenter.content.model.rc.CustomFolderContextFactory"/>
-->

    <!-- *******************************************************************
    * Uncomment the following <resource> element if you have configured the   *
    * Document Library service in your application and want to include the    *
    * Document Library main view in the catalog.                              *
    *                                                                         *
    * NOTE: application.classpath is an implicitly defined "repository" for   *
    *       accessing ADF Libraries that are included in your classpath       *
    ******************************************************************* -->
<!--
    <resource id="doclibMainView"
              name="Document Library Task Flow"
              description="Main view of the Document Library Service"
              repository="application.classpath"
              path="doclib-service-view.jar/ADF_
TaskFlow/oracle+webcenter+doclib+view+jsf+taskflows+mainView.xml#doclib-document-l
ibrary"/>
-->

    <!-- *******************************************************************
    * To create a link to a task flow in an ADF Library:                      *
    * 1) add a <resource> tag to your catalog definition & set the            *
    *    "repository" attribute to "application.classpath"                    *
    *    (see doclibMainView above)                                           *
    * 3) set the "repository" attribute to the name of your file system       *
    *    connection.                                                          *
    * 4) set the "path" attribute for the task flow you are interested in.    *
    *    The path is of the following form:                                   *
    *                                                                         *
    *    path_to_jar/ADF_TaskFlow/task_flow_path                              *
    *                                                                         *
    * To obtain the task_flow_path, use the file system connection from (1)   *
    * and navigate to the task flow in your ADF Library. Mouse-over your task *
    * flow so its tool-tip is displayed. The tooltip shows the fully qualified*
    * ID of the task flow. Take this value and replace all occurrances of "/" *
    * with "+". For example:                                                  *
    *                                                                         *
    * Tool-tip:
oracle/webcenter/collab/announcement/view/taskflows/main-view-definition.xml#annou
ncement-main-view
    * Task_flow_path:
oracle+webcenter+collab+announcement+view+taskflows+main-view-definition.xml#annou
ncement-main-view
    ********************************************************************** -->

    </contents>
</catalogDefinition>
```

After you add the specified code, the Discussions folder is displayed in the runtime Resource Catalog.

To run and test the page:

1. Run any editable JSPX page in your application.
2. Switch to page Edit mode.
3. Click an **Add Content** button on the page to view the Discussions folder in the Catalog dialog.

6.2.1.3 Enabling Task Flows in the Resource Catalog

If you want the Resource Catalog to display an out-of-the-box service task flow or custom task flow that you created in your application, you must add entries for the task flow in the `default-catalog.xml` file. Custom task flows must additionally be packaged in an ADF library, even if they are part of the same application.

This section describes how to package a custom task flow in an ADF library and how to enable the display of task flows in the Resource Catalog. It contains the following subsections:

- Section 6.2.1.3.1, "How to Package a Custom Task Flow in an ADF Library"
- Section 6.2.1.3.2, "How to Enable the Display of Task Flows in the Resource Catalog"

6.2.1.3.1 How to Package a Custom Task Flow in an ADF Library To package a custom task flow in an ADF library:

1. Create a deployment profile for the task flow:
   a. Right-click **ViewController** and choose **New**.
   b. In the New Gallery, expand **General**, select **Deployment Profile**, and then **ADF Library JAR File**, and click **OK**.
   c. In the Create Deployment Profile -- ADF Library JAR File dialog, enter a name for your deployment profile and click **OK**.
   d. In the ADF Library JAR Deployment Profile Properties dialog, click **OK**.
   e. In the Project Properties dialog, click **OK**.
2. In the Application Navigator, right-click the project folder, choose **Deploy**, *deployment profile name*, **to**, and then choose **to ADF Library JAR file**.

This creates a **deploy** folder including the JAR file, in the project folder located at `<Application_Root>\ViewController\deploy\`.

6.2.1.3.2 How to Enable the Display of Task Flows in the Resource Catalog

To enable the display of task flows in the Resource Catalog:

1. Open the `default-catalog.xml` file in the `<Application_Root>\mds\oracle\adf\rc\metadata` directory.
2. Add a `<resource>` element to your catalog definition similar to the following example:

```
<resource id="doclibMainView"
        name="Document Library Task Flow"
        description="Main view of the Document Library Service"
        repository="application.classpath"
        path="doclib-service-view.jar/ADF_
TaskFlow/oracle+webcenter+doclib+view+jsf+taskflows+mainView.xml#doclib-documen
t-library"/>
```

> **Note:** Ensure that the task flow library is in the application class path.

3. Set the `path` attribute for the task flow you want to include in the catalog.

   The path must be specified in the following format:

   `jar_file_name``/ADF_TaskFlow/`*`task_flow_path`*

   where.

   `jar_file_name` is the name of the task flow JAR file. This JAR is picked up from the application classpath.

   `task_flow_path` is the path to the task flow within the JAR file.

   To obtain the task flow path for any task flow exposed in the Resource Palette, including out-of-the-box service task flows, navigate to the task flow in your ADF Library and right-click the task flow. From the context menu, select **Show Catalog Reference**. A message dialog displays the values of the `repository` and `path` attributes for that task flow.
4. Save the `default-catalog.xml` file.

The task flows that you have defined in `default-catalog.xml` are now available in the Resource Catalog.

> **Note:** To enable users to view the task flow's content at runtime, ensure that the task flow you add to the Resource Catalog has a `TaskFlowPermission` grant in the application's `jazn-data.xml` file, with at least `View` action provisioned.

6.2.1.4 Enabling Documents in the Resource Catalog

The default catalog definition file has entries for displaying documents from the Documents service, but these are commented out. Consequently, documents are not available in the default version of the Resource Catalog (see Section C.2, "Default Catalog Definition").

If you have configured the Documents service in your application and want to expose its task flows in the Resource Catalog, you can uncomment Documents service entries in the default catalog definition file.

To enable display of documents:

1. Open the `default-catalog.xml` file in JDeveloper.
2. Uncomment the `<customFolder>` element with an `id` value `doclibDocuments`.
3. Save the file.

Documents from the Documents service now show in a Documents folder in the Resource Catalog.

6.2.2 What Happens at Runtime

After you enable Oracle Composer and define your catalog contents, you can run the page from Oracle JDeveloper. The Resource Catalog displays all elements that you defined in the catalog definition file.

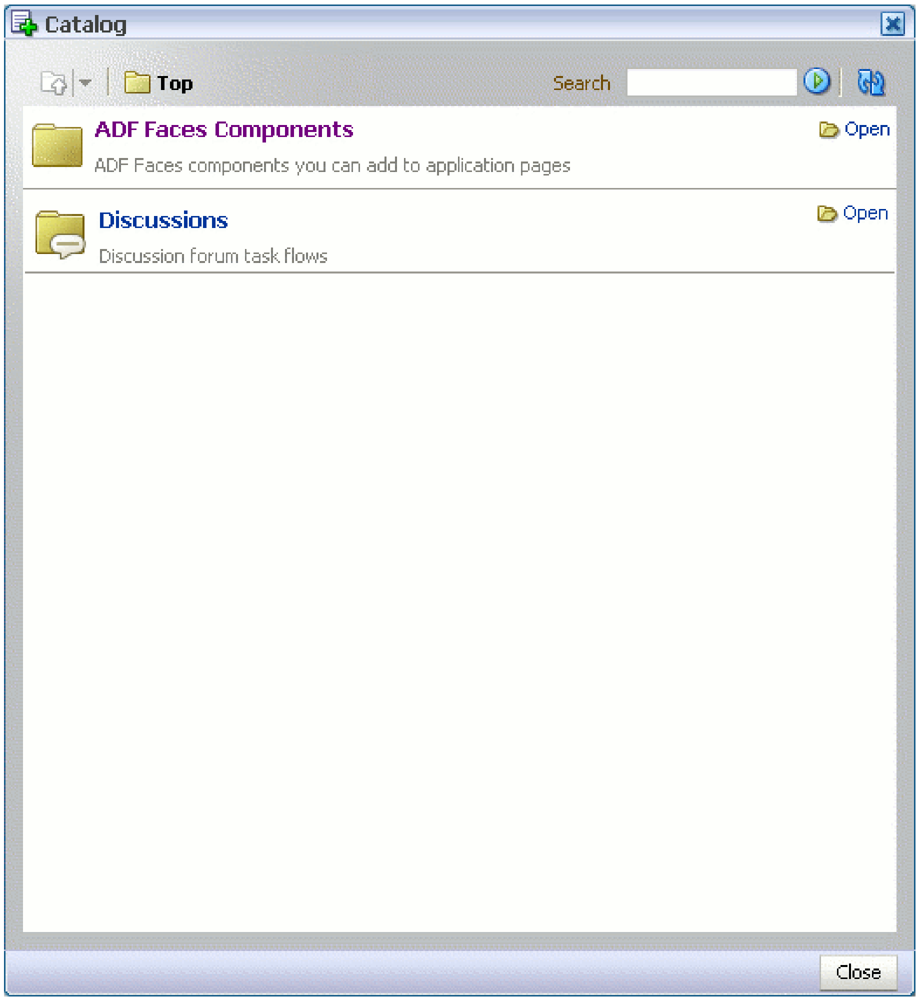
To run and test the page:

1. In the Application Navigator, open your customizable `MyPage.jspx` and run it.
2. Switch to page Edit mode.
3. Click an **Add Content** button on the page to display the Catalog dialog.

   In addition to the ADF Faces Components folder, the Resource Catalog displays all the elements you defined in the catalog definition file at design time.

For example, Figure 6–3 shows the Discussions folder that you enabled in Example 6–1.

Figure 6–3 Modified Resource Catalog with the Discussions Folder



ADF Faces Components
ADF Faces components you can add to application pages

Discussions
Discussion forum task flows

You can look at the sample application, `ModifiedResourceCatalog.jws`, containing a similar modified catalog definition, on the Oracle WebCenter Suite 11g Demonstrations and Samples page on the Oracle Technology Network (OTN) at:

`http://webcenter.oracle.com`

6.2.3 What You May Need to Know When Defining a Resource Catalog

To correctly implement resource catalogs, you must know where the application looks for the definitions and the XML schema upon which the definitions are based. See Appendix B, "Oracle Composer Component Properties and Files" for further information.

6.2.4 How to Define Task Flow Parameters and Attributes of the Enclosing Show Detail Frames for Task Flows

When you add a task flow from the Resource Catalog to your application page, the task flow is automatically enclosed in a `Show Detail Frame` component. To enable runtime editing of attributes associated with the enclosing `Show Detail Frame`, you must first define those attributes in the catalog definition file where the task flow is registered.

For example, if a user adds a task flow to a page with a dark background setting, the task flow continues to use the background setting of the enclosing `Show Detail Frame` component. This may not be the look the user wants. To enable the user to change the `Show Detail Frame`'s background property, you must define this attribute in the catalog definition file. When a user selects the task flow for editing, the Component Properties dialog displays this `Show Detail Frame` attribute along with other task flow attributes.

Similarly, to prepopulate task flow parameters so that users do not have to set parameter values at runtime, you can define those parameters in the catalog definition file using an EL expression. When a user selects the task flow for editing, the Component Properties dialog displays these parameters with prepopulated values.

To define task flow parameters and `Show Detail Frame` attributes:

1. Open the catalog definition file that contains the task flow for which you want to set the parameters and `Show Detail Frame` attributes.
2. Within the task flow's `<resource>` element, add an `<attribute>` entry for each parameter and `Show Detail Frame` attribute you want to define.

   The `attributeId` for a parameter must contain the parameter name prefixed with **`parameter.`**; the `attributeId` for a `Show Detail Frame` attribute must contain the attribute name prefixed with **`attr.`** as shown in the following example:

   ```
   <?xml version = '1.0' encoding = 'UTF-8'?>
   <catalogDefinition xmlns="http://xmlns.oracle.com/adf/rcs/catalog"
                      id="catalogDefinition" name="Default Resource Catalog"
                      description="Default resource catalog definition containing
   sample entries">
     <contents>
       <resource id="sampletaskflow" name="Sample Taskflow"
                 path="sampletaskflow.jar/ADF_
   TaskFlow/WEB-INF+sample-task-flow-definition.xml#sample-task-flow-definition"
                 repository="application.classpath">
         <attributes>
           <attribute value="dark" attributeId="attr.background"
                      isKey="false"/>
           <attribute value="#{myBean.myParam1}" attributeId="parameter.myParam1"
                      isKey="false"/>
         </attributes>
       </resource>
     </contents>
   </catalogDefinition>
   ```

   Note: Retain the default value of `false` for the `isKey` attribute. Set it to `true` only if the value is coming from a resource bundle.

3. Save the catalog definition file.

6.3 Creating a Custom Resource Catalog

You can define the contents of your application's Resource Catalog either by modifying the existing, default catalog definition file, `default-catalog.xml`, or by creating a new catalog definition file. Create a custom Resource Catalog when you do not want to expose the ADF Faces components that are available in the default catalog, but instead want to include components or task flows that you specify.

This section describes how to create a custom Resource Catalog. It contains the following subsections:

- Section 6.3.1, "How to Create a Custom Resource Catalog"
- Section 6.3.2, "What Happens at Runtime"

6.3.1 How to Create a Custom Resource Catalog

To override default settings and display your custom Resource Catalog you must first create a new catalog definition file. Then you must specify the name of the new file in the application's `adf-config.xml` file.

To create a catalog definition:

1. Open your application in Oracle JDeveloper.
2. From **File** menu, choose **New**.
3. In the New Gallery dialog box, expand **General**, select **XML**, then **XML Document**, and click **OK**.
4. In the Create XML File dialog box, enter the file name, for example, `users-catalog.xml`, and the path of your catalog definition file.

   Catalog definition files must be located in the Resource Catalog root directory (<*JDEV_HOME*>`\jdev\mywork\`<*Application_Name*>`\mds\oracle\adf\rc\metadata`) or a subdirectory of the root. For example, the Resource Catalog root might be:

   ```
   C:\JDeveloper\mywork\RCSampleApp\mds\oracle\adf\rc\metadata
   ```

   You could create your catalog here or create a subdirectory for it. For example:

   ```
   C:\JDeveloper\mywork\RCSampleApp\mds\oracle\adf\rc\metadata\mycatalogs
   ```

5. Click **OK**.
6. You can now code your XML file according to the rules of catalog definitions.

   Refer to the following to help you create the definition:

   - Section C.2, "Default Catalog Definition"
   - Section C.3, "XML Schema"
   - Section C.4, "Catalog Definition Attributes"

   > **Tip:** If you do not want to write your catalog definition from the start, you could copy the contents of `default-catalog.xml` into your new catalog definition as a starting point.

   Example 6–2 shows a sample catalog definition file containing the `ADF Faces Components` folder with its layout components, and a `Discussions` folder with the Discussions service task flows.

Example 6–2 The users-catalog.xml File

```
<?xml version="1.0" encoding="UTF-8" ?>
  <catalogDefinition xmlns="http://xmlns.oracle.com/adf/rcs/catalog"
                     id="catalogDefinition"
```

```
                     name="Default Resource Catalog"
                     description="Default resource catalog definition containing
sample entries">
  <contents>
    <customFolder id="facesComponents" name="ADF Faces Components"
        description="ADF Faces components you can add to application pages"

factoryClass="oracle.adfinternal.view.page.editor.componentcatalog.adapter.Compone
ntObjectFactory" />
    <folder id="forumsFolder">
      <attributes>
        <attribute value="Discussions" attributeId="Title" isKey="false" />
        <attribute value="Discussion Forum Taskflows" attributeId="Description"
isKey="false" />
        <attribute value="" attributeId="Subject" isKey="false" />
        <attribute value="/adf/webcenter/folderdiscussions_qualifier.png"
attributeId="IconURI" />
      </attributes>
      <contents>
      <!--  Forums Main View
      -->
        <resource path="forum-view.jar/ADF_
TaskFlow/oracle+webcenter+collab+forum+view+taskflows+main-task-flow.xml#forum-mai
n"
          repository="application.classpath" id="forumMainView">
          <attributes>
            <attribute value="Main View" attributeId="Title" isKey="false" />
            <attribute value="Taskflow with complete view"
                       attributeId="Description" isKey="false" />
            <attribute value="/adf/webcenter/viewtopics_qualifier.png"
attributeId="IconURI" />
            <attribute value="height:211px" attributeId="attr.contentStyle"
isKey="false" />
          </attributes>
        </resource>
      <!--  Watched forum View
      -->
        <resource path="forum-view.jar/ADF_
TaskFlow/oracle+webcenter+collab+forum+view+taskflows+watchedForum-task-flow.xml#f
orum-watchedForum"
               repository="application.classpath" id="forumWatchedForumView">
          <attributes>
            <attribute value="Watched Forums" attributeId="Title" isKey="false" />
            <attribute value="All discussion forums you are watching"
attributeId="Description" isKey="false" />
            <attribute value="/adf/webcenter/viewdiscussions_qualifier.png"
attributeId="IconURI" />
            <attribute value="height:211px" attributeId="attr.contentStyle"
isKey="false" />
          </attributes>
        </resource>
      </contents>
    </folder>
  </contents>
  </catalogDefinition>
```

where:

`<resource>` elements are used to define resources that you can add to the page

`<folder>` elements are used to define folders in the catalog

7. Add an `<rcv-config>` element to register `users-catalog` in your `adf-config.xml` file as shown in the following example:

   ```
   <rcv-config xmlns="http://xmlns.oracle.com/adf/rcs/viewer/adf-config">
     <default-catalog catalog-name="users-catalog"/>
   </rcv:rcv-config>
   ```

   The value for `catalog-name` specifies the name of the catalog definition file to use.

   The name you specify is relative to the Resource Catalog root directory and should not include the file extension `.xml`. For example, if you saved your new catalog definition as `C:\JDeveloper\mywork\RCSampleApp\mds\oracle\adf\rc\metadata\mycatalogs\users-catalog.xml`, then the catalog name is `mycatalogs/users-catalog`. The name uses the `/` separator because it represents part of an MDS path.

   These entries enable you to override the default Resource Catalog. You must add them specifically because they are not included in the `adf-config.xml` file by default.

   Note: For information about the Oracle Composer-specific configurations you can make in `adf-config.xml`, see Section B.2.2, "adf-config.xml."

8. Save `adf-config.xml`.

6.3.2 What Happens at Runtime

When you run your application and edit a page, Oracle Composer now displays content based on your custom catalog.

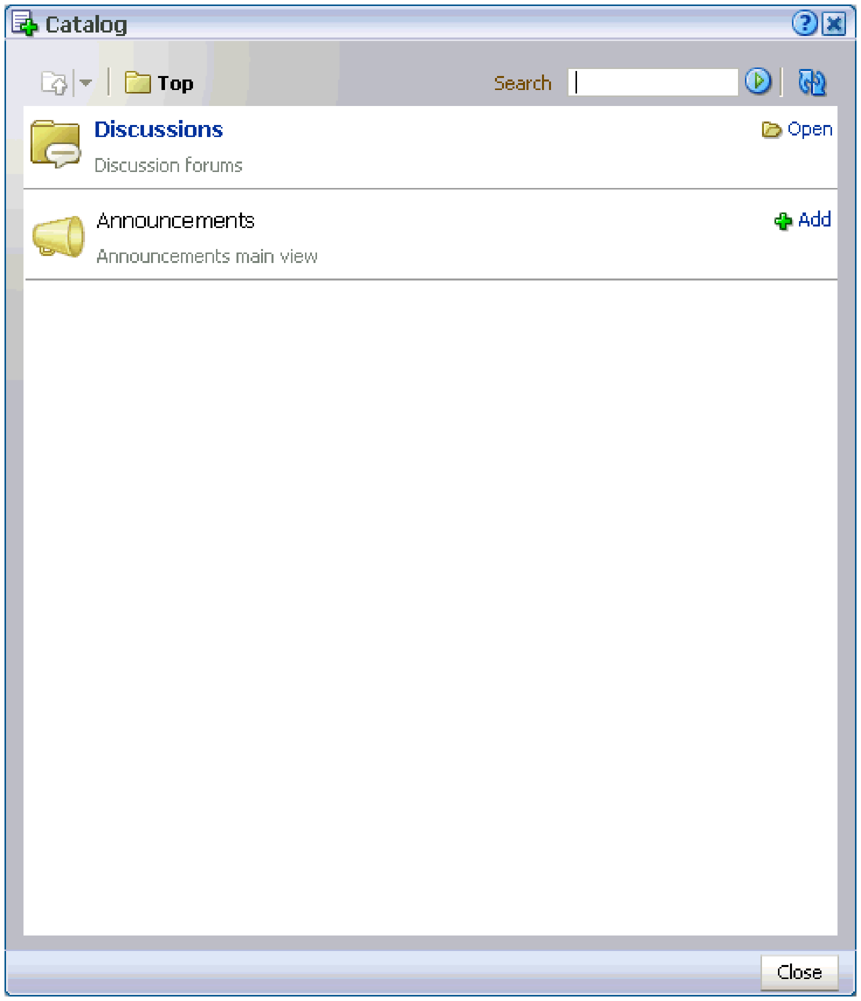
To run and test the page:

1. Run any editable JSPX page in your application.
2. Switch to page Edit mode.
3. Click an **Add Content** button on the page.

   The Catalog dialog displays your custom catalog.

   Figure 6–4 shows the custom catalog, `users-catalog`, that was described in Example 6–2.

Figure 6–4 Custom Resource Catalog Displayed in the Resource Catalog



The sample application, `CustomResourceCatalog.jws`, on the Oracle WebCenter Suite 11g Demonstrations and Samples page contains a similar custom catalog definition. You can find this page on the Oracle Technology Network (OTN) at:

`http://webcenter.oracle.com`

6.4 Filtering Items in the Resource Catalog

If you want to arrange catalog content so that it is essentially the same for all users, but displays different subsets of the content to different users, then you can configure your Resource Catalog to filter out selected items based on specific criteria.

For example, if two users `A` and `B` have different privileges on a page, you can configure your application to display the entire catalog to user `A` and only a subset of items in the catalog to user `B`.

This section describes how to filter items in the Resource Catalog. It contains the following subsections:

- Section 6.4.1, "How to Filter Items in the Resource Catalog"
- Section 6.4.2, "What Happens at Runtime"

6.4.1 How to Filter Items in the Resource Catalog

To display subsets of Resource Catalog items to different users, you must implement the `oracle.adf.rc.spi.plugin.catalog.CatalogDefinitionFilter` interface. Additionally, you must associate this filter with the Resource Catalog.

To implement the CatalogDefinitionFilter:

1. From the **File** menu, choose **New**.
2. In the New Gallery dialog, expand **General**, select **Java**, and then **Java Class**, and click **OK**.
3. In the Create Java Class dialog, specify a name for the class, for example, `CatalogFilter`.
4. Under the Optional Attributes section, add the `oracle.adf.rc.spi.plugin.catalog.CatalogDefinitionFilter` interface.
5. Click **OK**.
6. Specify the logic for filtering items.

Note: When implementing the filter logic it is common to want information about the current user and current page. This information can be obtained from the `FacesContext`.

Example 6–3 shows a sample `CatalogDefinitionFilter` implementation. It is configured to hide the `Discussions` folder (`forumsFolder`) for user `ngreenbe` from the custom catalog, `users-catalog.xml`, described in Section 6.3, "Creating a Custom Resource Catalog." The entire catalog, with the `ADF Faces Components` and `Discussions` folders, is shown to all other users.

Example 6–3 Sample Showing CatalogDefinitionFilter Implementation

```
package webcenter;

import oracle.adf.rc.catalog.CatalogElement;
import oracle.adf.rc.spi.plugin.catalog.CatalogDefinitionFilter;

import java.util.Hashtable;

import javax.faces.context.ExternalContext;
import javax.faces.context.FacesContext;

public class CatalogFilter implements CatalogDefinitionFilter {

  public boolean includeInCatalog(CatalogElement element, Hashtable env) {

    try {

      FacesContext fctx = FacesContext.getCurrentInstance();
      ExternalContext ectx = fctx.getExternalContext();
      String user = ectx.getRemoteUser();
      String resourceId = element.getId();
      if ((user != null && user.equals("ngreenbe")) &&
        resourceId != null && (resourceId.equals("forumsFolder"))) {

          return false;
        }

    }
    catch (Exception e) {
      System.out.println(e);
    }
    return true;
    }

}
```

7. Edit the application's catalog definition file to add the `definitionFilter` entry inside the `<catalogDefinition>` element:

   ```
   <catalogDefinition xmlns="http://xmlns.oracle.com/adf/rcs/catalog"
                      id="catalogDefinition"
                      name="Default Resource Catalog"
                      description="Default resource catalog definition containing
   sample entries"
                      definitionFilter="java.class.name">
   ```

   The value for *`java_class_name`* is the name of the Java class you created, for example, `webcenter.CatalogFilter`.

8. Save the catalog definition file.

6.4.2 What Happens at Runtime

When your catalog definition is opened, each entry in the catalog is passed through the `CatalogDefinitionFilter` to determine which entries should be displayed and which should be excluded. Depending on your implementation in the Java class, different subsets of items are displayed to different users.

Figure 6–6 and Figure 6–5 show the catalogs that are rendered for `ngreenbe` and any other user, for example, `sking`, on implementing the `CatalogDefinitionFilter` interface using the logic in Example 6–3.

Figure 6–5 Resource Catalog Displayed to ngreenbe

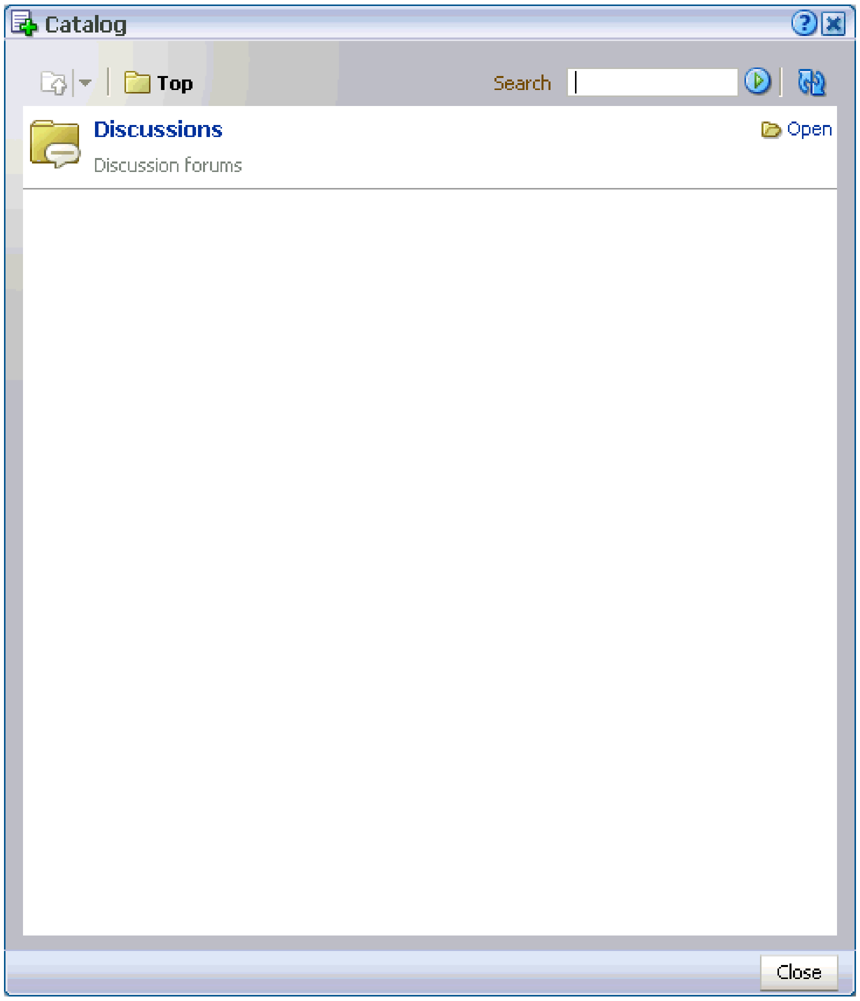
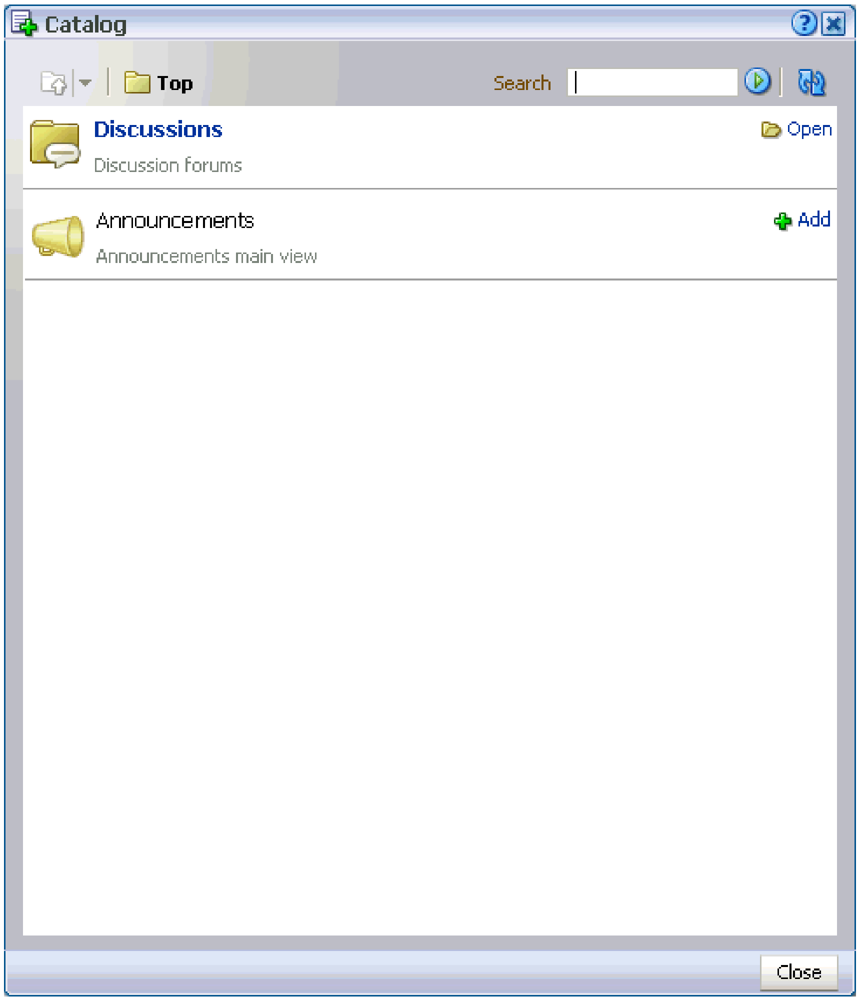


Figure 6–6 Resource Catalog Displayed to sking



The sample application, `CustomResourceCatalog.jws`, contains a similar implementation of `CatalogDefinitionFilter`. You can access it from the Oracle WebCenter Suite 11g Demonstrations and Samples page on the Oracle Technology Network (OTN) at:

`http://webcenter.oracle.com`

6.5 Configuring Multiple Resource Catalogs

To expose different items to different users based on specific criteria, such as the page being edited, the user, or the user role, you can associate multiple Resource Catalogs with your application. This section describes how. It contains the following subsections:

- Section 6.5.1, "How to Configure Multiple Resource Catalogs"
- Section 6.5.2, "What Happens at Runtime"

6.5.1 How to Configure Multiple Resource Catalogs

The main difference between using multiple catalogs and a single catalog is that, in addition to creating custom catalogs, you must implement the `ResourceCatalogSelector` API to select the appropriate catalog for the user and the page being edited. You must include the `catalog-selector` entry in your application's `adf-config.xml` file to specify the name of your `ResourceCatalogSelector` class. If you do not specify a `catalog-selector`, the default catalog is used for all editable pages and all users see the same content. If you specify a `catalog-selector`, the default catalog is used only when the specified selector returns null.

To implement multiple resource catalogs:

1. Create your custom catalogs by performing the steps outlined in Section 6.3, "Creating a Custom Resource Catalog."
2. Implement the `oracle.adf.rc.model.config.ResourceCatalogSelector` interface in your own resource catalog selector class, for example, `CatalogSelector.java`.

   Example 6–4 shows a catalog selector implementation where a custom Resource Catalog, `admin-catalog.xml`, is shown to `ahunold` who has administrator privileges, and `users-catalog.xml` is shown to users `sking` and `ngreenbe`. If there is a problem with rendering either of these catalogs, then the default catalog is shown to users.

Example 6–4 Sample ResourceCatalogSelector Showing Entries for Multiple Resource Catalogs

```
package webcenter;


import javax.faces.context.FacesContext;

import java.util.Map;

import javax.faces.component.UIViewRoot;
import javax.faces.context.FacesContext;
import javax.faces.context.ExternalContext;

import oracle.adf.rc.model.config.ResourceCatalogSelector;
import oracle.adf.rc.model.dc.RCVContext;

import oracle.adf.rc.model.config.ResourceCatalogSelector;

/**
 * CatalogSelector is a sample implementation of the ResourceCatalogSelector
 * interface.
 *
 * This implementation is based on the authenticated user
 */
public class CatalogSelector implements ResourceCatalogSelector {
  public CatalogSelector() {
  }

  /**
   * Returns the Id of the catalog that must be used for the current
   * user.
   *
   * @param context a Map cont
   * @return id of the catalog to be used for the current user or
   *         null if the default catalog must be used.
   */
  public String getCatalogName(Map context) {
        // if no catalog is selected, return null & the composer will use the
        // default catalog defined in the rcv-config
        String retval = null;

        // get information about the current user.
        FacesContext fctx = FacesContext.getCurrentInstance();
        ExternalContext ectx = fctx.getExternalContext();
        String user = ectx.getRemoteUser();

        if (user == null || user.length() == 0) {
          // user name is not available so use the default catalog
          retval = null;

        }
        else if (user.equals("ngreenbe") || user.equals("sking")  ) {
          // return users-catalog for users ngreenbe and sking
          retval = "users-catalog";


        }
        else if (user.equals("ahunold")) {
          // return admin-catalog for user ahunold
          retval = "admin-catalog";

        }

        return retval;

        }
    }
```

3. In your `adf-config.xml` file, enter the name of the class you created for selecting a resource catalog with the `<catalog-selector>` tag as show in the following example:

```
<rcv-config xmlns="http://xmlns.oracle.com/adf/rcs/viewer/adf-config">
  <catalog-selector class-name="webcenter.CatalogSelector"/>
  <default-catalog catalog-name="default-catalog"/>
</rcv-config>
```

   The value for `catalog-name` corresponds to the default catalog definition file stored in the Resource Catalog root within MDS.

   Note: For information about the Oracle Composer-specific configurations you can make in `adf-config.xml`, see Section B.2.2, "adf-config.xml."

6.5.2 What Happens at Runtime

Different catalogs are displayed to users based on the criteria you specified.

Based on the `ResourceCatalogSelector` implementation in Example 6–4, Figure 6–7 and Figure 6–8 show the two catalogs that are shown to user `sking` or `ngreenbe` and to administrator `ahunold` respectively.

Figure 6–7 Catalog Displayed to sking and ngreenbe

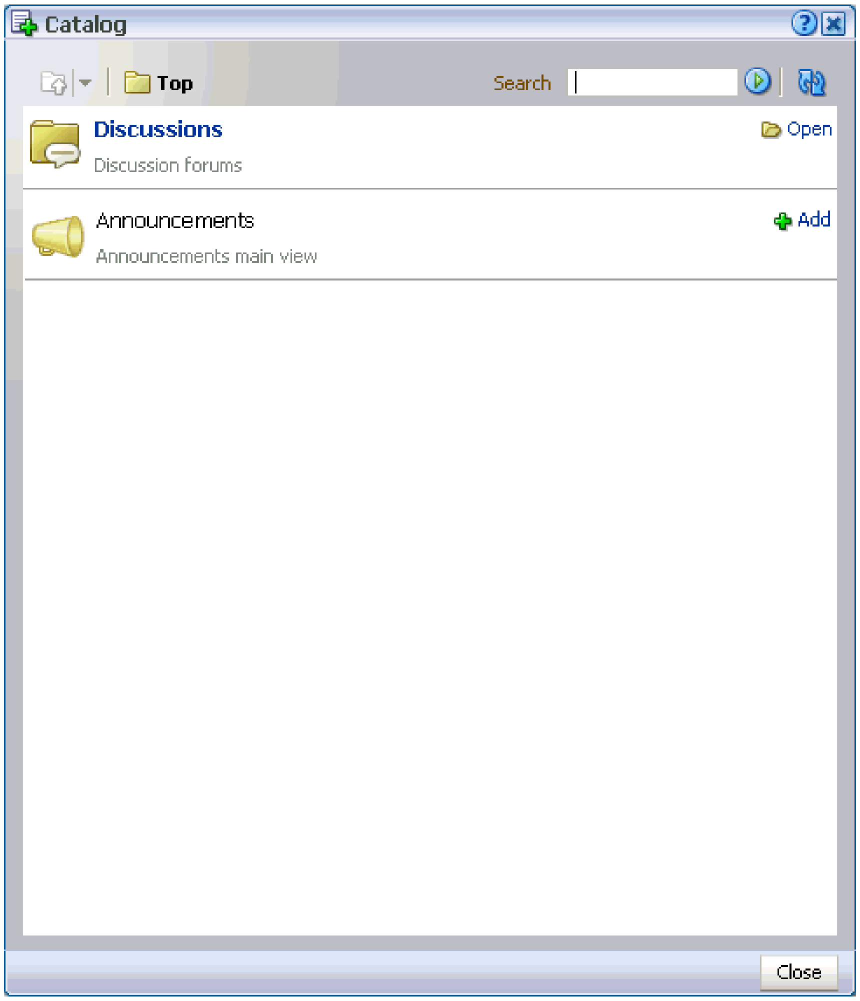
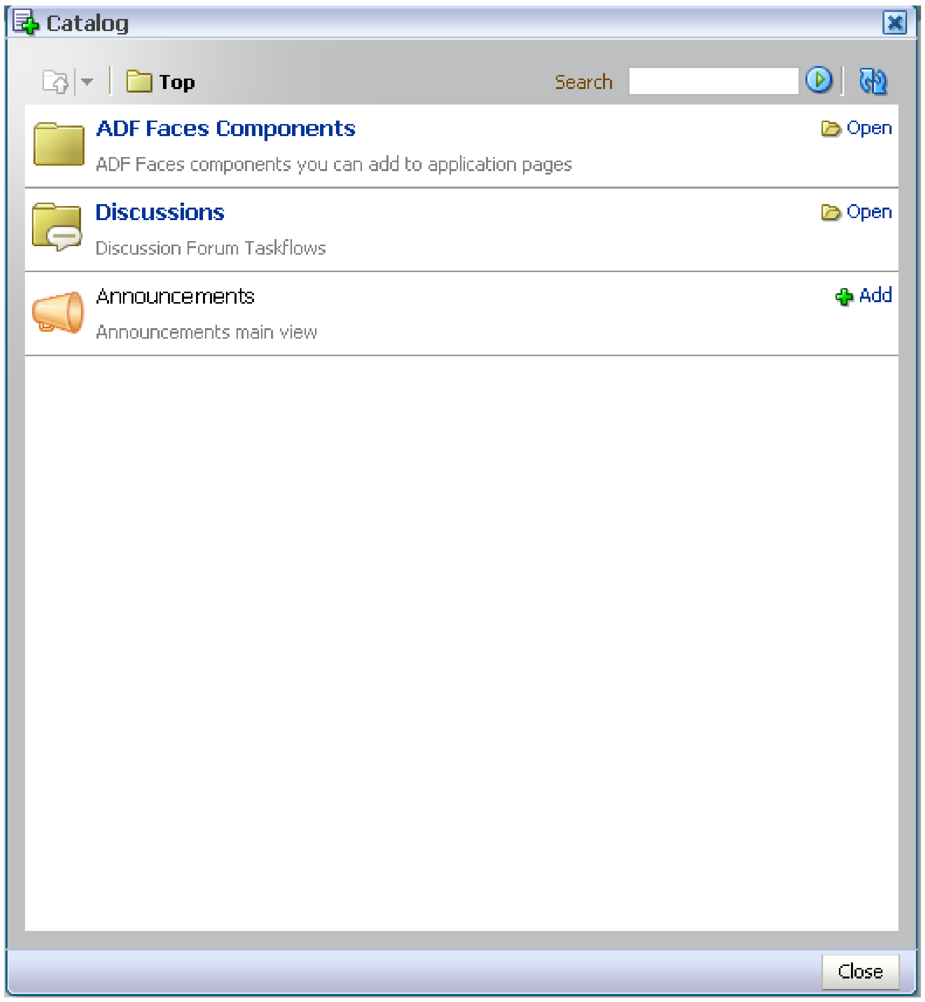


Figure 6–8 Resource Catalog Displayed to the ahunold



You can look at these catalog definitions in the `StoreFrontModule` in FOD. For information about FOD, see Chapter 2, "Introduction to the WebCenter Sample Application."

7

Enabling Runtime Creation and Management of Pages

This chapter describes how to use the Page service to create and manage pages at runtime. It includes the following sections:

7.1 Introduction to Page Creation and Management

The Page service enables you to create new pages and task flows in your application at runtime. You can base your pages on default or custom styles and templates. You can create them with either the Page - Create New task flow or the Page service APIs.

After you create pages or task flows, users can view and manage them with either the Page service data control or the Page service APIs. The APIs provide a means of defining a work area within the application, called a scope. Scopes are useful for categorizing custom pages that are of interest to a specific team or community (similar to a Group Space in WebCenter Spaces).

Table 7–1 describes the developer tools included with the Page service.

Table 7–1 Page Service Developer Tools

Tool	Description
Page - Create New task flow	A task flow for creating pages or task flows at runtime.

Table 7–1 (Cont.) Page Service Developer Tools

Tool	Description
PageServiceDC data control	A data control for viewing or deleting information about listed pages and task flows at runtime. **PageServiceDC** is available in design time after you add the Page service libraries to your application. By default, the Page service libraries are added to the application when you use the WebCenter template or when you add the Page - Create New task flow.
Page service APIs	APIs for creating and managing pages and task flows at runtime.

The Page service is integrated with many WebCenter Web 2.0 services, such as the Links, Search, and Tags services. You can track the most recent changes in pages with the Recent Activities service. Because it is preconfigured to work with the Recent Activities service, the Page service automatically produces the information that the Recent Activities service uses to display the most recent additions, changes, or deletions in pages.

For more information about the services at runtime, see *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

7.2 Creating Pages and Task Flows

You can create pages and task flows at runtime with either the Page - Create New task flow or the Page service APIs.

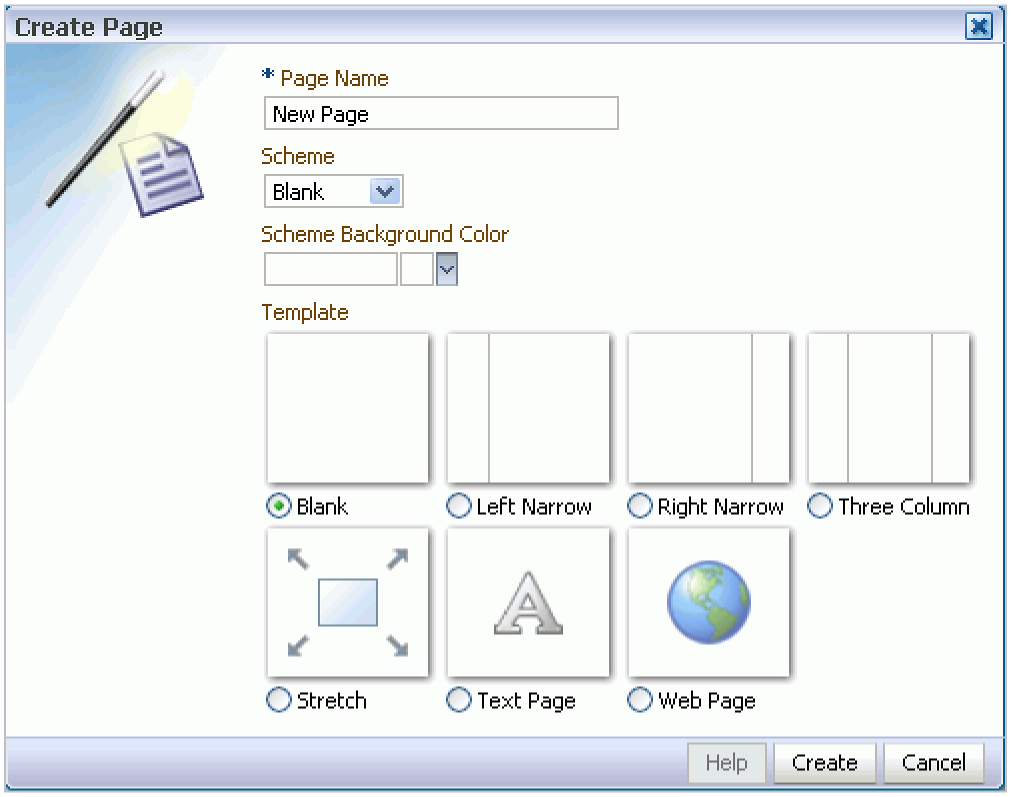
Note: The Page - Create New task flow provides a means of creating pages and task flows and editing task flow parameters. To execute operations, such as copy, rename, create scope, and so on, you must use the Page service APIs along with the Page - Create New task flow. The Page service APIs provide the flexibility of customization within the Page service.

The Page - Create New task flow enables you to invoke the Create Page dialog at runtime. The Create Page dialog in turn enables users to create pages based on predefined styles, schemes, and templates.

Page styles, schemes, and templates provide both a default page structure that describes the areas where you can place content (that is, the page layout) and a background color and image that contribute to a page look and feel. You can select the page style, scheme, scheme background color, and template when you create a page. You can additionally enhance the usefulness and presentability of a page using page layout components. These include an in-place HTML text editor, images, layout boxes, hyperlinks, and so on.

Figure 7–1 shows the Create Page dialog at runtime.

Figure 7–1   Create Page Dialog with Default Styles



* Page Name: New Page

Scheme: Blank

Template: [✓ Blank] Left Narrow Right Narrow Three Column

Some page styles include properties that suggest a particular use for the page. For example, the Web Page style includes a configurable property for specifying a URL. Among its many uses, the Web Page style provides a means of embedding wiki or blog pages and exposing external Web content within your application.

In many cases, you can switch the page scheme, scheme background color, and layout when you revise a page. You can also start with a blank page and create layout, look, and feel from scratch.

At runtime, you can view a list of pages created through the Page - Create New task flow and create, copy, update or delete those pages.

For more information, see the section "Working with Page Layouts, Styles, and Schemes" in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

7.2.1 How to Create Pages

This section describes how to add the Page - Create New task flow to your application to enable the creation of pages at runtime. It contains the following subsections:

- Section 7.2.1.1, "How to Add the Page - Create New Task Flow"
- Section 7.2.1.2, "Setting Security for the Page Service"
- Section 7.2.1.3, "How to Create Pages at Runtime"
- Section 7.2.1.4, "Structure of Pages Created at Runtime"
- Section 7.2.1.5, "How to Access Pages Created at Runtime"

7.2.1.1 How to Add the Page - Create New Task Flow

To add the Page - Create New task flow to your WebCenter application:

1. Follow the steps described in Section 3.3, "Creating WebCenter Application-Enabled Pages."
2. Open the customizable page on which you want to add the service.
3. In the Resource Palette, expand **My Catalogs**, **WebCenter Services Catalog**, and **Task Flows**.
4. Drag and drop the **Page - Create New** task flow onto your page.
5. When prompted, select **Region** as the way to create the task flow.
6. Click **OK**.
7. Save your page.
8. Run your application to see that Create Page is now on the page.
9. Click **Create Page**.

   The Create Page dialog opens (see Figure 7–1).
10. Enter a page name, then select the scheme, the scheme background color, and the template.
11. Click **Create**.

    A page is created based on the selected information. For more details on where the page is created, see Section 7.2.1.4, "Structure of Pages Created at Runtime."

Note: After you add the task flow, you can edit task flow parameters. For information about Page - Create New task flow parameters, see Section 7.3, "Defining Values for the Page - Create New Task Flow Parameters."

For detailed information about pages, such as how to delete pages created at runtime, see Section 7.5, "Managing Pages."

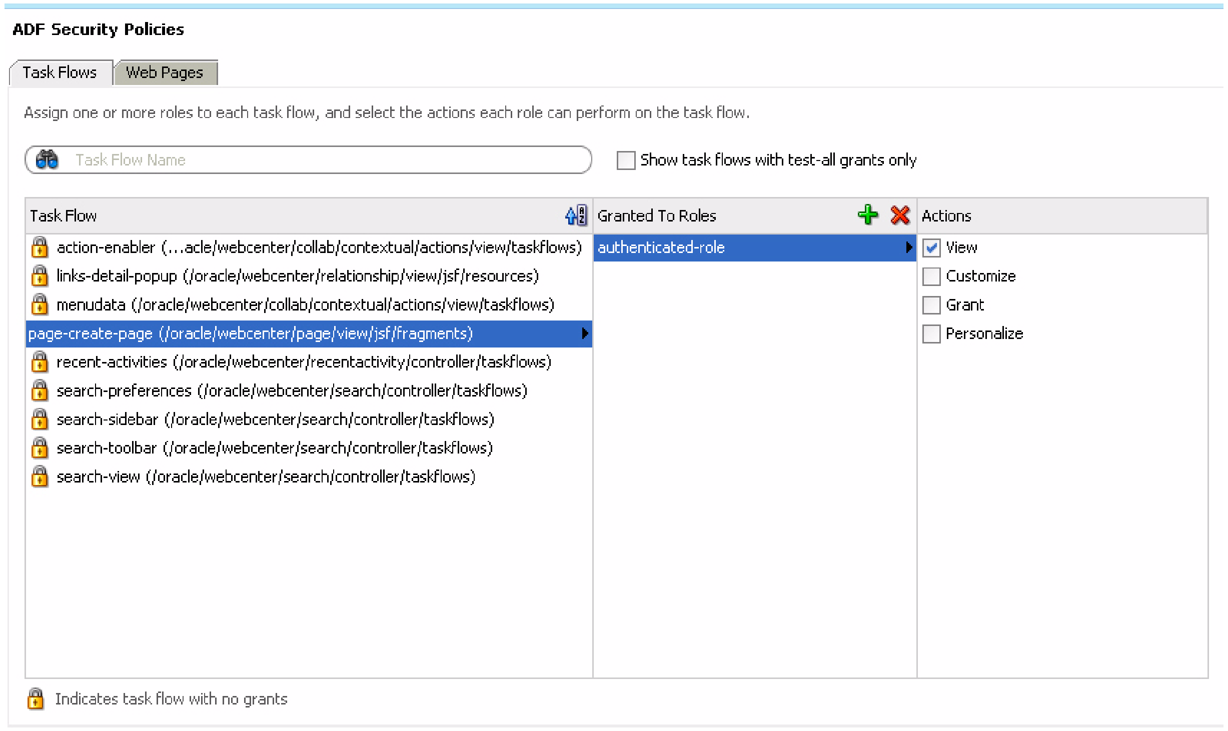
7.2.1.2 Setting Security for the Page Service

The Page service does not require its own, specific configuration of ADF security. It follows the security set up for the application. In a non-secured application, all pages created in the Create Page dialog appear in the data control, viewable by everyone. In a secured application, users see only the pages they create or the pages to which they are granted privileges.

To secure your application and the Page - Create New task flow:

1. Follow the steps in Chapter 3.5, "Implementing Security in Your Application."
2. After you add the **Page - Create New** task flow to an application page, double-click the `jazn-data.xml` file in Application Resources to open the ADF Security Policies Editor.
3. If necessary, click the **Task Flow** tab to bring it forward.
4. In the **Task Flow** column, select **page-create-new** task flow (Figure 7–2).

Figure 7–2 Adding Privileges to the Page - Create New Task Flow


ADF Security Policies

Task Flows | Web Pages

Assign one or more roles to each task flow, and select the actions each role can perform on the task flow.

Task Flow Name

☐ Show task flows with test-all grants only

Task Flow	Granted To Roles	Actions
🔒 action-enabler (...acle/webcenter/collab/contextual/actions/view/taskflows)	authenticated-role	✓ View
🔒 links-detail-popup (/oracle/webcenter/relationship/view/jsf/resources)		☐ Customize
🔒 menudata (/oracle/webcenter/collab/contextual/actions/view/taskflows)		☐ Grant
page-create-page (/oracle/webcenter/page/view/jsf/fragments)		☐ Personalize
🔒 recent-activities (/oracle/webcenter/recentactivity/controller/taskflows)		
🔒 search-preferences (/oracle/webcenter/search/controller/taskflows)		
🔒 search-sidebar (/oracle/webcenter/search/controller/taskflows)		
🔒 search-toolbar (/oracle/webcenter/search/controller/taskflows)		
🔒 search-view (/oracle/webcenter/search/controller/taskflows)		

🔒 Indicates task flow with no grants

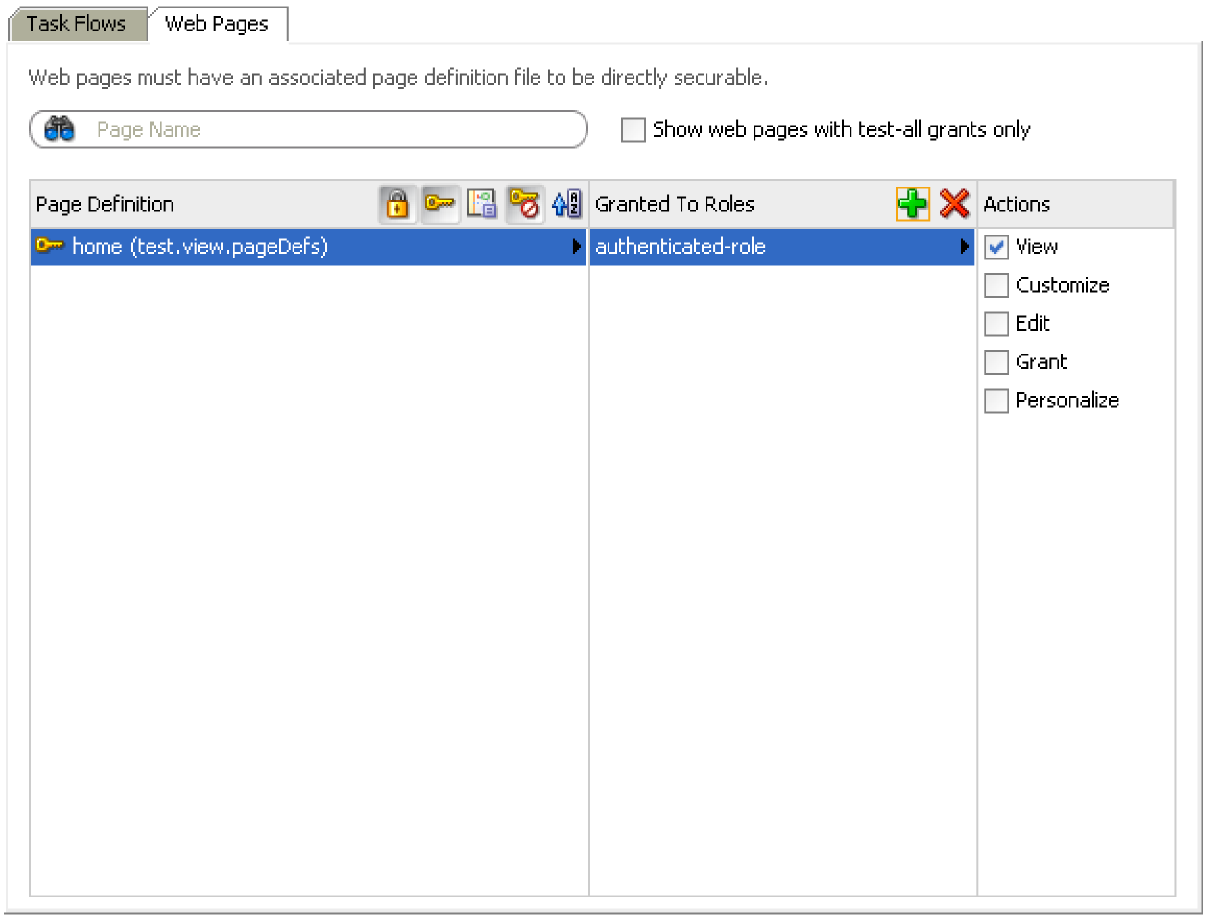
5. Under **Granted to Roles**, click the **Add** icon to add a user role on which to grant task flow access privileges.

6. After you have added user roles, select a role in the ADF Security Policies dialog and then, under Actions, select the privileges to apply to the selected role.

7. Click the **Web Pages** tab to apply security to the page on which you added the Page - Create New task flow (Figure 7–3).

   Follow the same procedure outlined in steps 3 through 6: select a page, add roles, apply privileges.

Figure 7–3 Adding Privileges to Page with the Page - Create New Task Flow



Web pages must have an associated page definition file to be directly securable.

Show web pages with test-all grants only

Page Definition	Granted To Roles	Actions
home (test.view.pageDefs)	authenticated-role	[✓ View]
		Customize
		Edit
		Grant
		Personalize

Note: The **PageServiceDC** data control does not require any security setting for a secure or non-secure application.

7.2.1.3 How to Create Pages at Runtime

For detailed information, see the chapter "Introducing the Page Service and Oracle Composer" in *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

7.2.1.4 Structure of Pages Created at Runtime

The pages you create at runtime using the Page - Create New task flow or the Page service APIs are stored in the `mds` folder, located under your application root. Table 7–2 provides examples of the default formats used for page and task flow file names.

Table 7–2 Default File Name Formats

File Type	Format
Page	`Page<N>.jspx` (value of N starts from 1) For example: `Page1.jspx`, `Page2.jspx`, `Page<N>.jspx`
Page Definition	`Page1PageDef.xml`, `Page2PageDef.xml`, `Page<N>PageDef.xml`

Table 7–2 (Cont.) Default File Name Formats

File Type	Format
Task Flow View pages	`Page1.jsff`, `Page2.jsff`, `PageN.jsff`
Task Flow View Page Definitions	`Page1PageDef.xml`, `Page2PageDef.xml`, `Page<N>PageDef.xml`
Task Flow Definitions	`Page1.xml`, `Page2.xml`, `Page<N>.xml`

When you use the Page service APIs to create pages, you can provide a custom value for the file name format. For example: myPage<N>.jspx (value of N starts from 1)

The following examples show the structure of a page (`Page1.jspx`) and its associated page definition (`Page1PageDef.xml`) created at runtime by the user `user1`.

Example 7–1 Page Created Under a Default Scope

```
mds\oracle\webcenter\page\scopedMD\s8bba98ff_4cbb_40b8_beee_
296c916a23ed\user\user1\Page1.jspx
```

Example 7–2 Page Definition Created Under a Default Scope

```
mds\pageDefs\oracle\webcenter\page\scopedMD\s8bba98ff_4cbb_40b8_beee_
296c916a23ed\user\user1\Page1PageDef.xml
```

Example 7–3 Page Created Under a Custom Scope

```
mds\oracle\webcenter\page\scopedMD\<scopeGUID>\Page1.jspx
```

Example 7–4 Page Definition Created Under a Custom Scope

```
mds\pageDefs\oracle\webcenter\page\scopedMD\<scopeGUID>\Page1PageDef.xml
```

In Example 7–4, the `scopeGUID` is a unique ID assigned to a specific scope by the Page service.

Example 7–5 Task Flow View Page Created Under a Default Scope

```
mds\oracle\webcenter\page\scopedMD\s8bba98ff_4cbb_40b8_beee_
296c916a23ed\user\user1\Page1.jspx
```

Example 7–6 Task Flow Definition Created Under a Default Scope

```
mds\oracle\webcenter\page\scopedMD\s8bba98ff_4cbb_40b8_beee_
296c916a23ed\user\user1\Page1.xml
```

Example 7–7 Task Flow View Page Definition Created Under a Default Scope

```
mds\pageDefs\oracle\webcenter\page\scopedMD\s8bba98ff_4cbb_40b8_beee_
296c916a23ed\user\user1\Page1PageDef.xml
```

Example 7–8 Task Flow View Page Created Under a Custom Scope

```
mds\oracle\webcenter\page\scopedMD\<scopeGUID>\Page1.jspx
```

Example 7–9 Task Flow Definition Created Under a Custom Scope

```
mds\oracle\webcenter\page\scopedMD\<scopeGUID>\Page1.xml
```

Example 7–10 Task Flow Page Definition Created Under a Custom Scope

```
mds\pageDefs\oracle\webcenter\page\scopedMD\<scopeGUID>\Page1PageDef.xml
```

In Example 7–10, the `scopeGUID` is a unique ID assigned to a specific scope by the Page service.

7.2.1.5 How to Access Pages Created at Runtime

To access pages with a default scope, enter the following URL:

```
http://<server>:<port>/<app name>/faces/oracle/webcenter/page/scopedMD/s8bba98ff_
4cbb_40b8_beee_296c916a23ed/user/<user guid>/Page1.jspx
```

To access pages with a custom scope, enter the following URL:

```
http://<server>:<port>/<app name>/faces/oracle/webcenter/page/scopedMD/<scope
guid>/Page1.jspx
```

7.3 Defining Values for the Page - Create New Task Flow Parameters

This section describes how to access and define values for Page - Create New task flow parameters. It contains the following subsections:

- Section 7.3.1, "How to Access Page - Create New Task Flow Parameters"
- Section 7.3.2, "Setting Scope in a Page - Create New Task Flow"
- Section 7.3.3, "Setting an Outcome Parameter"
- Section 7.3.4, "Specifying Styles"
- Section 7.3.5, "Specify an ADF Template"
- Section 7.3.6, "Showing a Command Link"
- Section 7.3.7, "Displaying an Image"
- Section 7.3.8, "Customizing the Label"

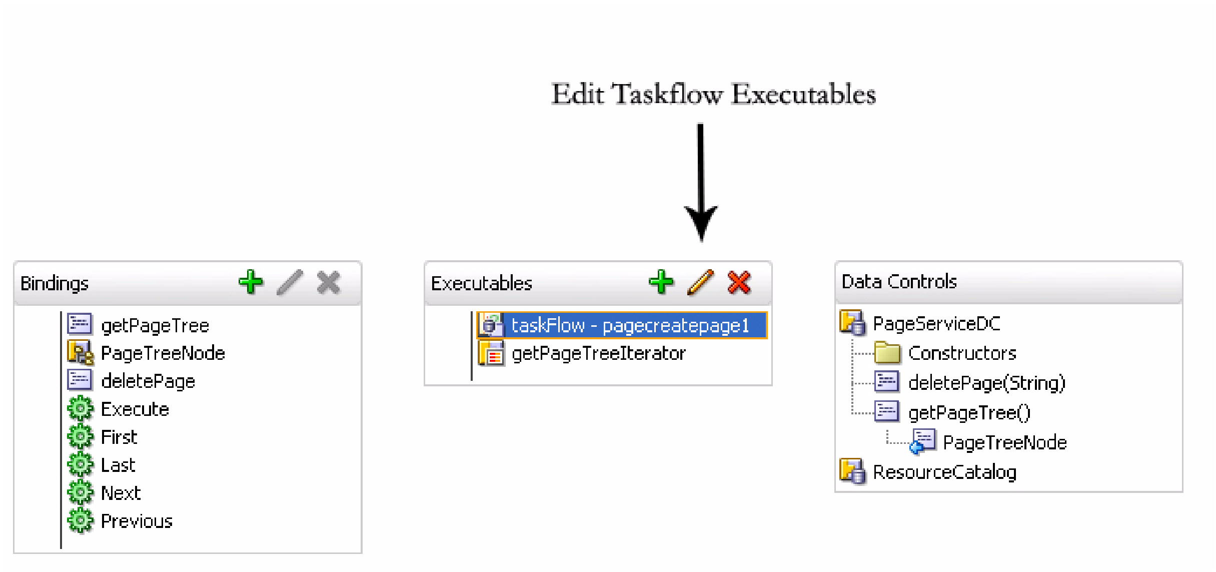
7.3.1 How to Access Page - Create New Task Flow Parameters

To access Page - Create New task flow parameters:

1. Go to the Application Navigator and right-click the `.jspx` page with the Page - Create New task flow.
2. Select **Go to Page Definition**, and scroll to select the **Create Page** task flow.
3. Click the **Edit** (pencil) icon to edit any of the parameters.

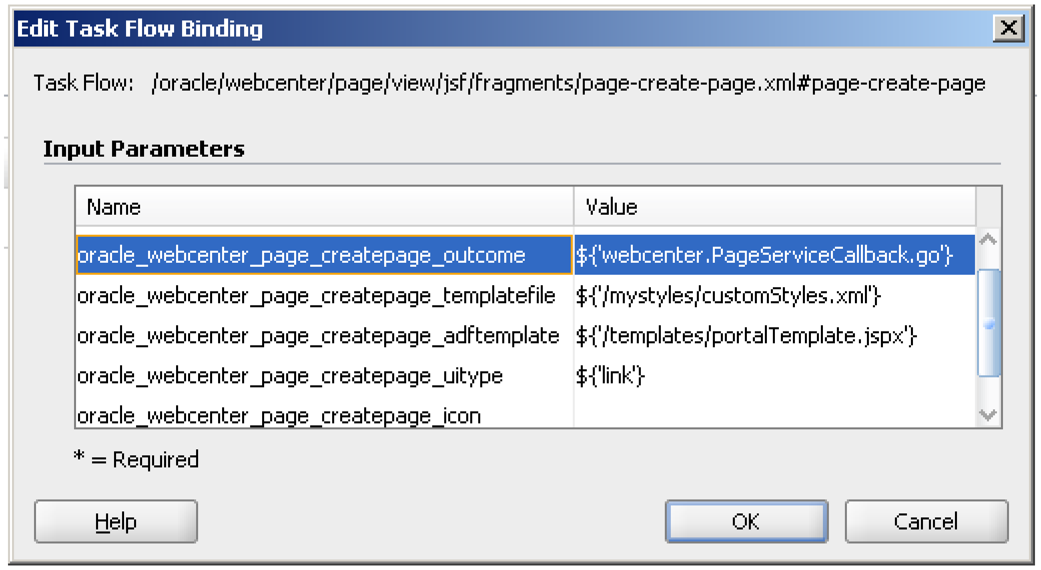
   The Page Data Binding Definition in the design view is shown in Figure 7–4.

Figure 7–4 Page Data Binding Definition



The **Edit Task Flow Binding** dialog opens, as shown in Figure 7–5.

Figure 7–5 Edit Task Flow Binding for the Page - Create New Task Flow



Task Flow: /oracle/webcenter/page/view/jsf/fragments/page-create-page.xml#page-create-page

Input Parameters

Name	Value
oracle_webcenter_page_createpage_outcome	${'webcenter.PageServiceCallback.go'}
oracle_webcenter_page_createpage_templatefile	${'/mystyles/customStyles.xml'}
oracle_webcenter_page_createpage_adftemplate	${'/templates/portalTemplate.jspx'}
oracle_webcenter_page_createpage_uitype	${'link'}
oracle_webcenter_page_createpage_icon	

* = Required

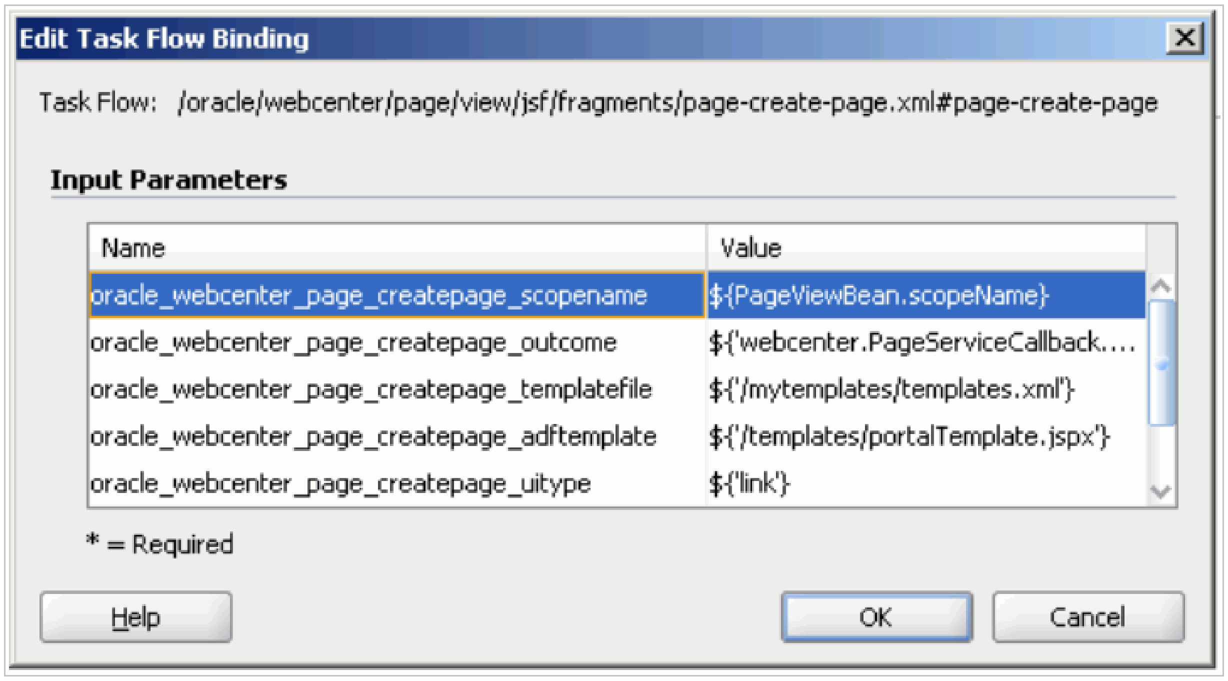
The following optional Page - Create New parameters are listed.

- `oracle_webcenter_page_createpage_scopename`
- `oracle_webcenter_page_createpage_outcome`
- `oracle_webcenter_page_createpage_templatefile`
- `oracle_webcenter_page_createpage_adftemplate`
- `oracle_webcenter_page_createpage_uitype`
- `oracle_webcenter_page_createpage_icon`
- `oracle_webcenter_page_createpage_label`

7.3.2 Setting Scope in a Page - Create New Task Flow

By default, the pages and task flows you create using the Page - Create New task flow are stored in the default scope. To assign a specific scope to the Page - Create New task flow, the scope name must be assigned to the `oracle_webcenter_page_createpage_scopename` parameter, as shown in Figure 7–6.

Figure 7–6 oracle_webcenter_page_createpage_scopename Parameter



Task Flow: /oracle/webcenter/page/view/jsf/fragments/page-create-page.xml#page-create-page

Name	Value
oracle_webcenter_page_createpage_scopename	${PageViewBean.scopeName}
oracle_webcenter_page_createpage_outcome	${'webcenter.PageServiceCallback....
oracle_webcenter_page_createpage_templatefile	${'/mytemplates/templates.xml'}
oracle_webcenter_page_createpage_adftemplate	${'/templates/portalTemplate.jspx'}
oracle_webcenter_page_createpage_uitype	${'link'}

* = Required

In Figure 7–6, `PageViewBean` is a custom managed bean that is invoking the `getScopeName()` method to fetch the value of `scopeName`. It uses the Page service API. By doing this, the Page - Create New task flow creates all pages and task flows under the specified scope, rather than default scope.

The following example code highlights only the creation of scope. If a scope already exists, then users can get the scope name from the `Scope` object.

Example 7–11 Creating a Scope for Application Pages

```
import oracle.webcenter.framework.service.Scope;
import oracle.webcenter.framework.service.ScopeAlreadyExistsException;
import oracle.webcenter.framework.service.ServiceContext;

public class PageViewBean {
    private String scopeName;

    public String getScopeName() {
        String scopeName = "MyScope";
        ServiceContext sContext = ServiceContext.getContext();
        try {
            Scope scope =
                sContext.createScope(scopeName); // Creates a new scope
            this.scopeName = scope.getName(); // Returns the scope name
            sContext.setScope(scope); // Setting the Scope. This step is
important.
        } catch (ScopeAlreadyExistsException scopeExistsException) {
            System.out.println("Scope Already Exists...");
        }
        return this.scopeName;
    }


    public void setScopeName(String scopeName) {
        this.scopeName = scopeName;
    }

}
```

7.3.3 Setting an Outcome Parameter

The outcome parameter defines a Java method that is called after the page is created. In this method, you can add dynamic content to the page, refresh the user interface that calls up the dialog to confirm creation of a new page, or initiate browser navigation to the newly created page. To specify a Java method to invoke after page creation (that is, it is called after users click the **Create** button in the Create Page dialog), enter the following in the Edit Task Flow Binding dialog:

```
oracle_webcenter_page_createpage_outcome = ${'view.MyTestClass.go'}
```

You can define the method either with a String parameter or with no parameter. The String parameter value is the new page's name with full path. If the method is defined both ways, then the one with a parameter is invoked.

In the following example, the method `go(String newPagePath)` is invoked. This method redirects users to the newly created page with the Java class `MyTestClass.java`. The value of `newPagepath` is passed by the Page service.

```
package view;

import javax.faces.context.ExternalContext;
import javax.faces.context.FacesContext;
import javax.faces.event.ActionEvent;

public class MyTestClass {
    public MyTestClass() {
    }

    /*
       This method is executed only when there is no other go(String newPagePath)
method with a String argument .
    */
    public void go() {
        System.out.println("go() method without parameters executed.");
    }

    /*
       This method is prioritized and executed even when there are other go()
methods without arguments.
    */
    public void go(String newPagePath) {
        System.out.println("go(String newPagePath) method with parameter
executed.");
        try {
            FacesContext context = FacesContext.getCurrentInstance();
            String viewID = context.getViewRoot().getViewId();
            System.out.println("ViewID=" + viewID);
            ExternalContext extCtx = context.getExternalContext();
            String targetURI =
                extCtx.getRequestContextPath() + "/faces" + viewID;
            System.out.println("Navigate to: " + targetURI);
            extCtx.redirect(targetURI);
            context.responseComplete();
        } catch (Exception ee) {
            System.out.println("Error: " + ee.toString());
        }

    }
}
```

7.3.4 Specifying Styles

The Page service provides out-of-the-box styles that are packaged within the Page service libraries. You can use these styles or create your own custom styles.

This section describes how to apply out-of-the-box styles and custom styles. It contains the following subsections:

- Section 7.3.4.1, "Out-of-the-Box Styles"
- Section 7.3.4.2, "Custom Styles"

Note: For more information about styles, see Section 7.6, "Introduction to Custom Styles and Templates."

7.3.4.1 Out-of-the-Box Styles

The following out-of-the-box-styles are provided (see Figure 7–1):

- Blank
- Left Narrow
- Right Narrow
- Three Column
- Stretch
- Text Page
- Web Page

To associate out-of-the-box styles with the Page - Create New task flow, use the optional parameter `oracle_webcenter_page_createpage_templatefile`. Use the parameter value to specify the location of the file that contains the list of out-of-the-box styles.

To associate out-of-the-box styles with the Page - Create New task flow:

1. Create a folder named `mystyles` under your application's `public_html` folder.
2. Within `mystyles`, create a file named `default_pageservice_styles.xml`.
3. Add entries for all the out-of-the-box styles using the `templateDef` element.

   The `templateDef` element takes the following attributes:

   - `name`: This attribute specifies the path to the `.jspx` page that defines a page style. When you use the default styles, you must specify the path to the page that defines the style you intend to use. The default page style page files are located within the Page service libraries. For example:

```
name="/oracle/webcenter/page/pstemplates/TemplateThreeColumn.jspx"
```

`TemplateThreeColumn.jspx` is the page within the Page service library that defines the Three Column Template style.

- `title`: The value you enter for `title` is used as the lable for the style in the Create Page dialog. You can modify the title of an out-of-the-box style to any name. For example: `title="Three Column Layout"` or `title="Sales Standard"`.
- `icon`: This attribute specifies an image icon for a style.

  For out-of-the-box styles, if you do not specify an icon for a style, then the Page service automatically uses the default icons assigned to each out-of-the-box style.

Example 7–12 shows the content of a sample `default_pageservice_styles.xml` file that lists the other out-of-the-box styles:

Example 7–12 Sample default_pageservice_styles.xml File

```
<?xml version="1.0" encoding="UTF-8" ?>
 <templatesDef xmlns="http://xmlns.oracle.com/webcenter/page">
  <templateDef name="/oracle/webcenter/page/pstemplates/TemplateThreeColumn.jspx"
title="Three Column Layout" />
  <templateDef name="/oracle/webcenter/page/pstemplates/TemplateText.jspx"
title="Text Page" />
  <templateDef name="/oracle/webcenter/page/pstemplates/TemplateStretch.jspx"
title="Stretch" />
  <templateDef name="/oracle/webcenter/page/pstemplates/TemplateWeb.jspx"
title="Web Page" />
  <templateDef name="/oracle/webcenter/page/pstemplates/TemplateBlank.jspx"
title="Blank" />
  <templateDef name="/oracle/webcenter/page/pstemplates/TemplateNarrowLeft.jspx"
title="Left Narrow Column Layout" />
  <templateDef name="/oracle/webcenter/page/pstemplates/TemplateNarrowRight.jspx"
title="Right Narrow Column Layout" />
 </templatesDef>
```

4. Open the Edit Task Flow Binding dialog as described in Section 7.3.1, "How to Access Page - Create New Task Flow Parameters."
5. Set the `oracle_webcenter_page_createpage_templatefile` parameter value to `${'/mystyles/default_pageservice_style.xml'}`.

   For example:

   ```
   oracle_webcenter_page_createpage_templatefile = ${'/mystyles/default_
   pageservice_style.xml'}
   ```

6. Click **OK**.

7.3.4.2 Custom Styles

In this example, you associate a custom style, labeled **News**, with the Page - Create New task flow.

To associate a custom style with the Page - Create New task flow:

1. Define your custom style in a `.jspx` file (for example, `NewsTemplate.jspx`).

Note: For information about how to create ADF templates, see Section 7.6, "Introduction to Custom Styles and Templates."

---

2. Add an entry for the custom style in an XML file (for example, `custom_pageservice_style.xml`).

   For example:

```
<?xml version="1.0" encoding="UTF-8" ?>
  <templatesDef xmlns="http://xmlns.oracle.com/webcenter/page">
    <templateDef name="NewsTemplate.jspx" title="News" icon="/images/news.png"
/>
  </templatesDef>
```

Note: For more information about the syntax to use for `templateDef` element and the style-related `.xml` file, see Section 7.3.4.1, "Out-of-the-Box Styles."

---

3. Save the `custom_pageservice_style.xml` under a folder in your application's `public_html` folder.

   For example:

```
…/public_html/mystyles/custom_pageservice_style.xml
```

4. The following files must be placed in the same folder as `custom_pageservice_style.xml`:

   - The file that defines the custom style, for example, `NewsTemplate.jspx`.
   - The page definition file of the custom style file, for example, `NewTemplatePageDef.xml`.
   - The ADF Template (for example, `defaultTemplate.jspx`) on which the custom style file is based. This step is applicable only if you are using ADF templates.

Note: If you specify the location of an ADF template using the `oracle_webcenter_page_createpage_adftemplate` as described in following section, then you *do not* need to put this file (`defaultTemplate.jspx`) under the `mystyles` folder. You can specify a different location for the file by following Section 7.3.5, "Specify an ADF Template."

---

5. Open the Edit Task Flow Binding dialog as described in Section 7.3.1, "How to Access Page - Create New Task Flow Parameters."

6. Set the `oracle_webcenter_page_createpage_templatefile` parameter to `${'/mystyles/custom_pageservice_style.xml'}`, where `mystyles` is a folder under the `public_html` folder of your application.

   For example:

```
oracle_webcenter_page_createpage_templatefile = ${'/mystyles/custom_
pageservice_style.xml'}
```

7.3.5 Specify an ADF Template

When you want to use the same template for both design time and runtime pages, you can use an ADF template. All that is required is to point the Page - Create New task flow to the template location.

To associate an ADF template with the Page - Create New task flow:

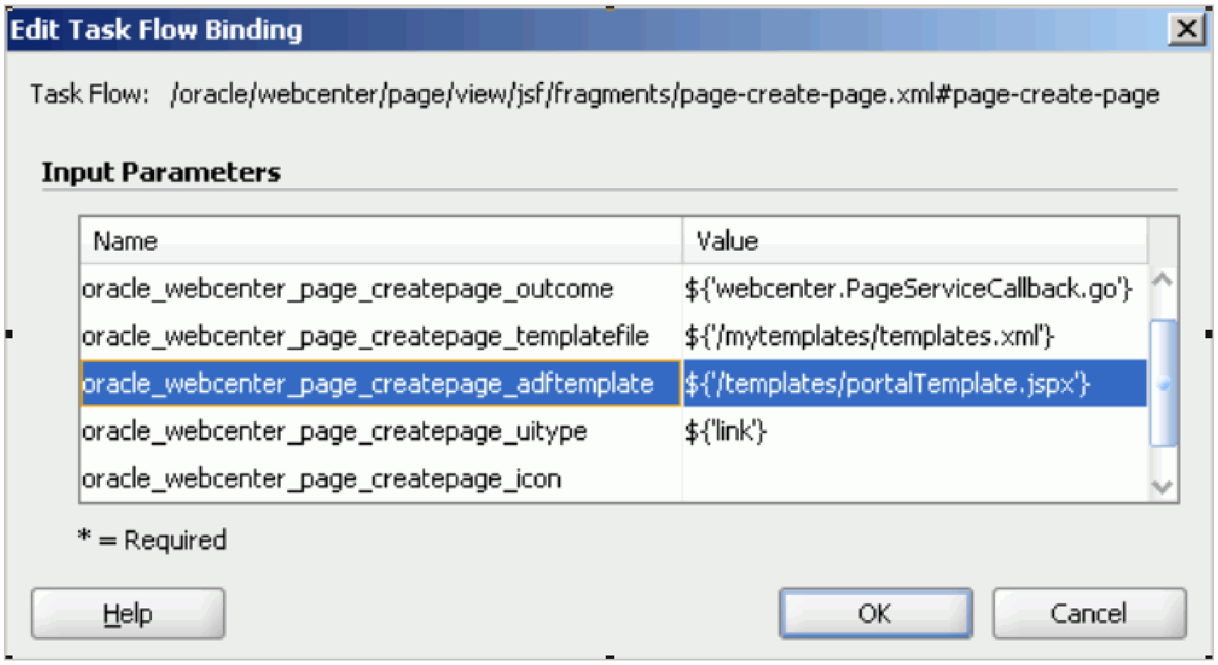
1. Open the Edit Task Flow Binding dialog as described in Section 7.3.1, "How to Access Page - Create New Task Flow Parameters."
2. In the Value column next to the `oracle_webcenter_page_creatpage_adftemplate` parameter, enter the path to the ADF template.

   For example:

   ```
   ${'templates/portalTemplate.jspx'}
   ```

   In Figure 7–7, an ADF template called `portalTemplate.jspx` is used.

Figure 7–7 oracle_webcenter_page_createpage_adftemplate Parameter



Edit Task Flow Binding

Task Flow: /oracle/webcenter/page/view/jsf/fragments/page-create-page.xml#page-create-page

Input Parameters

Name	Value
oracle_webcenter_page_createpage_outcome	${'webcenter.PageServiceCallback.go'}
oracle_webcenter_page_createpage_templatefile	${'/mytemplates/templates.xml'}
oracle_webcenter_page_createpage_adftemplate	${'/templates/portalTemplate.jspx'}
oracle_webcenter_page_createpage_uitype	${'link'}
oracle_webcenter_page_createpage_icon	

* = Required

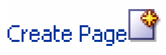
   ADF templates can be in your application's `public_html` folder. In this case, the path of `portalTemplate.jspx` is `../public_html/templates/portalTemplate.jspx`.

3. Click **OK**.

7.3.6 Showing a Command Link

By default, the Page - Create New task flow is rendered as a command *button*. You can instead specify that it is rendered as a command *link* (Figure 7–8).

Figure 7–8 Command Link



To render the Page - Create New task flow as a command link:

1. Open the Edit Task Flow Binding dialog as described in Section 7.3.1, "How to Access Page - Create New Task Flow Parameters."
2. Revise the `oracle_webcenter_page_createpage_uitype` parameter to the following value: `${'link'}`.

   For example:

   ```
   oracle_webcenter_page_createpage_uitype = ${'link'}
   ```
3. Click **OK**.

7.3.7 Displaying an Image

You can associate an icon with the Page - Create New task flow button or link. For example, Figure 7–9 shows the task flow rendered as a command *link* with a page icon.

Figure 7–9 Command Link with a Page Icon

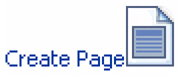
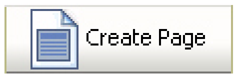


Figure 7–10 shows the task flow rendered as a command *button* with a page icon.

Figure 7–10 Command Button with a Page Icon



To specify an icon for the Page - Create New task flow button or link:

1. Open the Edit Task Flow Binding dialog as described in Section 7.3.1, "How to Access Page - Create New Task Flow Parameters."
2. Revise the `oracle_webcenter_page_createpage_icon` parameter to the following value: `{'/images/page.jpg'}`.

   For example:

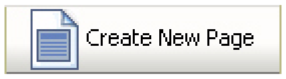
   ```
   oracle_webcenter_page_createpage_icon = ${'/images/page.jpg'}
   ```
3. Click **OK**.

7.3.8 Customizing the Label

By default, the label for the Page - Create New task flow command button or link is **Create Page**. You can provide your own value using the `oracle_webcenter_page_createpage_label` parameter value.

Figure 7–11 shows the task flow rendered as a command button with a custom label.

Figure 7–11 Custom Label



To change the command button or link label:

1. Open the Edit Task Flow Binding dialog as described in Section 7.3.1, "How to Access Page - Create New Task Flow Parameters."
2. Revise the `oracle_webcenter_page_createpage_label` parameter to your preferred string value.

   For example:

   ```
   oracle_webcenter_page_createpage_label = ${'Create New Page'}
   ```

3. Click **OK**.

7.4 How to Create Task Flow View Pages

The following types of objects are required for the creation of a task flow view page:

- Task flow view page (JSFF file)
- Page definition of view page
- Task flow definition XML file, which defines the task flow

To enable the creation of task flow view pages through the Page - Create New task flow:

1. At design time, add a **Page - Create New** task flow to your page (`home.jspx`).
2. Open the Edit Task Flow Binding dialog as described in Section 7.3.1, "How to Access Page - Create New Task Flow Parameters."
3. Change the label on the Create Page button or link by revising the value specified for the `oracle_webcenter_page_createpage_label` parameter.

   For example:

   ```
   oracle_webcenter_page_createpage_label = ${'Create Task Flow'}
   ```

4. Click **OK**.
5. Create an XML file (for example, `custom_taskflow_styles.xml`) to contain entries for all the task flow styles.

   For example:

   ```
   <?xml version='1.0' encoding="UTF-8"?>
   <templatesDef xmlns="http://xmlns.oracle.com/webcenter/page">
     <templateDef name="mytaskflow_view.jsff" title="News"
   icon="/images/news.png" type="taskflow"/>
   </templatesDef>
   ```

   The element `templateDef` in the XML file contains the following attributes:

   - `name`: to specify the name of the task flow view page
   - `title`: to specify the label to associate with the task flow view page style in the Create Page dialog.
   - `icon`: to specify the image icon for the task flow

   Note: For sample attribute syntaxes, see step 3 in Section 7.3.4.1, "Out-of-the-Box Styles."

6. Under your application's `public_html` folder, create a new folder (for example, `mystyles`) and add the `custom_taskflow_styles.xml` file to it.
7. Make sure that the following files are also under the `mystyles` folder:
   - `mytaskflow_view.jsff` (task flow view page)
   - `mytaskflow_viewPageDef.xml` (task flow view page definition)
   - `mytaskflow_view.xml` (task flow definition)
8. Open the Edit Task Flow Binding dialog again, and provide the path to the XML file you created in step 5 to the `oracle_webcenter_page_createpage_templatefile` parameter.

   For example:

   ```
   oracle_webcenter_page_createpage_templatefile=${'/mystyles/custom_taskflow_
   styles.xml' }
   ```

   In this example, `mystyles` is a folder under the `public_html` folder of your application.
9. Run the `home.jspx` page, and then click the **Create Task Flow** button.
10. Enter the title of the task flow, select the **News** style (see Section 7.3.4.2, "Custom Styles"), and then click **Create**.

    The new task flow view page is rendered.

For information on where the task flows are created, see section Section 7.2.1.4, "Structure of Pages Created at Runtime."

7.5 Managing Pages

You can manage pages using either the Page service data control or the Page service APIs. This section contains the following subsections:

- Section 7.5.1, "Using the Page Service Data Control to Manage Pages"
- Section 7.5.2, "Using the Page Service APIs to Manage Pages and Task Flows"

7.5.1 Using the Page Service Data Control to Manage Pages

You can customize Page service views with the Page service data control. The data control enables you to view information about existing pages at runtime and delete any of the listed pages. The Page service data control, `PageServiceDC`, is included in the Page service libraries and is available at design time after you add the libraries to the application.

This section describes how to add a Page service data control to a project and how to use it to view, edit, and delete pages. It contains the following subsections:

- Section 7.5.1.1, "How to Add the Page Service Data Control"
- Section 7.5.1.2, "How to View, Edit, and Delete Pages at Runtime"

7.5.1.1 How to Add the Page Service Data Control

This section provides an example of adding the Page service data control at design time.

To add a Page service data control to your project:

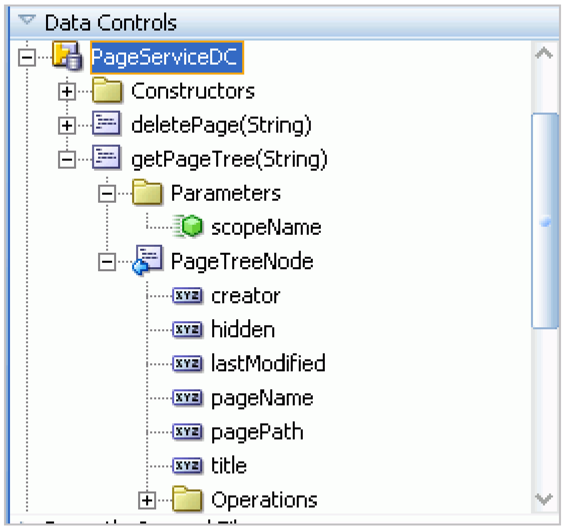
1. Ensure that you can see the PageServiceDC in the Application Navigator.

The PageServiceDC appears in the Application Navigator after you have performed either of the following actions:

- Added the **Page - Create New** task flow to your project (see Section 7.2.1.1, "How to Add the Page - Create New Task Flow").
- Added the data control from My Catalogs by expanding **WebCenter Services Catalog - Data Controls**, right-clicking **Page Data Control**, and selecting **Add to Project**.

2. In the Data Controls panel of the Application Navigator, expand **PageServiceDC**, as shown in Figure 7–12.

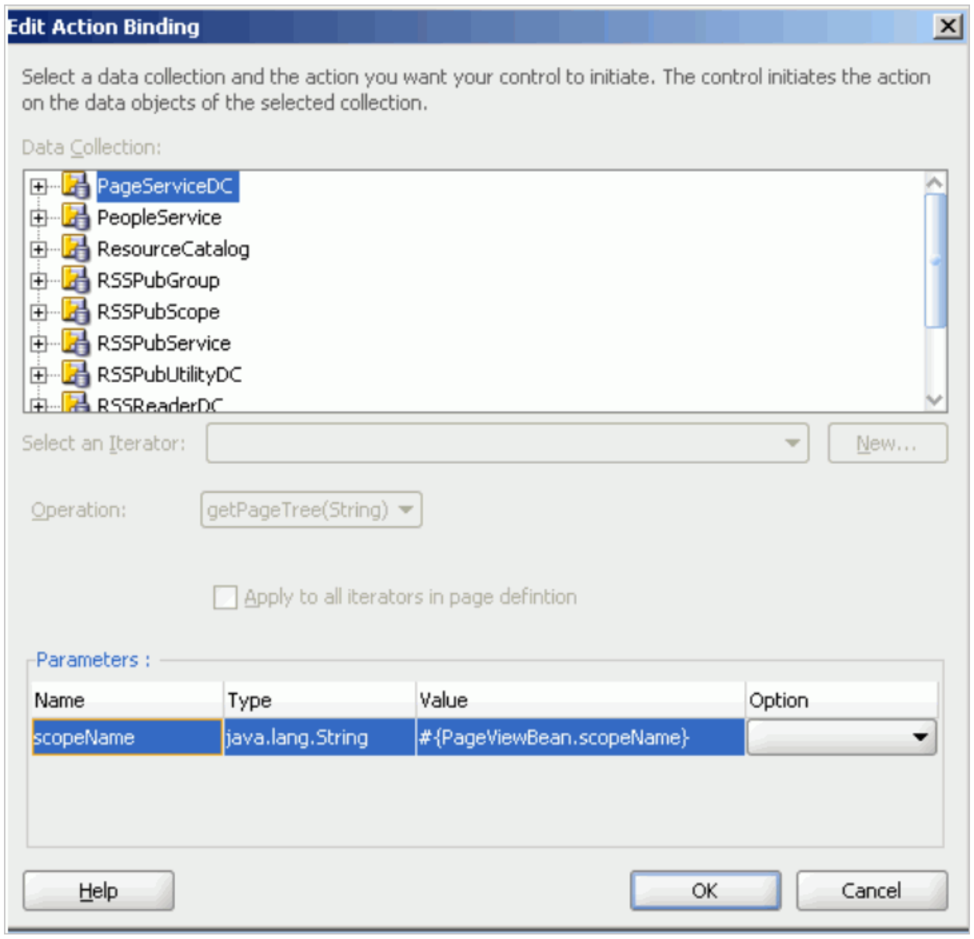
Figure 7–12 Page Service Data Control



3. Under **getPageTree()**, drag **PageTreeNode** and drop it on the page.
4. From the **Create** menu, select **Table**, and then select **ADF Read-only Table**.
5. In the dialog, enter the `scopeName` parameter to view all the pages listed under the specified scope.

   If you do not enter any value, then the data control takes the default scope. In Figure 7–13, the scope value is fetched from a custom managed bean.

Figure 7–13 scopeName Parameter


Edit Action Binding

Select a data collection and the action you want your control to initiate. The control initiates the action on the data objects of the selected collection.

Data Collection:

- PageServiceDC
- PeopleService
- ResourceCatalog
- RSSPubGroup
- RSSPubScope
- RSSPubService
- RSSPubUtilityDC
- RSSReaderDC

Select an Iterator:

Operation: getPageTree(String)

Apply to all iterators in page defintion

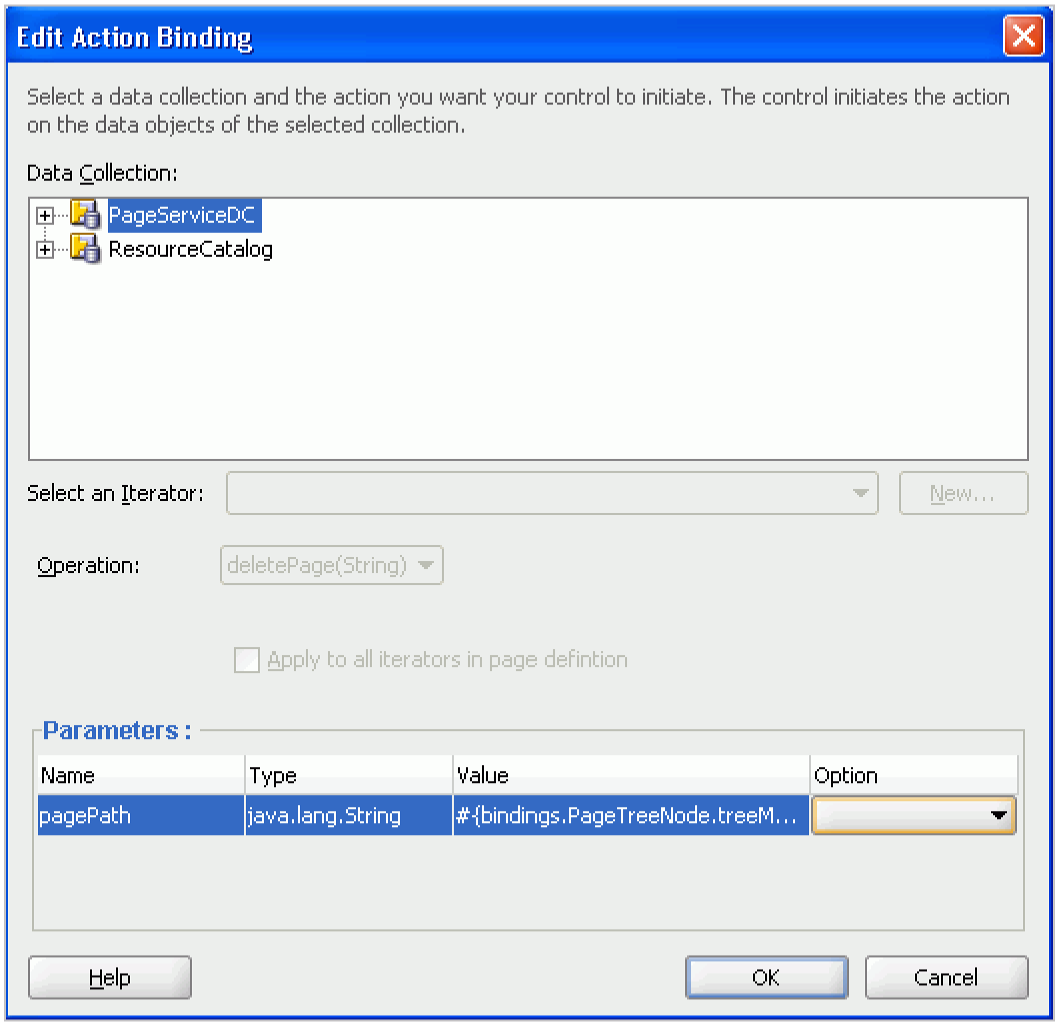
Parameters :

Name	Type	Value	Option
scopeName	java.lang.String	#{PageViewBean.scopeName}	

6. Add a new column to the table you created at step 4.
7. To add a column for deleting pages at runtime, drag the **deletePage(string)** method and drop it in the new column.
8. From the **Create** menu, select **Methods**, and then select **ADF Link**.
9. In the Parameters section of the Edit Action Binding dialog, shown in Figure 7–14, specify the following value for the `pageName` parameter:

   ```
   #{bindings.PageTreeNode.treeModel.rowData.pagePath}
   ```

Figure 7–14 Edit Action Binding Dialog


Edit Action Binding

Select a data collection and the action you want your control to initiate. The control initiates the action on the data objects of the selected collection.

Data Collection:
- PageServiceDC
- ResourceCatalog

Select an Iterator:

Operation: deletePage(String)

Apply to all iterators in page defintion

Parameters :

Name	Type	Value	Option
pagePath	java.lang.String	#{bindings.PageTreeNode.treeM...	

10. Click **OK**.
11. From the Data Controls panel, expand **PageServiceDC**, **getPageTree()**, **PageTreeNode**, and then **Operations**.
12. Drag **Execute** and drop it on the page as an **ADF Button**.
13. In the Source view, set the button's `Text` attribute to `Refresh`.

    This provides a refresh feature on the list of pages to enable users to refresh the list after they have added a page or deleted one from the list.
14. Optionally, to display pages as links that invoke page Edit mode, view the page source and replace the `outputText` element in the table's page path column with a `goLink` as follows:

    Replace

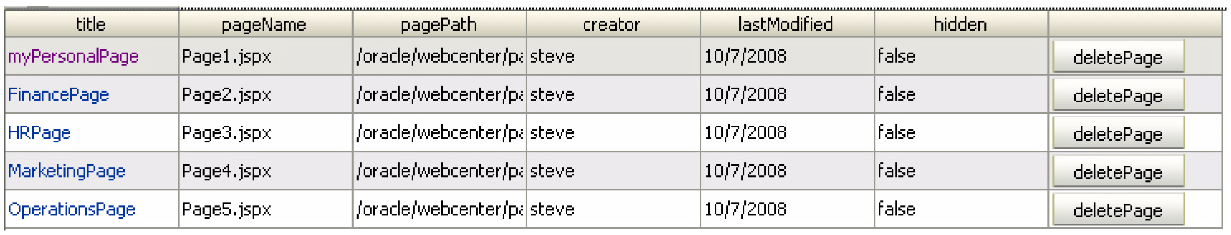
    ```
    <af:outputText value="#{row.pagePath}"/>
    ```

    with

    ```
    <af:goLink destination="/faces#{row.pagePath}" text="#{row.title}"
    targetFrame="_top"/>
    ```

15. Save your page, and run it to your browser.
16. The created data control table should look something like Figure 7–15.

Figure 7–15 Sample Data Control Table



title	pageName	pagePath	creator	lastModified	hidden	
myPersonalPage	Page1.jspx	/oracle/webcenter/pa	steve	10/7/2008	false	deletePage
FinancePage	Page2.jspx	/oracle/webcenter/pa	steve	10/7/2008	false	deletePage
HRPage	Page3.jspx	/oracle/webcenter/pa	steve	10/7/2008	false	deletePage
MarketingPage	Page4.jspx	/oracle/webcenter/pa	steve	10/7/2008	false	deletePage
OperationsPage	Page5.jspx	/oracle/webcenter/pa	steve	10/7/2008	false	deletePage

7.5.1.2 How to View, Edit, and Delete Pages at Runtime

To view a page, follow the link generated from step 14 in the previous section. If the application has been secured properly, then an edit link should render on the page at runtime.

To delete a page, click the **deletePage** button in the row of the relevant page. This button was generated from in step 7 in the previous section. Click the Refresh button for the table to remove the deleted page row from the table.

For detailed information about how to view, edit and delete pages at runtime, see the chapter "Creating, Editing, and Deleting Pages" in *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

7.5.2 Using the Page Service APIs to Manage Pages and Task Flows

The Page service provides APIs to manage pages and task flows. These include APIs to delete pages and task flows, change the hidden status, change the page scheme (name, background color, background image), and copy pages.

For more information about Page service APIs, see Section 7.8, "Introduction to the Page Service APIs."

7.6 Introduction to Custom Styles and Templates

At runtime, when uses click the Create Page option, the Create Page dialog displays a set of predefined styles for creating the page (see Figure 7–1). Users select one of these page styles to create a new page based on the associated style template. The *style* provides a special look and feel to your pages. The *template* provides the page layout. Templates are the ADF pages on which runtime pages are based. Pages can be based on default or custom styles and templates.

To provide custom selections for the Create Page dialog, you must define them, for example, in `/mytemplates/templates.xml`. Place the directory `/mytemplates` under the web content root.

If you choose *not* to use the default options available with the Create Page dialog, then you can use the following types of custom templates in your application:

- Page and page fragment templates for use with runtime pages
- ADF templates for use as the base for page and page fragment templates
- Templates to be used by the Create Page dialog itself

This section explains how to customize the Page - Create New task flow to use different templates for the Create Page dialog and for the new pages that you create at runtime. It contains the following subsections:

- Section 7.6.1, "How to Create Templates for Pages Created at Runtime"
- Section 7.6.2, "Using ADF Templates"
- Section 7.6.3, "Creating Styles for the Create Page Dialog"

Note: For more information about the `templates.xml` file, see Section 7.3.4, "Specifying Styles."

7.6.1 How to Create Templates for Pages Created at Runtime

You can create custom page templates according to your design requirements and associate page styles with these templates. If you want to restrict the style options in the Create Page dialog or provide a different set of styles, then you must perform the following high-level tasks:

- Create pages or page fragments to be used as templates for the different page styles.
- Edit the Create Page dialog parameter to specify new page styles.

This section steps you through both procedures. It contains the following subsections:

- Section 7.6.1.1, "How to Create a Page or Page Fragment Style"
- Section 7.6.1.2, "How to Edit Create Page Dialog Styles"

7.6.1.1 How to Create a Page or Page Fragment Style

To create a page or page fragment style:

1. In Oracle JDeveloper, create a new JSF template with all necessary information, such as header, welcome message, login/logout links, and so on.
2. When creating pages, create a page (JSPX) based on that template.

   When creating task flows, create a page fragment (JSFF) based on that template.

   When creating a page based on a template, the Page service actually copies the template and creates a new page based on it.
3. Put all the necessary content in the JSPX or JSFF file.

   For example, if there is a facet where you would like to display content, you can drop a Page Customizable component from Oracle Composer around it to make the page editable at runtime. Make sure you set the correct values for the `document title`, `styleClass`, and `inlineStyle` attributes.

   Note: If this page or page fragment will be used as a template for pages that can be customized at runtime, then you must ensure that the `ID` attribute on all components on this page or fragment is set to a unique value.

   Example 7–13 shows the sample code of a page fragment style, `pstemplateview.jsff`.

Example 7–13 Sample Code of a New Page Style

```
<?xml version='1.0' encoding='utf-8'?>
<!-- Copyright (c) 2006, 2008, Oracle and/or its affiliates.
All rights reserved. -->
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:pe="http://xmlns.oracle.com/adf/pageeditor"
          xmlns:cust="http://xmlns.oracle.com/adf/faces/customizable"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich">
     <jsp:directive.page contentType="text/html;charset=utf-8"/>
     <f:loadBundle basename="oracle.webcenter.page.view.resource.PGUIBundle"
                   var="res"/>
     <f:view>
       <af:document title="#{pageDocBean.title}" id="docrt">
         <af:form usesUpload="true" id="f1">
           <af:pageTemplate
viewId="/oracle/webcenter/page/pstemplates/defaultTemplate.jspx" id="T">
             <f:facet name="content">
               <pe:pageCustomizable id="pcl1">
                 <af:panelStretchLayout id="psl2"
                                        styleClass="replace_with_scheme_name"
                                        inlineStyle="replace_with_inline_style">
                   <f:facet name="center">
                     <af:panelGroupLayout id="pgl1"
                                          layout="scroll">
                       <pe:layoutCustomizable id="lc1"
                                 showLayoutChanger="#{pageServiceBean.isEditMode}"
                                 text="Change Layout"
                                 showIcon="false"
                                 type="threeColumnNarrow"
                                 shortDesc="Layout Changer">
                         <cust:panelCustomizable id="mainC"/>
                         <f:facet name="contentA">
                           <cust:panelCustomizable id="cnta"/>
                         </f:facet>
                         <f:facet name="contentB">
                           <cust:panelCustomizable id="cntb"/>
                         </f:facet>
                       </pe:layoutCustomizable>
                     </af:panelGroupLayout>
                   </f:facet>
                 </af:panelStretchLayout>
                 <f:facet name="editor">
                   <pe:pageEditorPanel id="pep1"/>
                 </f:facet>
               </pe:pageCustomizable>
             </f:facet>
           </af:pageTemplate>
         </af:form>
       </af:document>
```

Note: The page definition file of the JSPX page, which defines the style, must contain the following syntax for the `id` and `Package` attributes. The Page service replaces these values with the relevant values when creating a page based on a custom style:

```
<?xml version="1.0" encoding="UTF-8" ?>
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
                version="11.1.1.41.30"
                id="ps_pagedefusage"
                Package="ps_package">
```

Alternatively, the JSPX or JSFF template file that you are creating can be based on an ADF template.

Example 7–14 shows the sample code of an ADF template.

Note: If you choose to design your own ADF templates and want to set page style, then you must have a `Panel Group Layout` or `Panel Stretch Layout` component as a direct child of the `Page Customizable`. Additionally, you must set the placeholder for `styleClass` and `inlineStyle`, as shown in **bold** below. The page style values replace the placeholder text.

Example 7–14 Sample Code of a Page Style that uses an ADF Template

```
<?xml version='1.0' encoding='utf-8'?>
<!-- Copyright (c) 2006, 2008, Oracle and/or its affiliates.
All rights reserved. -->
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:pe="http://xmlns.oracle.com/adf/pageeditor"
          xmlns:cust="http://xmlns.oracle.com/adf/faces/customizable"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich">
  <jsp:directive.page deferredSyntaxAllowedAsLiteral="true"/>
  <jsp:directive.page contentType="text/html;charset=utf-8"/>
  <f:view>
    <af:document title="#{pageDocBean.title}" id="docrt">
      <af:form usesUpload="true" id="f1">
        <af:pageTemplate

viewId="/oracle/webcenter/webcenterapp/view/templates/WebCenterAppShellTemplate.js
px"
                value="#{bindings.shellTemplateBinding}" id="T">
          <f:facet name="content">
            <pe:pageCustomizable id="pcl1">
              <af:panelStretchLayout id="psl2"
                                     styleClass="replace_with_scheme_name"
                                     inlineStyle="replace_with_inline_style">
                <f:facet name="center">
                  <af:panelGroupLayout id="pgl1"
                                       layout="scroll">
                    <pe:layoutCustomizable id="lc1"

showLayoutChanger="#{pageServiceBean.isEditMode}"
                                           text="#{uib_o_w_w_r_WebCenter.LABEL_
CHANGE_LAYOUT}"
                                           showIcon="false" type="oneColumn"
                                           shortDesc="#{uib_o_w_w_r_
WebCenter.LABEL_CHANGE_LAYOUT}">
                      <cust:panelCustomizable id="mainC"/>
                      <f:facet name="contentA">
                        <cust:panelCustomizable id="cnta"/>
                      </f:facet>
                      <f:facet name="contentB">
                        <cust:panelCustomizable id="cntb"/>
                      </f:facet>
                    </pe:layoutCustomizable>
                  </af:panelGroupLayout>
                </f:facet>
              </af:panelStretchLayout>
              <f:facet name="editor">
                <pe:pageEditorPanel id="pep1"/>
              </f:facet>
            </pe:pageCustomizable>
          </f:facet>
        </af:pageTemplate>
      </af:form>
    </af:document>
  </f:view>
</jsp:root>
```

Example 7–15 shows the sample code of a page style, `NewsTemplate.jspx`, that is based on an ADF template, `portalTemplate.jspx`. The `viewId` attribute of the `af:pageTemplate` tag provides the name of the ADF template used.

For information about using ADF templates, see the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework.*

Example 7–15   Sample Code of a Page Style on an ADF Template

```
<?xml version='1.0' encoding='utf-8'?>
<!-- Copyright (c) 2006, 2008, Oracle and/or its affiliates.
All rights reserved. -->
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:pe="http://xmlns.oracle.com/adf/pageeditor"
          xmlns:cust="http://xmlns.oracle.com/adf/faces/customizable"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich">
  <jsp:directive.page contentType="text/html;charset=utf-8"/>
  <f:loadBundle basename="oracle.webcenter.page.view.resource.PGUIBundle"
                var="res"/>
  <f:view>
    <af:document title="#{pageDocBean.title}">
      <af:form usesUpload="true">
          <af:pageTemplate viewId="/mytemplates/portalTemplate.jspx"
 value="#{bindings.pageTemplateBinding}">
      <f:facet name="pageContent">
            <pe:pageCustomizable id="pgc1">
              <af:panelStretchLayout id="psl1"
                                     styleClass="#{pageDocBean.CSSStyle}"
                                     inlineStyle="#{pageDocBean.inlineStyle}">
                <f:facet name="center">
                  <af:panelGroupLayout id="pgl1" layout="scroll">
                    <pe:layoutCustomizable id="lc1"
                        showLayoutChanger="#{pageServiceBean.isEditMode}"
                                           text="#{res['TEMPLATE.CHANGE_LAYOUT']}"
                                           showIcon="false" type="oneColumn"
                                           shortDesc="#{res['TEMPLATE.CHANGE_
LAYOUT']}">
                      <cust:panelCustomizable id="mainC">
                        <cust:showDetailFrame id="sdf2"
                                              showMinimizeAction="none"
                                              showMoveAction="none"
                                              showRemoveAction="none"
                                              displayHeader="false"
                                              displayShadow="false"
                                              stretchContent="false"
                                              shortDesc="#{null}"
                                              background="light"
                                              selectChild="no"
                                              showResizer="never"
                                              text="Text">
                          <af:richTextEditor id="rtepc" clientComponent="true"
                                             simple="true" label="#{null}"
                                             rows="20" readOnly="false"
                                             contentStyle="width:100%;"/>
                        </cust:showDetailFrame>
                      </cust:panelCustomizable>
                      <f:facet name="contentA">
                        <cust:panelCustomizable id="cnta"/>
                      </f:facet>
                      <f:facet name="contentB">
                        <cust:panelCustomizable id="cntb"/>
                      </f:facet>
                    </pe:layoutCustomizable>
                  </af:panelGroupLayout>
                </f:facet>
              </af:panelStretchLayout>
              <f:facet name="editor">
                <pe:pageEditorPanel/>
              </f:facet>
            </pe:pageCustomizable>
          </f:facet>
        </af:pageTemplate>
      </af:form>
    </af:document>
  </f:view>
</jsp:root>
```

To display a different style, or use a different template for a style, you must define it, for example, in `/mytemplates/templates.xml`. Place the directory `/mytemplates` under the web content root. For more information about the `templates.xml` file, see Section 7.3.4, "Specifying Styles."

Example 7–16 shows a sample `templates.xml` file.

Example 7–16 templates.xml File

```
<?xml version='1.0' encoding="UTF-8"?>
<templatesDef xmlns="http://xmlns.oracle.com/webcenter/page">
  <templateDef name="TemplateBlank.jspx"
               title="Blank"/>
  <templateDef name="TemplateNarrowLeft.jspx"
               title="Left Narrow Column Layout"/>
  <templateDef name="TemplateNarrowRight.jspx"
               title="Right Narrow Column Layout"/>
  <templateDef name="TemplateThreeColumn.jspx"
```

```
             title="Three Column Layout"/>
  <templateDef name="TemplateStretch.jspx"
             title="Stretch"/>
  <templateDef name="TemplateText.jspx"
             title="Text"/>
  <templateDef name="TemplateWeb.jspx"
             title="Web"/>
  <templateDef name="NewsTemplate.jspx"
             title="News"/>
</templatesDef>
```

7.6.1.2 How to Edit Create Page Dialog Styles

To edit the styles for the Create Page dialog and add a reference to the new page style:

1. Create the `/mytemplates/templates.xml` file, commenting out `templateDef` entries for any default page styles that you do not want to display.
2. In the `templateDef` entry for the style that you want to associate with a different template, edit the `name` and `type` attributes, as shown in the following example:

   ```
   <templateDef name="pstemplateview.jsff"
                title="News"
                icon="/images/news.png"
                type="taskflow"/>
   ```

   where

   - `name` is the name of the page style that you want to use.
   - `title` is the label displayed for the style in the Create Page dialog.
   - `type` is the type of page style. It can take the value `page` or `task flow`. The default value `page`, creates a page based on the style's associated template. The value `task flow` creates a task flow view of a page based on the style's associated template.
3. If you want to add a new style to the Create Page dialog, add a `templateDef` entry, as shown in the following example:

   ```
   <templateDef name="pstemplateview3.jsff"
                title="Rich Text"
                type="taskflow"
                forGroupSpace="false"/>
   ```
4. Save the XML file.

At runtime, the Create Page dialog displays the styles you specified. The pages and task flow page views users create using those styles are based on the JSPX or JSFF templates you defined.

7.6.2 Using ADF Templates

If your page template is based on an ADF template, you can configure the page creation task flow to use a different ADF template depending on different criteria, such as user or scope.

You can specify the ADF template to use for runtime page creation in either of the following ways:

- By using the `oracle_webcenter_page_createpage_adftemplate` parameter in the page creation task flow (see Section 7.3.4, "Specifying Styles").

- By using the `ADFTemplateViewID` parameter in the `createPage()` and `createTaskflow()` APIs.

Any JSPX page template that is based on an ADF template contains an `af:pageTemplate` tag with the `viewId` attribute. The `viewId` attribute contains the name of the ADF template. Example 7–15 shows a page template that is based on the ADF template `portalTemplate.jspx`.

When you specify an ADF template name for use during page creation, the Page service searches for the `af:pageTemplate` tag in the page template and updates the `viewId` attribute with the value you provided.

This section describes how to specify which ADF template to use under a given circumstance. It contains the following subsections:

- Section 7.6.2.1, "How to Specify the ADF Template Name Using the Task Flow Parameter"
- Section 7.6.2.2, "How to Specify the ADF Template Name Using API Parameters"

7.6.2.1 How to Specify the ADF Template Name Using the Task Flow Parameter

To specify the ADF template name using the task flow parameter:

1. Open the JSPX file containing the Page - Create New task flow.
2. Right-click the page name and select **Go to Page Definition**.
3. In the page definition file, under Executables, select the Create Page task flow, and click the **Edit** icon on the header.
4. The Edit Task Flow Binding dialog opens with the list of input parameters supported by the task flow.
5. Set the `oracle_webcenter_page_createpage_adftemplate` parameter.

   If you define a constant value, for example `${'/mytemplates/defaultTemplate2.jspx'}`, then this ADF template is used for all pages created using the create page task flow.

   However, by providing an EL value for the parameter, you can ensure that a different ADF template is used based on different criteria, such as user or scope.

   For example, assume you have two users, `user1` and `user2`, and want to use a different ADF template for pages created by each user. You must enter the value `${MyClass.ADFTemplate}` for the `oracle_webcenter_page_createpage_adftemplate` parameter and define the method `MyClass.getADFTemplate()` so that it returns different ADF templates based on which user has logged in.
6. Click **OK**.

7.6.2.2 How to Specify the ADF Template Name Using API Parameters

To specify the ADF template name using API parameters:

1. Create a page or a task flow, and specify the ID of the ADF template view activity.

   The page or task flow is subsequently created based on that ADF template.

   To create a page:

```
PageService.createPage(
     String pageType, String nameFormat, String title,
     String pageTemplate, String pageTemplatePath,
     String ADFTemplateViewID,
     String cssStyle, String schemeBGImage, String schemeBGColor)
```

where `ADFTemplateViewID` is the ID of the ADF template view activity.

To create a task flow:

```
PageService.createTaskflow(
     String nameFormat, String title,
     String pageTemplate, String pageTemplatePath,
     String ADFTemplateViewID,
     String cssStyle, String schemeBGImage, String schemeBGColor)
```

where `ADFTemplateViewID` is the ID of the ADF page template view activity.

7.6.3 Creating Styles for the Create Page Dialog

To use your own page styles, define them in `/mytemplates.templates.xml` file. If you want to maintain a separate XML file with a different set of page styles, you can create a new XML file and reference that from the Page - Create New task flow. For example, to display two different options to two users with different privileges, then you must create two XML files and define different styles in each XML file. You can then ensure that the Create Page dialog displays the respective options to each user.

This section describes how to create and reference a new style for the runtime Create Page dialog. It contains the following subsections:

- Section 7.6.3.1, "How to Create a Style for the Create Page Dialog"
- Section 7.6.3.2, "How to Reference a New Style from the Page - Create New Task Flow"

7.6.3.1 How to Create a Style for the Create Page Dialog

To create a style for the Create Page dialog:

1. In the `/mytemplates` directory, create a new XML file, for example `templates2.xml`.
2. Add a `templatesDef` element, and within that add a `templateDef` entry for each style that you want to include in the Create Page dialog.

   Example 7–17 shows the code entered into sample XML file `templates2.xml` that references the `pstemplateview.jsff` fragment you created in the previous section.

Example 7–17 Sample Code for the Create Page Dialog Style

```
<?xml version="1.0" encoding="UTF-8" ?>
- <templatesDef xmlns="http://xmlns.oracle.com/webcenter/page">
<templateDef name="pstemplateview.jsff" title="News" icon="/images/news.png"
             type="taskflow" />
  </templatesDef>
```

where

- `name` is the name of the page or page fragment to use as a template.
- `title` is the label for the style in the Create Page dialog.
- `type` is the type of template. It can take that value `page` or `taskflow`. The default value `page` creates a page based on the style's associated template. The value `taskflow` creates a task flow view of a page based on the style's associated template.

3. Save the XML file.

7.6.3.2 How to Reference a New Style from the Page - Create New Task Flow

To reference the new XML file from the Page - Create New task flow:

1. Open the JSPX page that contains the Page - Create New task flow.
2. Right-click the JSPX page, and select **Go to Page Definition**.
3. In the page definition file, under Executables, select the Create Page task flow, and click the **Edit** icon on the header.

   The Edit Task Flow Binding dialog opens with the list of input parameters supported by the task flow.
4. Provide the template file name for the `oracle_webcenter_page_createpage_templatefile` parameter.

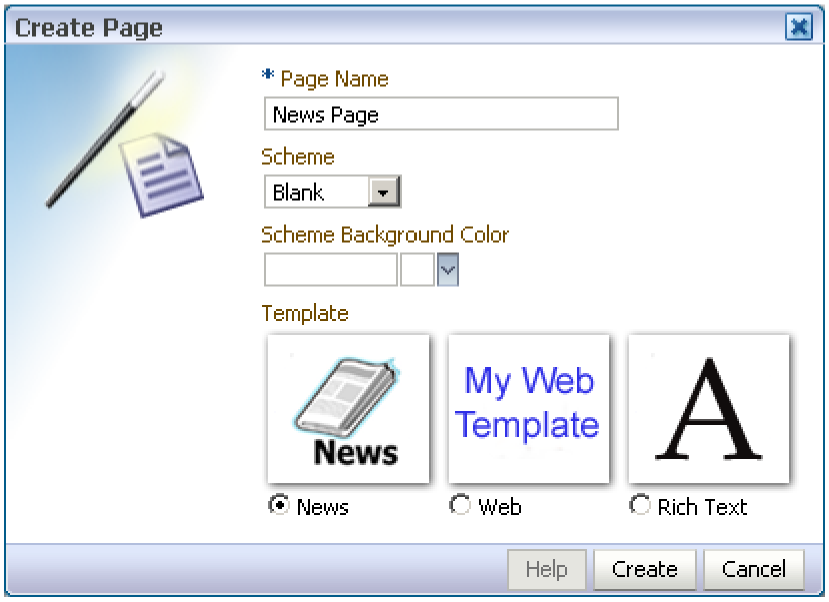
   If you define a constant value, for example `${'/mytemplates/templates2.xml'}`, then this template is used for the Create Page dialog at all times.

   However, by providing an EL value for the parameter, you can ensure that a different template is used for the dialog based on different criteria, such as user or scope.

   For example, assume you have two users, `user1` and `user2`, and want to display a different page style to each user. In this case, you must have two templates for the Create page dialog. Let us say you have created a new template, `templates2.xml`, and updated the default template, `templates.xml`. You can then specify `${TemplateBean.template}` for the `oracle_webcenter_page_createpage_templatefile` parameter.

   To use this value, you must first create the managed bean, `TemplateBean.java`, with method `getTemplate()`, which returns `templates.xml` for `user1` and `templates2.xml` for `user2`. At runtime, user1 and user2 are each shown different options in the Create Page dialog, as shown in Figure 7–16 and Figure 7–17.

Figure 7–16 Create Page Dialog using the templates.xml Style


Create Page

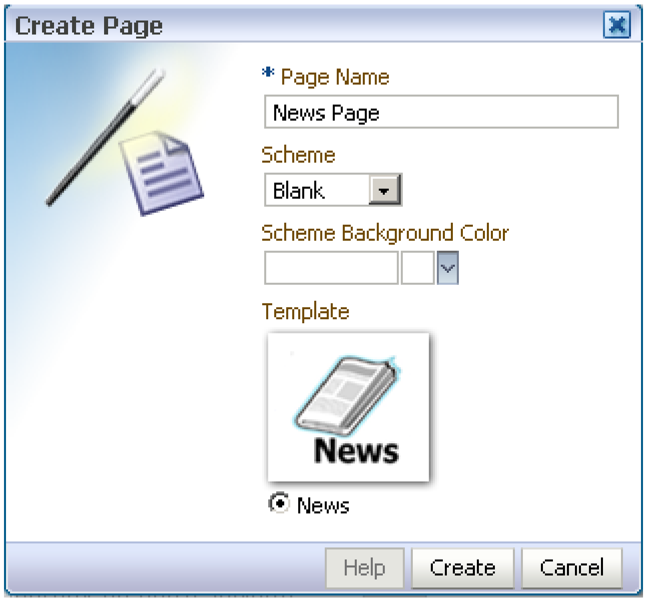
* Page Name: News Page

Scheme: Blank

Scheme Background Color

Template: [✓ News] Web Rich Text

Figure 7–17 Create Page Dialog using the templates2.xml Style


Create Page

* Page Name: News Page

Scheme: Blank

Scheme Background Color

Template: [✓ News]

5. Click **OK**.
6. Click **Save All** to save your work.

7.7 Customizing Page Service Views

This section describes various ways to customize Page service views. It contains the following subsections:

- Section 7.7.1, "Rendering Pages with ADF Faces Components"
- Section 7.7.2, "Managing User Security on Pages and Task Flows"

7.7.1 Rendering Pages with ADF Faces Components

You can use ADF Faces components to render runtime pages. For example, you can use ADF Faces components to render pages as tabs, links, and image links—or thumbnail views of pages. This section provides some examples of the types of pages you can create using ADF Faces components. It contains the following subsections:

- Section 7.7.1.1, "Rendering Pages as Tabs Using ADF Faces Components"
- Section 7.7.1.2, "Rendering Pages as Links Using ADF Faces Components"
- Section 7.7.1.3, "Rendering Pages as Image Links Using ADF Faces Components"

7.7.1.1 Rendering Pages as Tabs Using ADF Faces Components

When you design your application to render pages as tabs, you can use a range of ADF Faces components to accomplish this. This section provides information about the ADF Faces components that support rendering pages as tabs. It contains the following subsections:

- Section 7.7.1.1.1, "Rendering Pages as Tabs Using af:navigationPane"
- Section 7.7.1.1.2, "Rendering Pages as Tabs Using af:panelTabbed"

7.7.1.1.1 Rendering Pages as Tabs Using af:navigationPane Example 7–18 illustrates how pages can be rendered as tabs using `navigationPane` components.

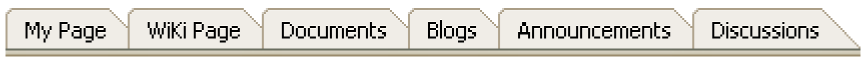
Here a method called `pages` from a custom bean (`MyPageServiceBean`) is used to get the list of pages. For more information about getting the list of pages with the Page service API, see the `getPages()` method in Page service API section.

Example 7–18 Rendering Pages as Tabs Using af:navigationPane

```
<af:navigationPane id="tabs">
  <af:forEach var="tab" items="#{MyPageServiceBean.pages}">
    <af:commandNavigationItem text="#{tab.title}"  id="cni1"/>
    </af:forEach>
  </af:navigationPane>
```

The output of the code depicted in Figure 7–18 renders as a series of tabs (Figure 7–18).

Figure 7–18 Render Pages as Tabs



7.7.1.1.2 Rendering Pages as Tabs Using af:panelTabbed Example 7–19 illustrates how pages can be rendered as tabs using `navigationPane` components.

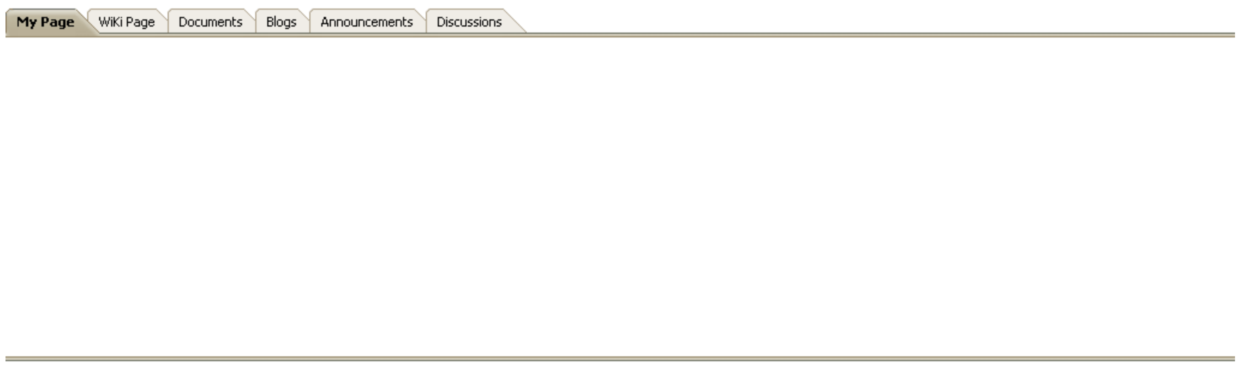
Here a method called `pages` from a custom bean (`MyPageServiceBean`) is used to get the list of pages. For more information about getting list of pages using Page service API, see the `getPages()` method in Page service API section.

Example 7–19 Rendering Pages as Tabs Using af:panelTabbed

```
<af:panelTabbed id="pt1" >
    <af:forEach var="tab" items="#{ MyPageServiceBean.pages}">
        <af:showDetailItem text="#{tab.title}" id="sdi1"/>
     </af:forEach>
   </af:panelTabbed>
```

The output of the code in Example 7–19 renders as a series of tabs (Figure 7–19).

Figure 7–19 Render Pages as af:panelTabbed Component



7.7.1.2 Rendering Pages as Links Using ADF Faces Components

This section provides examples of using ADF Faces Components to render pages as different types of links. It includes the following subsections:

- Section 7.7.1.2.1, "Rendering Pages as Links Using af:commandLink"
- Section 7.7.1.2.2, "Rendering Pages as Links Using af:goLink"

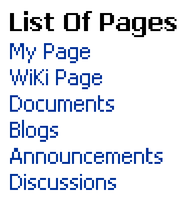
7.7.1.2.1 Rendering Pages as Links Using af:commandLink Example 7–20 illustrates how pages can be rendered as links using `af:commandLink`.

Example 7–20 Rendering Pages as Links Using af:commandLink

```
<af:panelGroupLayout id="pgl1" layout="vertical">
  <af:forEach var="tab" items="#{MyPageServiceBean.pages}">
     <af:commandLink text="#{tab.title}" id="cl1"/>
   </af:forEach>
</af:panelGroupLayout>
```

The output of the code depicted in Example 7–20 renders as a list of links (Figure 7–20).

Figure 7–20 Render Pages as Links



7.7.1.2.2 Rendering Pages as Links Using af:goLink Example 7–21 illustrates how pages can be rendered as links using `af:goLink`.

Example 7–21 Rendering Pages as Links Using af:goLink

```
<af:panelGroupLayout id="pgl2" layout="vertical">
     <af:activeOutputText value="List Of Pages" id="aot2"
                            inlineStyle="font-weight:bold; font-size:small;"/>
      <af:forEach var="tab" items="#{pageServiceBean.pages}">
```

```
        <af:goLink text="#{tab.title}" id="gl1"/>
      </af:forEach>
    </af:panelGroupLayout>
```

The output of the code depicted in Example 7–21 renders as a list of links (Figure 7–20).

7.7.1.3 Rendering Pages as Image Links Using ADF Faces Components

Image links are linked thumbnail views of a page that users click to access the full page view. Example 7–22 illustrates how pages can be rendered as image links.

Example 7–22 Rendering Thumbnail Views of Pages

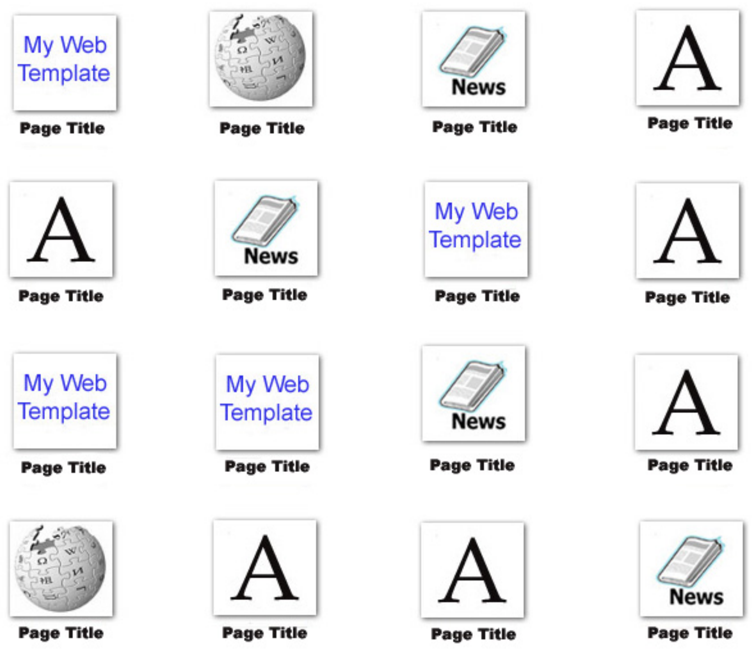
```
  <af:panelGroupLayout id="pg1_11" layout="scroll">
          <af:panelGroupLayout id="pg11" layout="horizontal" halign="center"
                               valign="middle">
            <h:panelGrid id="ds" columns="4" cellpadding="10" cellspacing="5"
                         style="text-align: center;" rowClasses="bottomAlign">
              <af:forEach varStatus="stat" begin="0"
end="#{MyPageServiceBean.pageCount}">
                <af:panelGroupLayout id="sdsd" layout="vertical"
                                     halign="center">
                  <af:commandImageLink text="" id="cil1"
                                       icon="/<image_name>}"/>
                  <af:activeOutputText value="Page Title" id="aot3"/>
                </af:panelGroupLayout>
              </af:forEach>
            </h:panelGrid>
          </af:panelGroupLayout>
        </af:panelGroupLayout>
```

In Example 7–22, `pageCount` is a method from a custom bean (`MyPageServiceBean`) that gets the number of pages in the current scope. Users must write this method on their own using the other methods from Page service APIs.

In Example 7–22, `<image_name>` is the name of the image with which you associate your page.

The output of the code in Example 7–22 is rendered as a linked series of thumbnail views of a page (Figure 7–21).

Figure 7–21 Render Pages as Image Links



7.7.2 Managing User Security on Pages and Task Flows

You can manage access to pages and task flows either through the Security Panel, which is a part of Composer, or through Page service APIs.

For more information about runtime security, see the "Setting Page Access" section in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

For more information about setting page access with the Page service APIs, see *Oracle Fusion Middleware Web 2.0 Services Java API Reference for Oracle WebCenter*.

7.8 Introduction to the Page Service APIs

The Page service provides the following APIs:

- Create Page

  Create Page at a user-defined Scope
- Delete Page

  Delete Page with Page Path API
- Create Task Flow

  Create Task Flow at a user-defined Scope
- Delete Task Flow
- Create Scope
- Delete Scope
- Copy Page
- Get Page List

- Change Page Title
- Change Hidden Status
- Change Page Scheme (name, background color, background image)
- Set Page Access

For more information about the Page service APIs, see *Oracle Fusion Middleware Web 2.0 Services Java API Reference for Oracle WebCenter*.

7.8.1 Using the Page Service APIs

This section provides information about Page service APIs. It includes information about the location of these APIs, how to make your application ready to use them, and how to use them to create pages. It contains the following subsections:

- Section 7.8.1.1, "Location of the Page Service APIs"
- Section 7.8.1.2, "How to Set up Your Application to Use the Page Service APIs"
- Section 7.8.1.3, "Example: How to Create a Page"

7.8.1.1 Location of the Page Service APIs

The Page service APIs are located in the `oracle.webcenter.page.model.PageService` class.

Example 7–23 Using Page Service APIs to Create a Task Flow View Page

```
PageDef createTaskflow(String nameFormat, String title, String pageTemplate,
  String pageTemplatePath, String cssStyle, String schemeBGImage, String
schemeBGColor)
  throws DuplicateNameException, InvalidNameException, LockUnavailableException;
```

where

- `nameFormat` is the name format of the task flow; it can contain a subpath, such as `users/UserA/taskflow{0}.jspx`.
- `title` is the title of the task flow.
- `pageTemplate` is the template used for the task flow.
- `pageTemplatePath` is the path for the task flow template.
- `cssStyle` is the cascading style sheet file for this task flow.
- `schemeBGImage` is the background image for the custom scheme.
- `schemeBGColor` is the background color for the custom scheme.

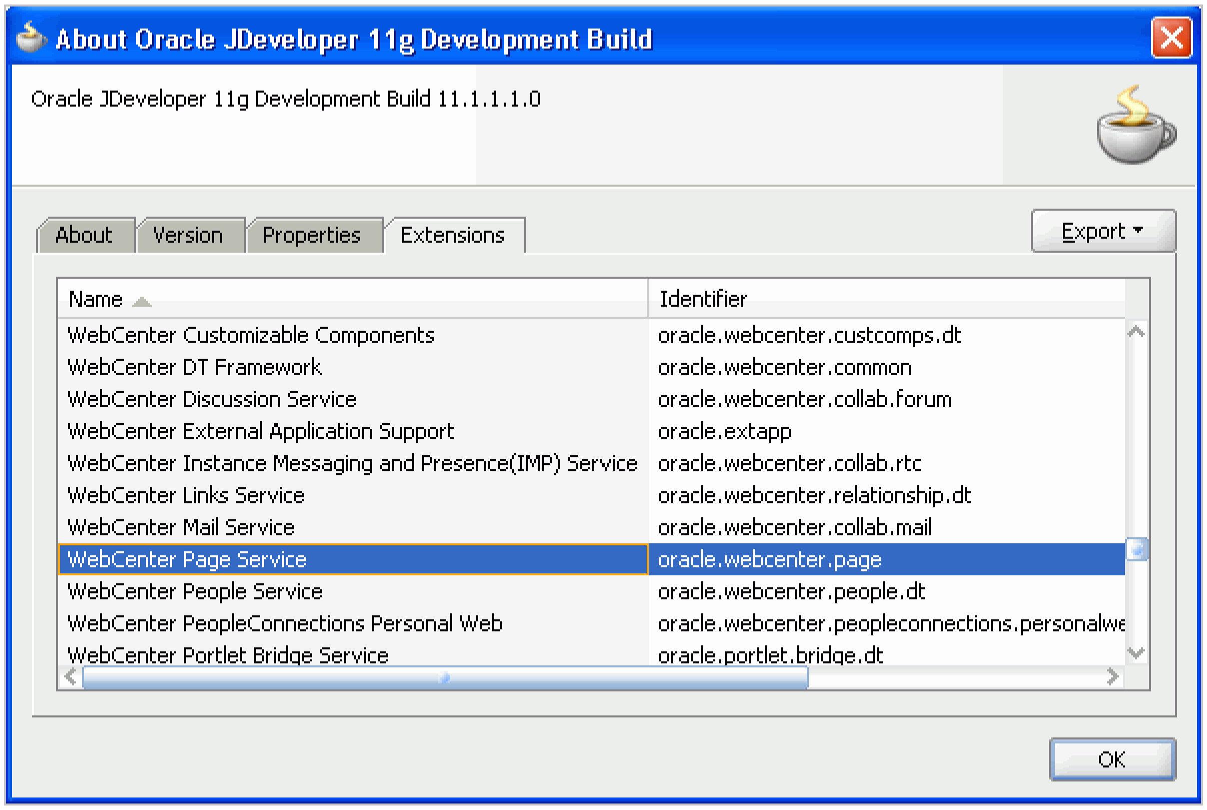
It returns the `PageDef` of the newly created task flow and throws the following:

- `InvalidNameException` if the name supplied cannot be used for update.
- `DuplicateNameException` if the page with the same name already exists.
- `LockUnavailableException` if another page or task flow operation is in progress.

7.8.1.2 How to Set up Your Application to Use the Page Service APIs

To use the Page service APIs, the WebCenter extension must be present in your JDeveloper application. In JDeveloper, go to the **Help - About** menu, and click the **Extensions** tab. Look for the WebCenter Page Service, as shown in Figure 7–22.

Figure 7–22 JDeveloper Extensions


About Oracle JDeveloper 11g Development Build
Oracle JDeveloper 11g Development Build 11.1.1.1.0
About
Version
Properties
Extensions
Export
OK

Name	Identifier
WebCenter Customizable Components	oracle.webcenter.custcomps.dt
WebCenter DT Framework	oracle.webcenter.common
WebCenter Discussion Service	oracle.webcenter.collab.forum
WebCenter External Application Support	oracle.extapp
WebCenter Instance Messaging and Presence(IMP) Service	oracle.webcenter.collab.rtc
WebCenter Links Service	oracle.webcenter.relationship.dt
WebCenter Mail Service	oracle.webcenter.collab.mail
WebCenter Page Service	oracle.webcenter.page
WebCenter People Service	oracle.webcenter.people.dt
WebCenter PeopleConnections Personal Web	oracle.webcenter.peopleconnections.personalwe
WebCenter Portlet Bridge Service	oracle.portlet.bridge.dt

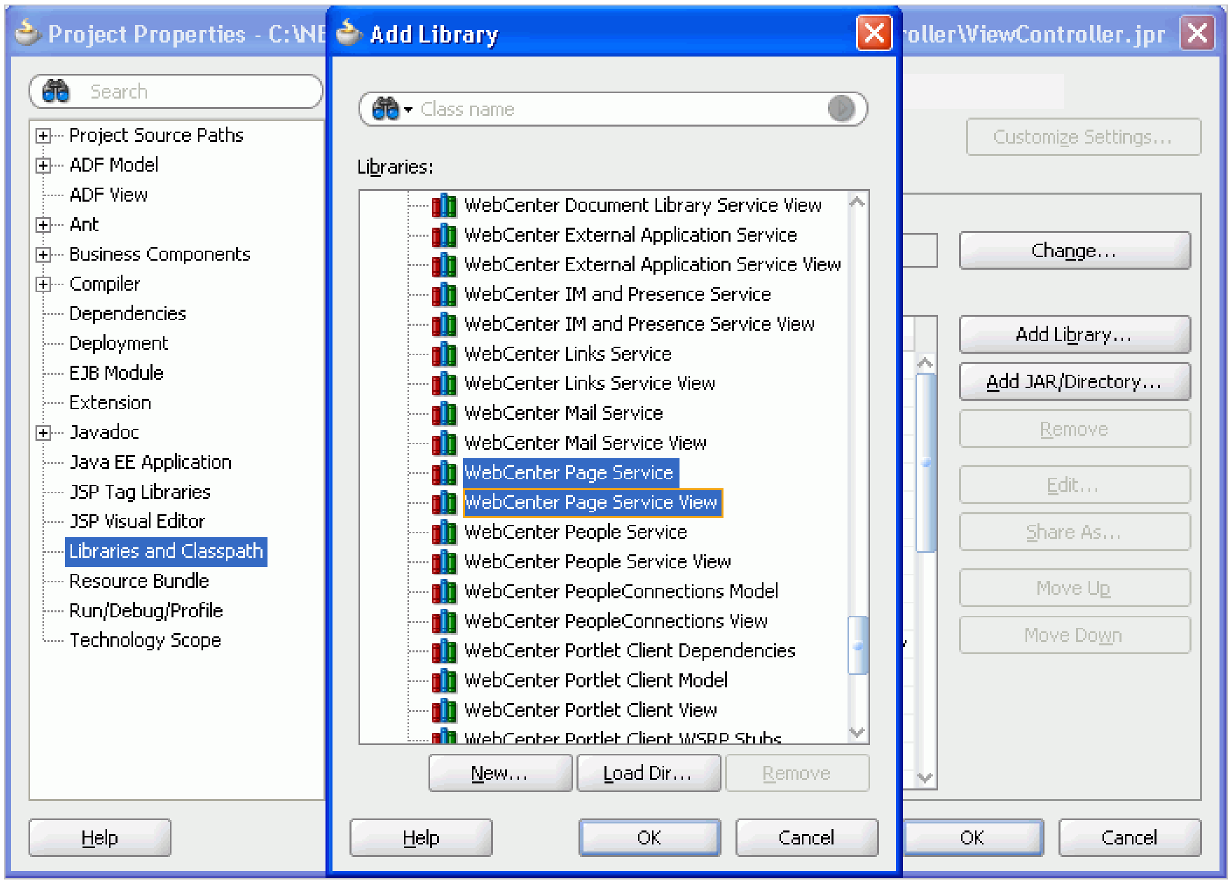
Note: For information about adding the WebCenter extension, see Chapter 3, "Preparing Your Development Environment."

After confirming that the Page service is included with the WebCenter extension, you must add the Page service libraries to the project.

To add Page service libraries to your project:

1. Right-click your **ViewController** project and select **Project Properties** and then **Libraries and Classpath**.
2. Click **Add Library** and select the **WebCenter Page Service** and **WebCenter Page Service View** libraries, as shown in Figure 7–23.

Figure 7–23 Adding Page Service Libraries



3. Click **OK**.

If you do not want to use JDeveloper as the code editor, then you can set the CLASSPATH to the following jar files:

- `Jdev_Home/jdeveloper/webcenter/jlib/pagem.jar`
- `Jdev_Home/jdeveloper/webcenter/jlib/pages.jar`
- `Jdev_Home/jdeveloper/webcenter/jlib/page-service-view.jar`
- `Jdev_Home/jdeveloper/webcenter/jlib/page-service-skin.jar`

7.8.1.3 Example: How to Create a Page

This section provides an example of how to create a page using the Page service APIs. Your requirements may call for different methods. Use the way the `createPage` method is invoked in this sample code as a model for invoking the methods you require on the `PageService` object.

Typically, pages created using Page service APIs are based on templates. Before you start creating pages using the APIs, you must ensure that each template file, for example `MyPageTemplate1.jspx`, has a page definition file, in this case `MyPageTemplate1PageDef.xml`, in the same directory as the JSPX page.

Example 7–24 shows how to use the `createPage() API` to create a scope named `MyScope`, create an MDS session instance, instantiate the Page service class with this new scope, and then create a page within this scope. The new page is created based on the `MyPageTemplate1.jspx` template.

Example 7–24 Sample Showing How to Create a Page

```
...

public void createMyPage
{
    try
    {
      MDSSession mdsSess =
(MDSSession)ADFContext.getCurrent().getMDSSessionAsObject();
      PageServiceConfig config = new PageServiceConfig(mdsSess, DEFAULT_SCOPE);
      mPageService = PageServiceFactory.createInstance(config);

      String pageNameFormat ="Page{0}.jspx";
      String pageTitle ="New Page";

      //
      // The template used to create the new page is
/mytemplates/MyPageTemplate1.jspx
      // path "/mytemplates" should be defined as a MDS namespace.
      // Also MyPageTemplate1.jspx should have a pageDef called
"MyPageTemplate1PageDef.xml and
      // it has the same path as the jspx file.

      PageDef newPage = mPageService.createPage(
        PERSONAL_USER_PAGE, pageNameFormat, pageTitle,
        "MyPageTemplate1.jspx",
        "/mytemplates/",
        null, null, null);

    }
    catch(Exception e)
    {
      // Handle exception
    }
}

...
```

Note: Pages and task flows created using Page service APIs do not have permissions granted by default. You must explicitly define permissions on the page.

For more information about the Page service APIs, see *Oracle Fusion Middleware Web 2.0 Services Java API Reference for Oracle WebCenter*.

7.9 Page Service Samples

The Custom Portal Demo sample illustrates some Page service functionality, including the following:

- How to add the Page - Create New task flow
- How to configure task flow parameters
- How to use an ADF template
- How to use custom styles
- How to manage pages

The Custom Portal Demo sample is posted on the Oracle WebCenter Suite 11g Demonstrations and Samples page on the Oracle Technology Network (OTN) at:

`http://www.oracle.com/technology/products/webcenter`

8 Integrating Content

Oracle WebCenter enables content integration through:

- Content Repository data controls, which enable read-only access to a content repository, and maintain tight control over the way the content displays in the application.
- The Documents service, which enables users to view and manage documents in your organization's content repositories.

This chapter describes how to configure Java Content Repository (JCR) connections and data controls. It discusses, through examples, how to use these JCR data controls to integrate and publish decentralized content in your WebCenter application. The JCR data controls enable you to publish content from any JCR 1.0 (JSR 170) Level 1-compliant repository, such as Oracle Content Server and Oracle Portal. For more information about managing and including content in your WebCenter applications, see:

- Chapter 14, "Integrating the Documents Service" to integrate the Documents service in custom WebCenter applications at design time to create a user-friendly interface to manage documents at runtime.
- "Managing Content Repositories" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter* to configure and manage content repositories used by WebCenter Spaces and any other custom WebCenter application.
- "Working with the Documents Service" in *Oracle Fusion Middleware User's Guide for Oracle WebCenter* to work with the Documents service and task flows at runtime in WebCenter Spaces and any other custom WebCenter application.

This chapter includes the following sections:

8.1 Introduction to Content Integration

The content integration capabilities of Oracle WebCenter Suite enable you to integrate decentralized content located across multiple repositories through JCR adapters using JCR APIs. JCR APIs enable independent access to content, regardless of the underlying repository or the type of the content; for example, documents, relational content, and so on. The JCR 1.0 API, as defined in JSR 170, provides a set of basic capabilities for reading, writing, browsing, and searching content. The JCR data controls also enable you to connect to and read from other JCR 1.0 repositories. To add content available in Oracle Content Server, Oracle Portal, and file systems to JSF pages, you first create connections to the repositories, then use the connections to create data controls based on the repositories.

The Documents service is another way of integrating content within a WebCenter application. Both content data controls and the Documents service use content repository connections described in this chapter. For information about the Documents service, see Section 14.2.2, "Adding the Documents Service at Design Time".

8.1.1 Overview of Content Data Controls and Adapters

A data control is a container for all the data objects, collections, methods, and operations used to create user interface (UI) components within your WebCenter application. The data controls provide you with easy-to-use methods that you can drag and drop onto JSF pages to publish content as ADF components, such as URLs, files, and folders. While methods, parameters, and default attributes to publish content are similar across all JCR data controls, the content integration module gives you the flexibility to customize attributes based on your requirements. In your WebCenter application, you can create content repository data controls that connect to different repositories through the following adapters:

- Oracle Content Server: The Oracle Content Server adapter is used to integrate content from an automated information system managed by Oracle Content Server. This system enables sharing, managing, and distributing of business information through a Web site as a common access point.
- Oracle Portal: The Oracle Portal adapter is used to integrate content from a content repository located in the Oracle Portal schema.
- File System: The File System adapter is used to add content located on your operating system's file system to JSF pages.

Note: The File System adapter is intended to be used in the development environment only.

The Oracle Content Server, Oracle Portal, and file system adapters are bundled with Oracle WebCenter Extension bundle.

8.1.2 Overview of Content Data Control Methods, Parameters, and Default Attributes

Each type of content data control contains methods and parameters to publish content as links, tables, files, and folders, and to add search and advanced search capabilities for your content. Parameters include two types of attributes: default and custom. The default attributes are common across File System, Oracle Portal, and Oracle Content Server data controls. The custom attributes are requirement-specific and can be added while creating content data controls.

The following sections describe the methods, parameters, and default attributes that are common across File System, Oracle Portal, and Oracle Content Server data controls:

- Section 8.1.2.1, "The getItems Method"
- Section 8.1.2.2, "The search Method"

- Section 8.1.2.3, "The advancedSearch Method"
- Section 8.1.2.4, "The getURI Method"
- Section 8.1.2.5, "The getAttributes Method"

8.1.2.1 The getItems Method

The `getItems` method returns the files and folders stored starting at a particular location in the repository. This method enables you to publish content in forms, tables, and hierarchical trees. Using this method, you can also create navigation lists and buttons.

Table 8–1 describes the parameters of the `getItems` method.

Table 8–1 Parameters of the getItems Method

Parameter	Description
`path`	Defines the starting point for `getItems`.
`type`	Specifies what should be returned: only files, only folders, or any object.

The `getItems` method returns the attributes described in Table 8–2.

Table 8–2 Return Attributes of the getItems Method

Parameter	Description
`name`	Describes the name of the returned file or folder.
`path`	Describes the location of the returned file or folder within the content repository.
`URI`	The direct access URL of a file or folder.
`primaryType`	Describes whether the returned object is a file or folder or some other type.

8.1.2.2 The search Method

The `search` method enables you to create a simple search by name pattern or keyword.

Table 8–3 describes the parameters of the `search` method.

Table 8–3 Parameters of the search Method

Parameter	Description
`path`	Starting path of the search.
`isRecursive`	Specifies whether only the specified folder (=`false`) or the whole tree starting at the specified path (=`true`) should be searched. The default value is `false`, that is, if this field is left blank, then the search is performed in the specified folder. Only if the value is `true`, the search is performed in the entire hierarchy.
`keyword`	Search keyword for full text search.
`namePattern`	Pattern search on name. Use the `%` wildcard to match any number of characters and the _ wildcard to match one character.

The `search` method returns the attributes described in Table 8–4.

Table 8–4 Return Attributes of the search Method

Parameter	Description
`name`	Describes the name of the returned file or folder.
`path`	Describes the location of the returned file or folder within the content repository.
`URI`	The direct access URL of a file or folder.
`primaryType`	Describes whether the returned object is a file or folder or some other type.

8.1.2.3 The advancedSearch Method

The `advancedSearch` method enables you to perform an advanced search by creating a set of search criteria out of any available attribute.

Table 8–5 describes the parameters of the `advancedSearch` method.

Table 8–5 Parameters of the advancedSearch Method

Parameter	Description
`path`	Starting path of the search.
`isRecursive`	Specifies whether only the specified folder (=`false`) or the whole tree starting at the specified path (=`true`) should be searched. The default value is `false`, that is, if this field is left blank, then the search is performed in the specified folder. Only if the value is `true`, the search is performed in the entire hierarchy.
`keyword`	Search keyword for full text search.
`namePattern`	Pattern search on name. Use the `%` wildcard to match any number of characters and the _ wildcard to match one character.
`matchAny`	Specifies whether all predicates (=`false`) or any predicate (=`true`) should be matched.
`predicates`	A collection of `SimplePredicate` parameters that consist of attributes, comparators, and values.
`type`	Specifies what should be returned: only files, only folders, or any object.

The `advancedSearch` method returns the attributes described in Table 8–6.

Table 8–6 Return Attributes of the advancedSearch Method

Parameter	Description
`name`	Describes the name of the returned file or folder.
`path`	Describes the location of the returned file or folder within the content repository.
`URI`	The direct access URL of a file or folder.
`primaryType`	Describes whether the returned object is a file or folder or some other type.

8.1.2.4 The getURI Method

The `getURI` method returns the URI attribute, which is the direct access URL of the file or folder. Its `path` parameter describes the path to the object. You can use this method to create links to content and to inline content in your page. The `getURI` method returns the `URI` attribute.

8.1.2.5 The getAttributes Method

The `getAttributes` method returns the list of attributes and their values for a given file or folder. Its `path` parameter describes the path to the object.

Table 8–7 describes the attributes that the `getAttributes` method returns.

Table 8–7 Return Attributes of the getAttributes Method

Parameter	Description
`name`	Name of the attribute.
`value`	Value of the attribute.

8.2 Configuring Content Repository Connections

This section describes how to configure content repository connections based on file system, Oracle Portal, and Oracle Content Server adapters. Content repository connections are required to configure data controls so that the repository content can be published on JSF pages. For information about configuring and using data controls, see Section 8.4, "Configuring Content Data Controls for JCR Adapters" and Section 8.5, "Integrating Content Using Content Data Controls: Examples".

Connections can be created under Application Resources in the Application Navigator and under IDE Connections in the Resource Palette, as described in Section 3.9, "Accessing Connection Wizards".

This section includes the following:

- Section 8.2.1, "How to Create a Content Repository Connection Based on the Oracle Content Server Adapter"
- Section 8.2.2, "How to Create a Content Repository Connection Based on the Oracle Portal Adapter"
- Section 8.2.3, "How to Create a Content Repository Connection Based on the File System Adapter"
- Section 8.2.4, "What Happens When You Create a Repository Connection"
- Section 8.2.5, "What You May Need to Know When Creating a Repository Connection"
- Section 8.2.6, "How to Edit a Common Repository Connection"
- Section 8.2.7, "How to Edit a WebCenter Application-Specific Content Repository Connection"
- Section 8.2.8, "How to Use an Existing Repository Connection for a New WebCenter Application"

8.2.1 How to Create a Content Repository Connection Based on the Oracle Content Server Adapter

This section describes how to configure an Oracle Content Server-based content repository connection. To successfully perform the subsequent steps, you must use Oracle Content Server release 10.1.3.4.1. For information about how to configure

Oracle Content Server for Oracle WebCenter, see the section "Oracle Content Server Prerequisites" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

To create an Oracle Content Server-based repository connection:

1. In Oracle JDeveloper, open your WebCenter application.
2. In the Application Resources panel, right-click **Connections** and choose **New Connection**, then **Content Repository**.
3. In the Create Content Repository Connection dialog, specify a connection name for the connection, for example, `MyOCSConnection`.

   Tip: You can choose between creating application-specific connections in the Application Resources panel and creating common connections in the Resource Palette (IDE Connections).

4. Select **Oracle Content Server** from the **Repository Type** box.
5. Select **Set as primary connection for Documents service**, if you intend to make it the default connection for the Documents service. Selecting this option writes a configuration to the `adf-config.xml` file that identifies this connection as the default connection for the Documents service. If a Document Library task flow is used without any `connectionName` input parameter, then this connection is used. For information about the Documents service, see Chapter 14, "Integrating the Documents Service".

   Note: To be able to set a connection as primary connection for the Documents service, you must select the **Application Resources** option in **Create Connection In**. This is because connections created as IDE connections cannot be used directly, they must be added to applications.

6. Under **Configuration Parameters**, enter values for the parameters, as shown in Table 8–8 and Figure 8–1.

Table 8–8 Configuration Parameters for Oracle Content Server

Configuration Parameters	Values
CIS Socket Type	Determines whether the client library connects on the Content Server listener port or the Web server filter. It accepts `socket`, `socketssl`, or `web`. `socket`: Uses an `intradoc` socket connection to connect to the Oracle Content Server. The client IP address must be added to the list of authorized addresses in the Oracle Content Server. `socketssl`: Uses an `intradoc` socket connection to connect to the Oracle Content Server that is secured using the SSL protocol. The client's certificates must be imported in the server's trust store for the connection to be allowed. `web`: Uses an HTTP(S) connection to connect to the Oracle Content Server. Table 8–9 includes more information on the configuration parameters required for each CIS socket type.
Server Host Name	Host name of the system where the Oracle Content Server is running. For example, `mycontentserver.mycompany.com`.

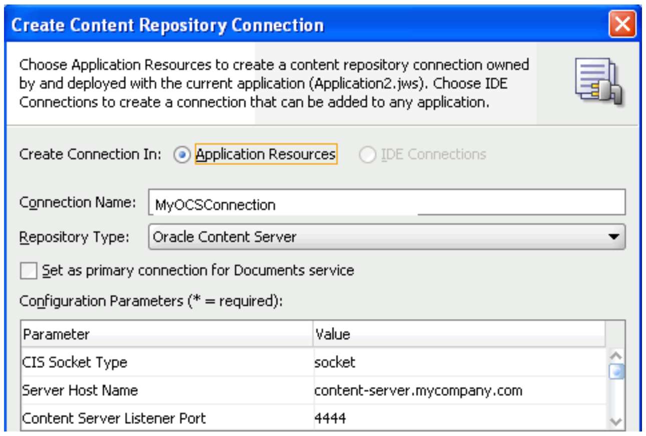
Table 8–8 (Cont.) Configuration Parameters for Oracle Content Server

Configuration Parameters	Values
Content Server Listener Port	Port of the server specified in the **Server Host Name** field. For example, `4444`.
URL of the Web Server Plugin	If the CIS socket type is `web`, then the URL must be in this format: `http://`*`host_name`*`/`*`web_root`*`/`*`plugin_root`*. For example, `http://mycontentserver/cms/idcplg`
KeyStore File Location	Location of key store that contains the private key used to sign the security assertions. The key store location must be an absolute path. For example, `c:\keys\keystore.xyz`.
KeyStore Password	The password required to access the keystore.
Private Key Alias	The client private key alias in the keystore. The key is used to sign messages to the server. The public key corresponding to this private key must be imported in the server keystore.
Private Key Password	The client private key password required to retrieve the key from the keystore.

Table 8–9 Oracle Content Server Connection Parameters for Each CIS Socket Type

Connection Parameters	CIS Socket Type: web	CIS Socket Type: socket	CIS Socket Type: socketssl
Server Host Name	Not applicable	Optional Defaults to localhost.	Optional Defaults to localhost.
Content Server Listener Port	Not applicable	Port specified for the incoming provider in the server. Defaults to 4444.	Port specified for the `sslincoming` provider in the server. Defaults to `4444`.
URL of the Web Server Plugin	Mandatory	Not applicable	Not applicable
Key Store and Private Key	Not applicable	Not applicable	Mandatory
Identity Propagation	Not supported at runtime	Supported For testing purpose, connects as guest if no/invalid user name is specified.	Supported For testing purpose, connects as guest if no/invalid user name is specified.
External Application	Mandatory	Supported Password is not used.	Supported Password is not used.
User Name and Password	Supported	Supported The password is not verified during the test connection operation.	Supported The password is not verified during the test connection operation.

Figure 8–1 Content Repository Connection - Oracle Content Server



Choose Application Resources to create a content repository connection owned by and deployed with the current application (Application2.jws). Choose IDE Connections to create a connection that can be added to any application.

Create Connection In: [✓ Application Resources] IDE Connections

Connection Name: MyOCSConnection

Repository Type: Oracle Content Server

Set as primary connection for Documents service

Configuration Parameters (* = required):

Parameter	Value
CIS Socket Type	socket
Server Host Name	content-server.mycompany.com
Content Server Listener Port	4444

7. In the **Login Timeout (ms)** field, specify the time in milliseconds. This timeout determines how long the application must wait when trying to establish a session with the content repository. If the network connection or the content repository server to which you are trying to connect is slow, then consider overriding the default value specified in this field.

8. Select the applicable authentication method. See Section 8.2.5.1, "What You Should Know About Using Identity Propagation and External Application Authentication Methods" for information. If you select the Identity Propagation authentication method, then you must configure secure socket layer (SSL) in Oracle Content Server. For information, see the section "Oracle Content Server - Security Considerations" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

9. **User Name** and **Password** fields under the **Specify login credentials for the current JDeveloper session** checkbox are optional for the Identity Propagation authentication method.

> **Note:** If the **Specify login credentials for the current JDeveloper session** checkbox is selected, then the credentials you entered are used. If the checkbox is not selected, then the connection is tested using External Application credentials (if they exist), otherwise null credentials are used.
>
> If you have selected **External Application** for authentication and also specified either public or shared credentials, then you can leave these fields blank, as the public or shared credentials can be used to login at design time. However, if you have selected **External Application** but have not specified public or shared credentials, then you must specify user name and password here.

10. Click **OK**.

11. In the Application Resources panel, expand the repository connection you just created. The Connection Credentials dialog displays.

Note: The Connection Credentials dialog displays if you do not save the credentials, that is, you do not select the **Specify login credentials for the current JDeveloper session** checkbox, or if you do not specify an External Application service that uses public or shared credentials.

---

12. Enter the **User Name** and **Password** for the Oracle Content Server connection and click **OK**. The connection displays under the Application Resources panel.

8.2.2 How to Create a Content Repository Connection Based on the Oracle Portal Adapter

This section describes how to create a content repository connection based on the Oracle Portal adapter. Before creating a repository connection, see Section 8.2.5, "What You May Need to Know When Creating a Repository Connection". You can use this connection to configure a content data control that will enable you to add content from the Oracle Portal repository to JSF pages.

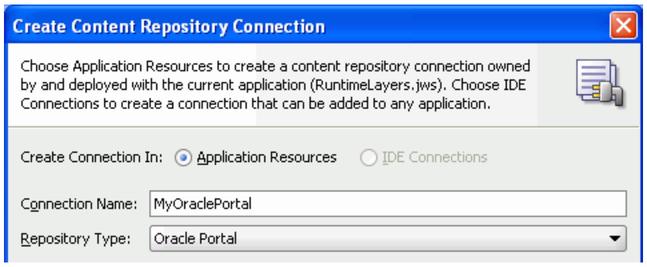
To create an Oracle Portal repository connection:

1. In Oracle JDeveloper, open your WebCenter application.
2. In the Application Resources panel, right-click **Connections** and choose **New Connection** and then **Content Repository**.
3. In the Create Content Repository Connection dialog, specify a name for the connection; for example, `MyOraclePortal`.

   Tip: You can choose between creating application-specific connections in the Application Resources panel and creating common connections in the Resource Palette (IDE Connections).

4. Select **Oracle Portal** from the **Repository Type** box, as shown in Figure 8–2.

Figure 8–2 Content Repository Connection - Oracle Portal



Choose Application Resources to create a content repository connection owned by and deployed with the current application (RuntimeLayers.jws). Choose IDE Connections to create a connection that can be added to any application.

Create Connection In: [✓ Application Resources] IDE Connections

Connection Name: MyOraclePortal

Repository Type: Oracle Portal

5. Select **Set as primary connection for Documents service**, if you intend to make it the default connection for the Documents service. Selecting this option writes a configuration to the `adf-config.xml` file that identifies this connection as the default connection for the Documents service. If a document library task flow is used without any `connectionName` input parameter, then this connection is used. For information about the Documents service, see Chapter 14, "Integrating the Documents Service".

Note: To be able to set a connection as primary connection for the Documents service, you must select the **Application Resources** option in **Create Connection In**. This is because connections created as IDE connections cannot be used directly, they must be added to applications.

---

6. Select a connection from the **Database Connection** dropdown list, or create a new connection to the Oracle Portal schema:

   a. In the Create Database Connection wizard, specify the connection name and type.

   b. Enter the database user name and password of the Oracle Portal schema. By default, the user is PORTAL.

   c. Under **Oracle (JDBC) Settings**, if you intend to specify a custom JDBC URL, then select the **Enter Custom JDBC URL** checkbox, and specify the database URL of the portal schema in the `jdbc` format:

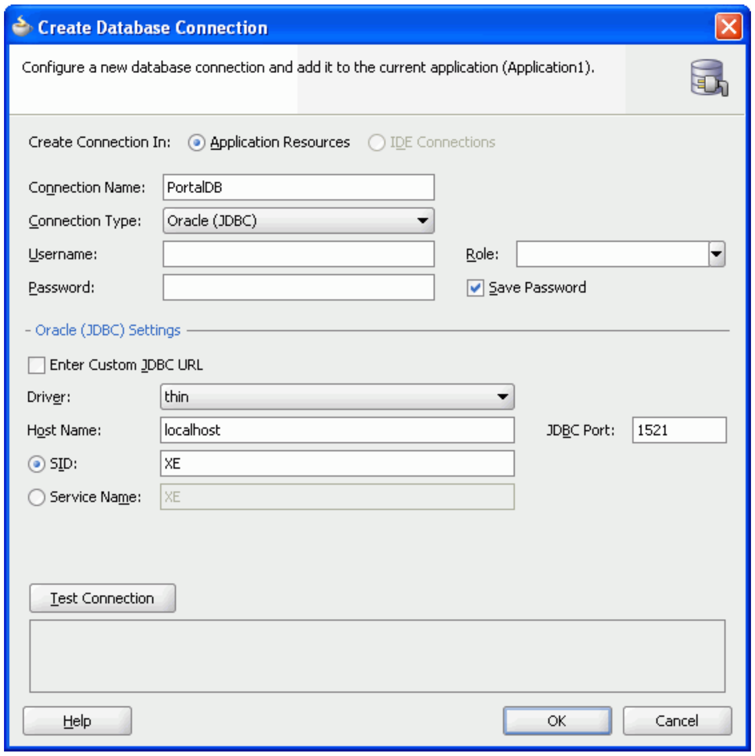
   ```
   jdbc:oracle:thin:@dbhost:dbport:dbsid
   ```

   d. If you did not perform the above step, then select a driver, enter the host name, JDBC port, SID, and service name, as shown Table 8–10 and Figure 8–3.

Table 8–10 Parameters for Creating Oracle Portal-based Content Data Control

Parameters	Description
Driver	There are two types of drivers: thin and oci8. The thin driver can be used to connect to Oracle Database release 8*i* or later databases with TCP/IP network protocols. This driver is included in the default Oracle JDBC library for all projects. The oci8 driver is used when creating a Java application that runs against an Oracle Application Server. This is a thick driver optimized for the Oracle Database. It is a mix of Java and native code. This driver handles any database protocol (TCP, IPX, BEQ, and so on). It is recommended for applications that are run from the computer on which they are stored.
Host Name	Name of the Oracle Database. Use an IP address or a host name that can be resolved by TCP/IP; for example, `myserver`. The default value is `localhost`.
JDBC Port	Value to identify the TCP/IP port.
SID	Unique system identifier of an Oracle database instance.
Service Name	The service name for an Oracle database instance.

Figure 8–3 Create Database Connection



Create Database Connection

Configure a new database connection and add it to the current application (Application1).

Create Connection In:	[✓ Application Resources] IDE Connections
Connection Name:	PortalDB
Connection Type:	Oracle (JDBC)
Username:	
Role:	
Password:	
Save Password	✓

Oracle (JDBC) Settings

Enter Custom JDBC URL	
Driver:	thin
Host Name:	localhost
JDBC Port:	1521
[✓ SID:]	XE
Service Name:	XE

e. Click **OK**.

7. In the Create Content Repository Connection dialog, in the **Login Timeout (ms)** field, specify the time in milliseconds. This timeout determines how long the application must wait when trying to establish a session with the content repository. If the network connection or the content repository server to which you are trying to connect is slow, then consider overriding the default value specified in this field.

8. Select the applicable authentication method. See Section 8.2.5.1, "What You Should Know About Using Identity Propagation and External Application Authentication Methods" for information.

9. Select the **Specify login credentials for current JDeveloper session** checkbox and enter the Oracle Portal single sign-on user name and password for logging in to your portal instance. The Oracle Portal single sign-on user name and password entered here are used to test the connection and are functional only at design time. Then click **OK**.

Note: If the **Specify login credentials for current JDeveloper session** checkbox is selected, then the credentials you entered are used. If the checkbox is not selected, then the connection is tested using External Application credentials (if they exist), otherwise null credentials are used.

If you have selected **External Application** for authentication and also specified either public or shared credentials, then you can leave these fields blank, as the public or shared credentials can be used to login at design time. If you specified both public and shared credentials for the external application, then the public credentials have higher precedence. However, if you have selected **External Application** but have not specified public or shared credentials, then you must specify user name and password here.

---

10. In the Application Resources panel, expand the repository connection you just created. The Connection Credentials dialog displays.

Note: The Connection Credentials dialog displays if you do not save the credentials, that is, you do not select the **Specify login credentials for the current JDeveloper session** checkbox, or if you do not specify an External Application service that uses public or shared credentials.

---

11. Enter the user name and password for Oracle Portal connection and click **OK**. The connection displays under the Application Resources panel.

See Also: To create a data control using this connection, perform the procedure described in Section 8.4.1, "How to Configure a Content Repository Data Control".

8.2.3 How to Create a Content Repository Connection Based on the File System Adapter

This section describes the procedure to create a content repository connection based on the file system adapter.

To create a File System repository connection:

1. In Oracle JDeveloper, open your WebCenter application.
2. In the Application Resources panel, right-click **Connections** and choose **New Connection** and then **Content Repository**.
3. In the Create Content Repository Connection dialog, enter a name for the connection, for example, `MyConnection`.

   Tip: You can choose between creating application-specific connections in the Application Resources panel and creating common connections in the Resource Palette (IDE Connections).

4. Select **File System** from the **Repository Type** box.
5. Select **Set as primary connection for Documents service**, if you intend to make it the default connection for the Documents service. Selecting this option writes a configuration to the `adf-config.xml` file that identifies this connection as the

default connection for the Documents service. If a document library task flow is used without any `connectionName` input parameter, then this connection is used. For information about the Documents service, see Chapter 14, "Integrating the Documents Service".

Note: To be able to set a connection as primary connection for the Documents service, you must select the **Application Resources** option in **Create Connection In**. This is because connections created as IDE connections cannot be used directly, they must be added to applications.

---

6. Under **Configuration Parameters**, select the **Base Path** row, and enter the path to the folder in which your content is placed, for example, `C:\MyContent`.

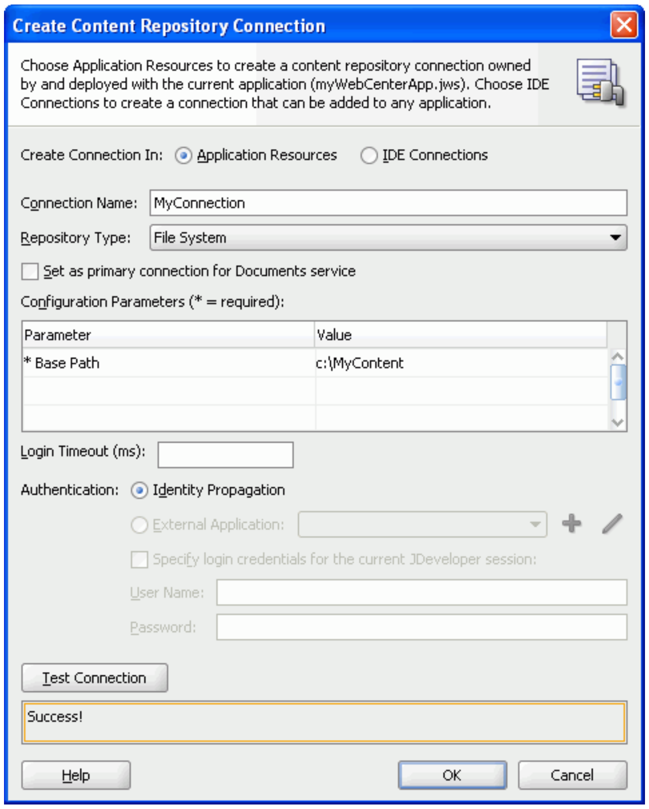
Note: The following must be considered while creating a file system connection:

- The base path value is used as the root (`"/"`) for the file system-based data control in Section 8.5, "Integrating Content Using Content Data Controls: Examples".
- When you work on a UNIX system, the File System adapter inherits the case-sensitive file name characteristic of UNIX systems. So, on UNIX systems, you must ensure that references to files follow the same case as that used in the original file names. For example, suppose the `Test.html` file was created on a Microsoft Windows system. When you reference this file on a Linux system, you must ensure that you use `Test.html`, and not `test.html` or `TEST.html`.

---

7. Leave the **Login Timeout (ms)** field blank.
8. Click **Test Connection** to check whether you have entered the connection details correctly. You should see a `Success!` status, as shown in Figure 8–4.

Figure 8–4 File System Connection



Create Content Repository Connection

Choose Application Resources to create a content repository connection owned by and deployed with the current application (myWebCenterApp.jws). Choose IDE Connections to create a connection that can be added to any application.

Create Connection In: [✓ Application Resources] IDE Connections

Connection Name: MyConnection

Repository Type: File System

Set as primary connection for Documents service

Configuration Parameters (* = required):

Parameter	Value
* Base Path	c:\MyContent

Login Timeout (ms):

Authentication: [✓ Identity Propagation] External Application:

Specify login credentials for the current JDeveloper session:

User Name:

Password:

Test Connection

Success!

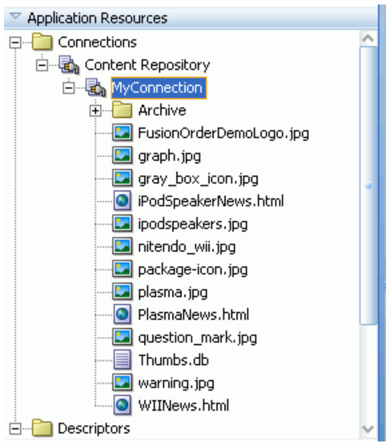
9. Click **OK**.

10. In the Application Resources panel, expand the repository connection you just created.

8.2.4 What Happens When You Create a Repository Connection

When you create a connection to a repository, the contents in the main directory of the repository display under the Content Repository connection in the Application Resources panel, as shown in Figure 8–5. You can double-click folders and files to view them.

Figure 8–5 Application Resources



```
Application Resources
  Connections
    Content Repository
      MyConnection
        Archive
        FusionOrderDemoLogo.jpg
        graph.jpg
        gray_box_icon.jpg
        iPodSpeakerNews.html
        ipodspeakers.jpg
        nitendo_wii.jpg
        package-icon.jpg
        plasma.jpg
        PlasmaNews.html
        question_mark.jpg
        Thumbs.db
        warning.jpg
        WIINews.html
  Descriptors
```

You can use repository connections to create JCR data controls that enable integration of the repository content with JSF pages. See Section 8.4, "Configuring Content Data Controls for JCR Adapters" and Section 8.5, "Integrating Content Using Content Data Controls: Examples" for information. These connections can also be consumed through the Documents service task flows, see Chapter 14, "Integrating the Documents Service" for information.

8.2.5 What You May Need to Know When Creating a Repository Connection

This section includes:

- Section 8.2.5.1, "What You Should Know About Using Identity Propagation and External Application Authentication Methods"
- Section 8.2.5.2, "What You Should Know About Oracle Portal"

8.2.5.1 What You Should Know About Using Identity Propagation and External Application Authentication Methods

The Create Content Repository Connection dialog provides the following options for authentication methods:

- **Identity Propagation**: If you select this option, no credentials are passed to the repository. The repository connector instead uses the current user's identity as determined from the Java security context. This must only be used when the application and the repository use the same identity store to authenticate users.

  To apply this authentication method, it is best to configure security for your application using the Configure ADF Security wizard since repositories may support only authenticated users or provide only limited access to the public or guest user. See Section 8.4.4, "Securing a Content Repository Data Control".

- **External Application**: The External Application method can be used in all other cases where the current user identity should not be propagated directly to the repository. For more information, see Section 24.2, "Working with External Applications".

  The External Application service allows different types of credentials to be associated with a connection:

  – Public credentials are used whenever an application is not secured, or the user has not yet logged in.

  – Shared credentials are used for any authenticated user.

  – If shared credentials are not specified, then mapped credentials can be used to obtain a result similar to identity propagation when the application and content repository do not use the same user store. In that case the external application service provides runtime screens through which users can provide their credentials for accessing the content repository. Those credentials are securely stored and used for any future connection that this user tries to establish.

  If you intend to configure the External Application service, then click **New** to launch the Register External Application wizard and do the following:

  a. Specify name of the external application.

  b. In the **Login URL** field, enter the URL of the external application, for example, `http://content-server.mycompany.com/idc/`

Note: This URL is optional and is required only to provide *click through login* to the content repository's own user interface. See the External Application service documentation for more information on how to configure click through login.

---

c. Then, under **Authentication Details**, select **Basic** from the dropdown list. Leave the fields under **Login Details** blank.

   This option is selected when *click through login* is not required. If *click through login* is enabled, then you must select another option based on the authentication method used in the repository.

d. Accept the default values and click **Next**.

e. Enter shared credentials that can be used at design time and runtime. Do not specify shared credentials, if users must specify their own credentials.

f. Enter public credentials that can be used at design time and runtime. Specify public credentials only if you intend public users to view documents or contents.

   Click **Finish**.

8.2.5.2 What You Should Know About Oracle Portal

The following should be considered while using the Oracle Portal adapter:

- To use Oracle Portal-based content data control capabilities, you can install Oracle Portal release 10.1.4.1, 10.1.4.2, or 11. If you use the Oracle Portal release 10.1.4.1, then you must update it with the required patches. Consult Oracle Application Server Release Notes for Microsoft Windows (for the required platform) for release 11.1.1.0.0 to know the exact patch number.
- Only `file`, `image`, `imagemap`, and `text` item types and custom types based on these item types are supported.
- The `portal:container` page type and its extensions are supported.
- Content is always exposed in the default language of the page group. For example, if there are three page groups with different default languages, then the content displays only for those default languages and not for any translations that exist.

8.2.6 How to Edit a Common Repository Connection

Common repository connections (IDE connections) are created and edited under the Resource Palette.

To edit a common repository connection:

1. Under the Resource Palette, right-click the repository connection you intend to edit and choose **Properties**, as shown in Figure 8–6. The Edit Repository Connection dialog displays, as shown in Figure 8–7.

Figure 8–6 Resource Palette - Properties

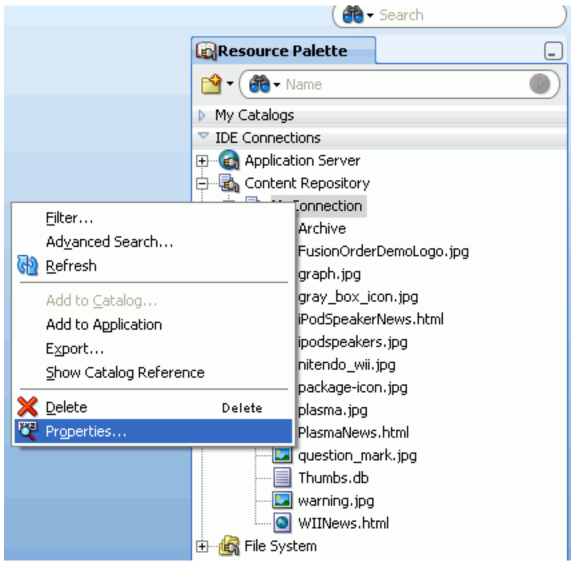
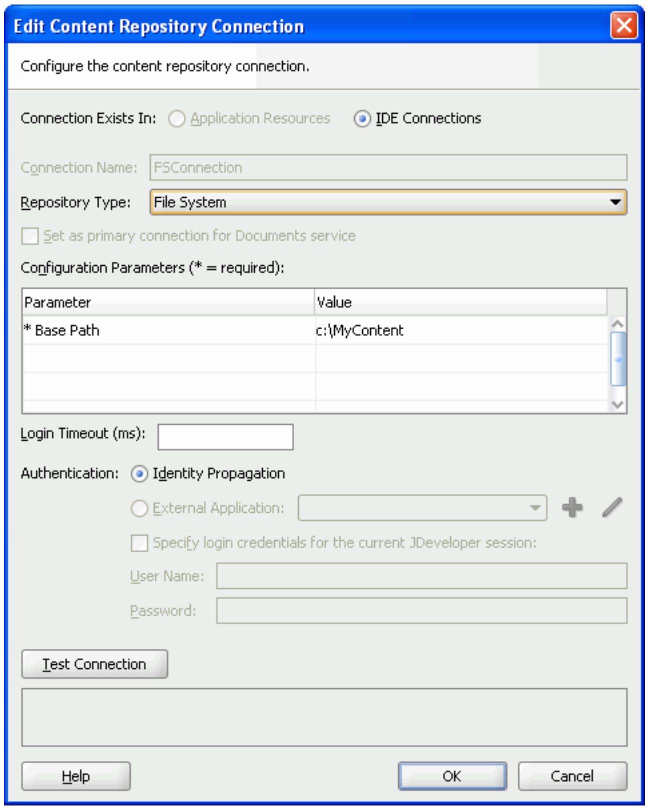


Figure 8–7 Edit Content Repository Connection


Edit Content Repository Connection

Configure the content repository connection.

Connection Exists In: ( ) Application Resources [✓ IDE Connections]

Connection Name: FSConnection

Repository Type: File System

[ ] Set as primary connection for Documents service

Configuration Parameters (* = required):

Parameter	Value
* Base Path	c:\MyContent

Login Timeout (ms):

Authentication: [✓ Identity Propagation]

( ) External Application:

[ ] Specify login credentials for the current JDeveloper session:

User Name:

Password:

2. Change the appropriate parameters.
3. Test the connection and click **OK**.

8.2.7 How to Edit a WebCenter Application-Specific Content Repository Connection

Application-specific content repository connections exist under the Application Resources panel.

To edit a WebCenter application-specific content repository connection:

1. In the Application Resources panel, right-click the connection you intend to edit, and choose **Properties**, as shown in Figure 8–8. The Edit Content Repository Connection dialog displays, as shown in Figure 8–9.

Figure 8–8 Application Resources - Properties

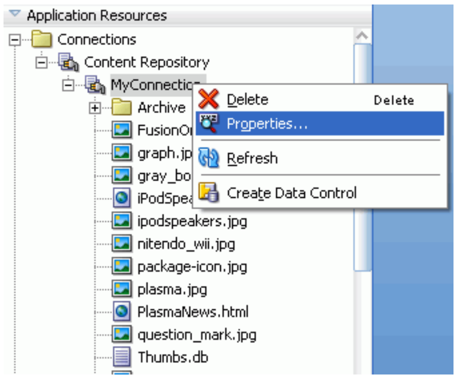
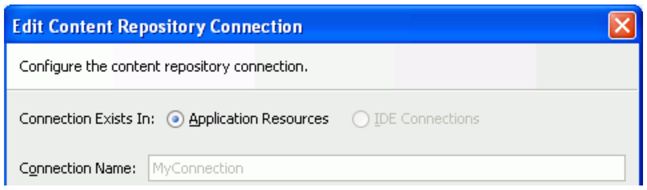


Figure 8–9 Edit Content Repository Connection



Configure the content repository connection.

Connection Exists In: (●) Application Resources ( ) IDE Connections

Connection Name: MyConnection

2. Change the appropriate parameters. Depending on the repository type, see the earlier sections that describe how to create repository connections. For example, for Oracle Content Server, see Section 8.2.1, "How to Create a Content Repository Connection Based on the Oracle Content Server Adapter".
3. Test the connection and click **OK**.

8.2.8 How to Use an Existing Repository Connection for a New WebCenter Application

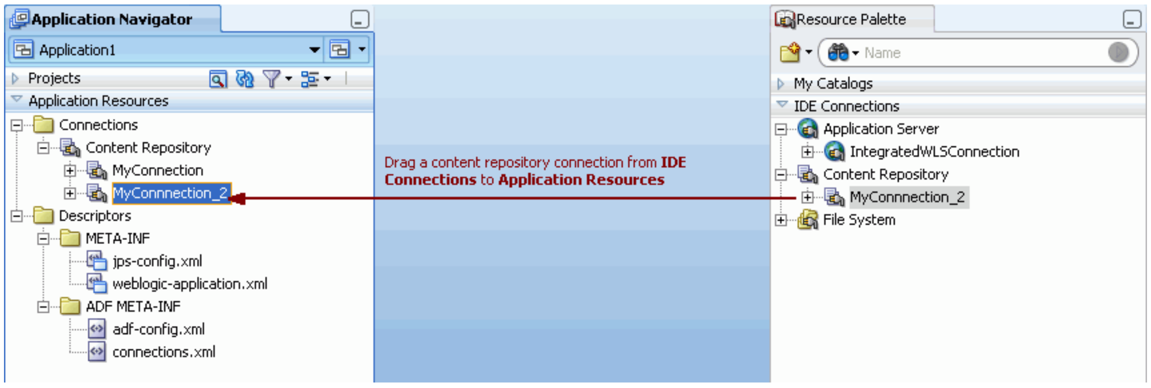
You can use an existing repository connection for any WebCenter application, if you created it as a common repository under the Resource Palette.

To use an existing repository connection:

1. In Oracle JDeveloper, open the WebCenter application for which you intend to use an existing repository connection.
2. Go to the Resource Palette and select the repository connection that you intend to use for a new application, for example `MyConnection_2`, and drop it under the Application Resources panel of the new application, as shown in Figure 8–10.

> **Tip:** Alternatively, right-click the connection and choose **Add to Application**. The connection is displayed under the Application Resources panel.

Figure 8–10 Dragging an Existing Common Repository Connection (IDE Connection) to an Application



You can now configure a data control for your new application from this repository connection, as described in Section 8.4.1, "How to Configure a Content Repository Data Control".

8.3 Adding Content Using Content Repository Connections

Contents of a content repository connection display in the Application Resources, under **Connections**, **Content Repository**. From here you can add:

- Images stored in your content repository as **ADF Image**.
- Images, PDF, and HTML files as **ADF Inline Frame**.
- Files and folders as **ADF Go Link**.

This section includes the following:

- Section 8.3.1, "How to Add Content as an ADF Inline Frame"
- Section 8.3.2, "How to Add Content as an ADF Image"
- Section 8.3.3, "How to Add Content as an ADF Go Link"

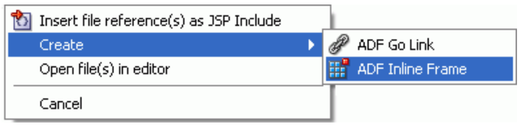
8.3.1 How to Add Content as an ADF Inline Frame

In a JSF page, you can add HTML, PDF, and image files (PNG, JPEG, and GIF) as ADF Inline Frames.

To add a file as an ADF Inline Frame:

1. In the Application Resources, expand **Content Repository** and the content repository connection.
2. Select a file and drop it onto the JSF page. From the **Create** menu, choose **ADF Inline Frame**, as shown in Figure 8–11.

Figure 8–11 Create Menu - ADF Inline Frame Option



The Source view should look like this:

```
<af:inlineFrame id="inlineFrame1"
 source="/content/conn/connection name/path/directory_name/file name"
 inlineStyle="width:100%;height:100%"/>
```

3. Run the JSF page to view the output in a browser window.

8.3.2 How to Add Content as an ADF Image

In a JSF page, you can add image files such as PNG, JPEG, and GIF, as ADF Images.

To add an image file as an ADF Image:

1. In the Application Resources, expand **Content Repository** and the content repository connection.
2. Select an image file and drop it onto the JSF page. From the Create menu, choose **ADF Image**. The Source view should look like this:

```
<af:image id="image1"
 source="/content/conn/connection name/path/directory_name/file name"/>
```

3. Run the JSF page to see the image in a browser window.

8.3.3 How to Add Content as an ADF Go Link

In a JSF page, you can add files and folders as **ADF Go Links**.

To add a file as an ADF Go Link:

1. In the Application Resources, expand **Content Repository** and the content repository connection.
2. Select a file and drop it onto the JSF page. From the Create menu, choose **ADF Go Link**. The Source view should look like this:

```
<af:goLink id="goLink1" text="file name"
 destination="/content/conn/connection name/path/directory_name/file name"/>
```

3. Run the JSF page to see the link in a browser window.

8.4 Configuring Content Data Controls for JCR Adapters

This section describes how to create a content data control based on a content repository connection. This section includes the following:

- Section 8.4.1, "How to Configure a Content Repository Data Control"
- Section 8.4.2, "What Happens When You Configure a Content Repository Data Control"
- Section 8.4.3, "How to Edit a Content Repository Data Control"

8.4.1 How to Configure a Content Repository Data Control

The procedure to create content repository data controls is the same irrespective of the types of the connections that you use to create data controls. If you have not created a connection to your content repository, then see Section 8.2, "Configuring Content Repository Connections". In this section, a content data control is created using an Oracle Content Server-based connection.

To create a content data control using an existing content repository connection:

1. Under the Application Navigator, select the **Model** project, in which the project entries for the new data control should be created.

   You must do this to ensure that the data control definition is separate from the data control usage. When you select the **Model** project, data control definition files are created in its sub folder, **Application Sources** and not in the **ViewController** project in which the user interface is created.

2. Under the Application Resources panel, expand the **Connections** folder in which you created the repository connection.

3. Drag the repository connection that you intend to use for the new data control and drop it under the Data Controls panel.

   Alternatively, from the File menu, choose **New**. In the New Gallery, expand **Business Tier**, select **Content Repository** and then **Content Repository Data Control**, and click **OK**.

   The Create Content Repository Data Control dialog displays with the name of the connection selected in the **Connection Name** dropdown list.

   > **Note:** If you created your repository connection under the Resource Palette, then right-click the connection under the Resource Palette and choose **Create Data Control**. From the Resource Palette, you can also drag and drop the required connection onto the Data Controls panel.
   >
   > To add a new connection, click the **Create new content repository connection** icon to display the Create Content Repository Connection dialog. For more information, see Section 8.2, "Configuring Content Repository Connections".

4. In the **Data Control Name** field, enter a name for your data control, for example `MyDataControl`.

5. To add custom attributes, click **Add**. Then, enter a name for the attribute as it should appear in the Data Controls panel, select its type, and enter the JCR path.

   In Oracle Content Server, a metadata attribute named `dPropertyName` is mapped in the adapter as shown here:

   ```
   jcr:content/idc:metadata/idc:dPropertyName
   ```

   For example, if the property name is `dWorkflowState`, then it is mapped as `jcr:content/idc:metadata/idc:dWorkflowState`.

   In Oracle Portal, a metadata attribute is mapped in the adapter as shown here:

   ```
   portal:name of the attribute in Oracle Portal
   ```

   > **Note:** To retrieve the JCR paths of item attributes, run the `getAttributes` method on the required items. Then reenter the wizard to include those paths by right-clicking the respective data control in the Data Controls panel and choosing **Edit Definition**.

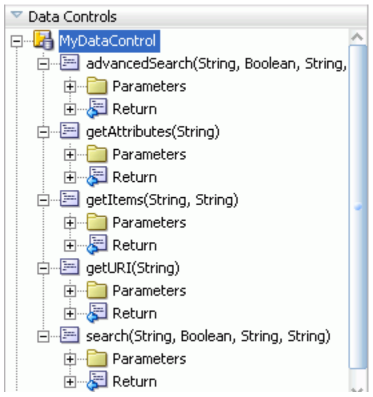
6. Click **OK**.

Note: The data controls include the predefined node types of JCR. These basic node types represent folders (`nt:folder`) and files (`nt:file`). These are derived from `nt:hierarchyNode`, which is a child of the `nt:base` supertype. Additional node types can be derived from the basic types to customize the experience to a particular repository.

8.4.2 What Happens When You Configure a Content Repository Data Control

When the data control is successfully configured, it shows under the Data Controls panel. Expand it in the Data Controls panel to see the hierarchical list of methods, parameters, and operations for the new data control, as shown in Figure 8–12. For detailed information on using the Data Control panel, see Chapter titled "Using Oracle ADF Model in A Fusion Web Application" of *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

Figure 8–12 Data Controls Panel - Oracle Content Server



```
Data Controls
MyDataControl
  advancedSearch(String, Boolean, String,
    Parameters
    Return
  getAttributes(String)
    Parameters
    Return
  getItems(String, String)
    Parameters
    Return
  getURI(String)
    Parameters
    Return
  search(String, Boolean, String, String)
    Parameters
    Return
```

Table 8–11 describes the icons and object hierarchy of a data control.

Table 8–11 The Data Controls Panel Icons and Object Hierarchy

Icon	Name	Description	Used to Create...
	Data Control	Represents a data control. You cannot use the data control itself to create UI components, but you can use any of the child objects listed under it. Depending on how your business services were defined, there may be multiple data controls.	Serves as a container for the other objects, and is not used to create anything
	Method	Represents an operation in the data control or one of its exposed structures that may accept parameters, perform some business logic and optionally return single value, a structure or a collection of those	Command components For methods that accept parameters: command components and parameterized forms

Table 8–11 (Cont.) The Data Controls Panel Icons and Object Hierarchy

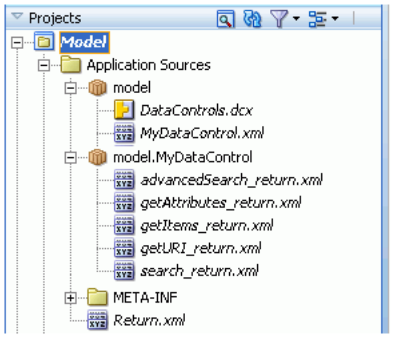
Icon	Name	Description	Used to Create...
	Method Return	Represents an object that is returned by a custom method. The returned object can be a single value or a collection. A method return appears as a child under the method that returns it. The objects that appear as children under a method return can be attributes of the collection, other methods that perform actions related to the parent collection, and operations that can be performed on the parent collection.	For single values: text fields and selection lists For collections: forms, tables, trees, and range navigation components When a single-value method return is dropped, the method is not invoked automatically by the framework. A user either has to also create an invoke action as an excecutable, or drop the corresponding method as a button to invoke the method.
	Attribute	Represents a discrete data element in an object (for example, an attribute in a row). Attributes appear as children under the collections or method returns to which they belong. Only the attributes that were included in the view object are shown under a collection. If a view object joins one or more entity objects, that view object's collection will contain selected attributes from all of the underlying entity objects.	Label, text field, date and selection list components
	Operation	Represents a built-in data control operation that performs actions on the parent object. Data control operations are located in an Operations folder under collections or method returns, and also under the root data control node. The operations that are children of a particular collection or method return operate on those objects only, while operations under the data control node operate on all the objects in the data control. If an operation requires one or more parameters, they are listed in a Parameters folder under the operation.	UI components such as buttons or links
	Parameter	Represents a parameter value that is declared by the method or operation under which it appears. Parameters appear in the Parameters folder under a method or operation.	Label, text, and selection list components

The following files (Figure 8–13) are created under the **Model** project:

- `DataControls.dcx`: Oracle JDeveloper creates this file the first time a data control is created. This file lists all the Oracle ADF data controls created under the current project. This file is required to initialize the data control.
- *`datacontrolname`*`.xml`: This file includes attributes, accessors, and operations of a data control.
- `advancedSearch_return.xml`: This file includes the return type definition for the `advancedSearch` method.
- `getAttributes_return.xml`: This file includes the return type definition for the `getAttributes` method.
- `getItems_return.xml`: This file includes the return type definition for the `getItems` method.
- `getURI_return.xml`: This file includes the return type definition for the `getURI` method.

- `search_return.xml`: This file includes the return type definition for the `search` method.
- `Return.xml`: This file defines the standard operations on collections.

Figure 8–13 Projects Panel - Model Project



8.4.3 How to Edit a Content Repository Data Control

This section describes a generic procedure to edit content data controls that you configured as described in Section 8.4, "Configuring Content Data Controls for JCR Adapters".

To edit a content data control:

1. In Oracle JDeveloper, go to the application that contains your content data control.
2. Under Data Controls panel, right-click the data control you intend to edit, and choose **Edit Definition**, as shown in Figure 8–14. The Edit Content Repository Data Control dialog displays, as shown in Figure 8–15.

Figure 8–14 Edit Data Controls

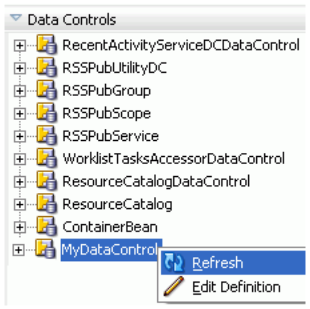
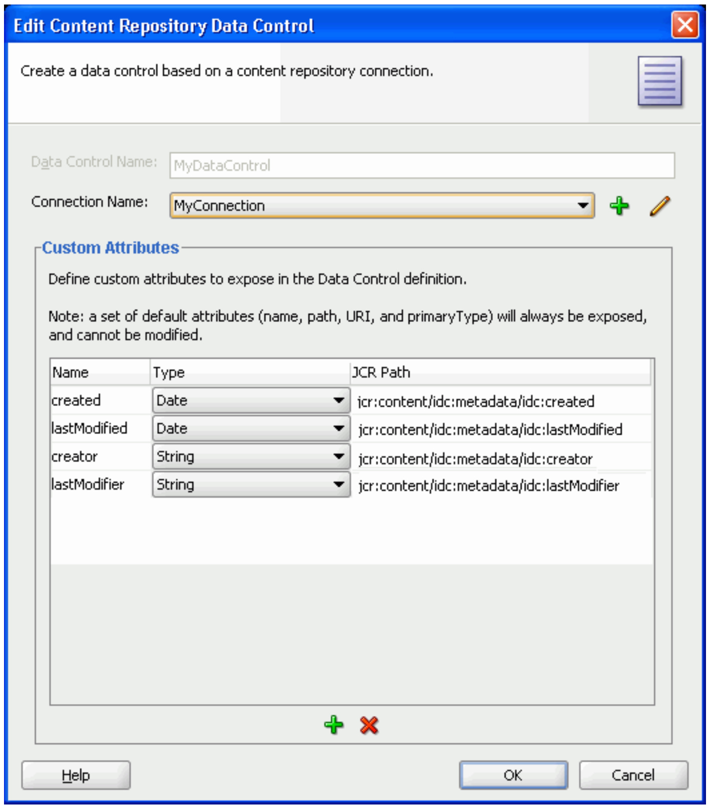


Figure 8–15 Edit Content Repository Data Control Dialog



Create a data control based on a content repository connection.

Data Control Name: MyDataControl

Connection Name: MyConnection

Custom Attributes

Define custom attributes to expose in the Data Control definition.

Note: a set of default attributes (name, path, URI, and primaryType) will always be exposed, and cannot be modified.

Name	Type	JCR Path
created	Date	jcr:content/idc:metadata/idc:created
lastModified	Date	jcr:content/idc:metadata/idc:lastModified
creator	String	jcr:content/idc:metadata/idc:creator
lastModifier	String	jcr:content/idc:metadata/idc:lastModifier

3. To select a different connection, use the **Connection Name** dropdown list.
4. To add a connection, click the **Create new content repository connection** icon to display the Create Content Repository Connection dialog. Depending on the type of connection that you intend to create, see relevant sections in Section 8.2, "Configuring Content Repository Connections" for information about its configuration parameters.
5. To edit an existing connection, click the **Edit selected content repository connection** icon to display the Edit Content Repository Connection dialog. Depending on the type of connection that you intend to modify, see relevant sections in Section 8.2, "Configuring Content Repository Connections" for information about its configuration parameters.
6. To add or remove custom attributes, use the **Add new attribute** icon and the **Remove selected attribute** icon respectively in the Custom Attributes box.
7. Click **OK** to apply the changes.

8.4.4 Securing a Content Repository Data Control

You can enable security for your content repository connections. For information, see Chapter 24, "Securing Your WebCenter Application".

8.5 Integrating Content Using Content Data Controls: Examples

In this section, you will use `getURI`, `getItems`, `search`, and `advancedSearch` methods of your data control. The basic procedure to use any data control method is to drag and drop it onto the JSPX page. When a method is dropped, its ADF Faces tag is added to the source of the JSPX page, and the tag is displayed in the Structure window. For example, dropping the URI attribute of `getURI` as a Go Link adds an `af:goLink` to the JSPX page. The destination of the `af:goLink` is set to the EL expression `#{bindings.URI.inputValue}`. This EL expression ties the destination value of the `af:goLink` to the binding container's URI value.

In the page definition file, `methodIterator` is added to the `executables` element, and `methodAction` is added to the `bindings` element. The `methodIterator` and `methodAction` elements define which data control and method is to be used. The `NamedData` attribute of the `methodAction` defines what input parameter values should be provided to this method call. The path and type parameters are common across all data control methods. However, `NamedData` for search and `advancedSearch` includes additional parameters that are unique to those methods. For example, in the `advancedSearch` method, the `NamedData` can also be used to define the search predicates, whether the search is recursive, and so on.

The page definition includes the `AttrNames` element, which defines the attributes of the items available in the bindings container to ADF. These items are based on default attributes and custom attributes that were defined when creating the data control.

For detailed information on using the Data Control panel and working with page definition files, see the chapter "Using Oracle ADF Model in A Fusion Web Application" of *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

The examples in this section use the following data control methods:

- `getURI` to add textual and clickable image links to the repository folder.
- `getItems` to publish the repository content in an ADF table and tree.
- `search` and `advancedSearch` to display those items of the repository that match the search criteria.

This section includes the following:

- Section 8.5.1, "How to Publish Content As Links"
- Section 8.5.2, "What Happens at Runtime"
- Section 8.5.3, "How to Publish Content in a Table"
- Section 8.5.4, "What Happens at Runtime"
- Section 8.5.5, "How to Publish Folder Content in a Tree"
- Section 8.5.6, "What Happens at Runtime"
- Section 8.5.7, "How to Add Search Capabilities to Content Repositories"
- Section 8.5.8, "What Happens at Runtime"
- Section 8.5.9, "What You May Need to Know When Using Search Capabilities"

8.5.1 How to Publish Content As Links

This section describes how to create hyperlinks to files stored in a file system and convert them into textual and image links. You use the **ADF Go Link** and the `getURI` method to create textual links and the Image of **ADF Faces** to create image links.

This section includes the following procedures:

- Section 8.5.1.1, "Publishing Content As a Textual Link"
- Section 8.5.1.2, "Creating a Clickable Image to Link to a Document"

Before you begin:

1. Configure a repository connection as described in Section 8.2, "Configuring Content Repository Connections".
2. Configure a data control as described in Section 8.4, "Configuring Content Data Controls for JCR Adapters".
3. Create a JSPX page as described in Section 3.3, "Creating WebCenter Application-Enabled Pages."

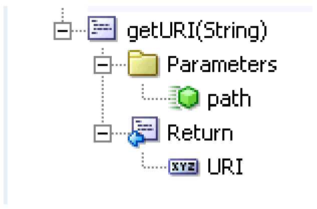
8.5.1.1 Publishing Content As a Textual Link

In this section, you will use the **Oracle ADF Go Link** option of the `getURI` method to publish your content as a textual link. You will also create a link to show a specific item of a repository.

To publish content as a textual link:

1. In the Application Navigator, double-click a `.jspx` page to open it in the visual editor.
2. In the Data Controls panel, under the repository connection, expand the **getURI (String)** method and expand **Return**. You should see the `URI` attribute, as shown in Figure 8–16.

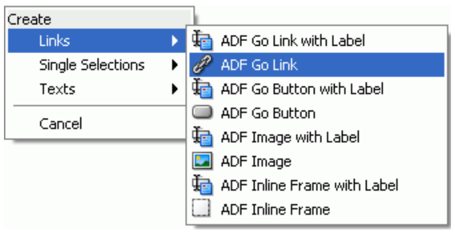
Figure 8–16 The URI Attribute of the getURI Method



3. To create a textual link, select the **URI** attribute and drop it on to the page, or in the Structure window under `af:form`. From the **Create** menu, choose **Links** and then **ADF Go Link**, as shown in Figure 8–17.

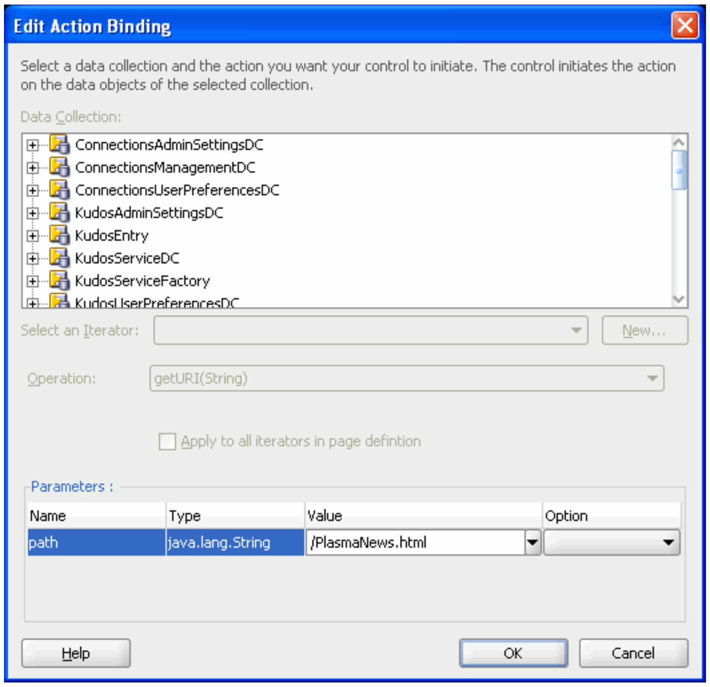
   If this is the first time you have dropped a node onto the page, then the Edit Action Binding dialog displays.

Figure 8–17 Oracle JDeveloper Context Menu for the getURI method



4. In the **Value** field of the **path** parameter, enter the path of the file for which you intend to create the link, as shown in Figure 8–18. You must enter a leading slash (`/`), for example `/PlasmaNews.html`. To modify or delete this path later, click the arrow icon next to the node in design mode and choose **Go to Binding** from the context menu.

Figure 8–18 Edit Action Binding



Edit Action Binding

Select a data collection and the action you want your control to initiate. The control initiates the action on the data objects of the selected collection.

Data Collection:

- ConnectionsAdminSettingsDC
- ConnectionsManagementDC
- ConnectionsUserPreferencesDC
- KudosAdminSettingsDC
- KudosEntry
- KudosServiceDC
- KudosServiceFactory
- KudosUserPreferencesDC

Select an Iterator:

Operation: getURI(String)

Apply to all iterators in page defintion

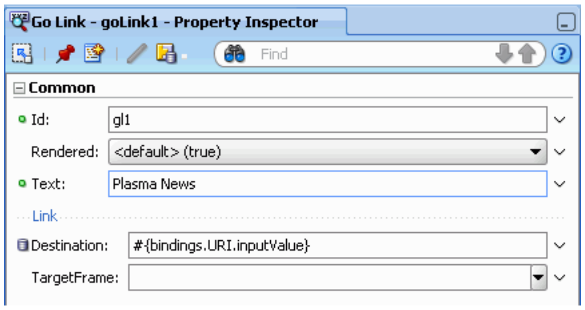
Parameters :

Name	Type	Value	Option
path	java.lang.String	/PlasmaNews.html	

Note: To grant edit, personalize, customize, and view permissions at the attribute level, see Section 8.4.4, "Securing a Content Repository Data Control".

5. Click **OK**.
6. Right-click the page and select **Run**. In your browser, you should see the URL for the file path entered in the Edit Action Binding dialog without any formatting.
7. By default, the link displays the text `goLink1`. In the Structure window, select **af:goLink - goLink1** and view the properties in the Property Inspector.
8. In the **Text** field, enter a name for the link, for example, `Plasma News`, as shown in Figure 8–19.

Figure 8–19 Go Link Properties



Go Link - goLink1 - Property Inspector

Common

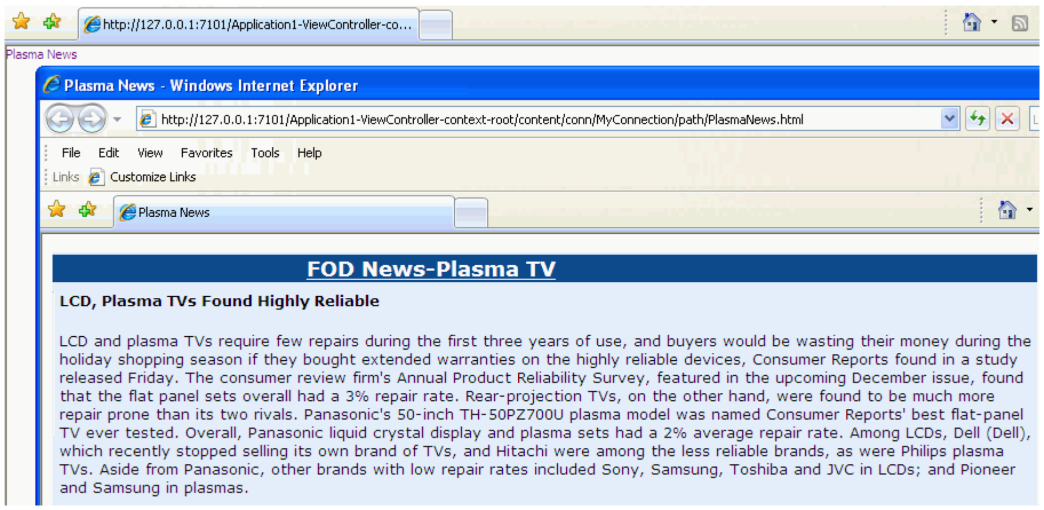
Id:	gl1
Rendered:	<default> (true)
Text:	Plasma News

Link

Destination:	#{bindings.URI.inputValue}
TargetFrame:	

9. Right-click your page and choose **Run**. The page appears in your browser window with the new link, as shown in Figure 8–20.

Figure 8–20 ADF Go Link in a Browser


http://127.0.0.1:7101/Application1-ViewController-co...
Plasma News
Plasma News - Windows Internet Explorer
http://127.0.0.1:7101/Application1-ViewController-context-root/content/conn/MyConnection/path/PlasmaNews.html
File Edit View Favorites Tools Help
Links Customize Links
Plasma News

FOD News-Plasma TV

LCD, Plasma TVs Found Highly Reliable

LCD and plasma TVs require few repairs during the first three years of use, and buyers would be wasting their money during the holiday shopping season if they bought extended warranties on the highly reliable devices, Consumer Reports found in a study released Friday. The consumer review firm's Annual Product Reliability Survey, featured in the upcoming December issue, found that the flat panel sets overall had a 3% repair rate. Rear-projection TVs, on the other hand, were found to be much more repair prone than its two rivals. Panasonic's 50-inch TH-50PZ700U plasma model was named Consumer Reports' best flat-panel TV ever tested. Overall, Panasonic liquid crystal display and plasma sets had a 2% average repair rate. Among LCDs, Dell (Dell), which recently stopped selling its own brand of TVs, and Hitachi were among the less reliable brands, as were Philips plasma TVs. Aside from Panasonic, other brands with low repair rates included Sony, Samsung, Toshiba and JVC in LCDs; and Pioneer and Samsung in plasmas.

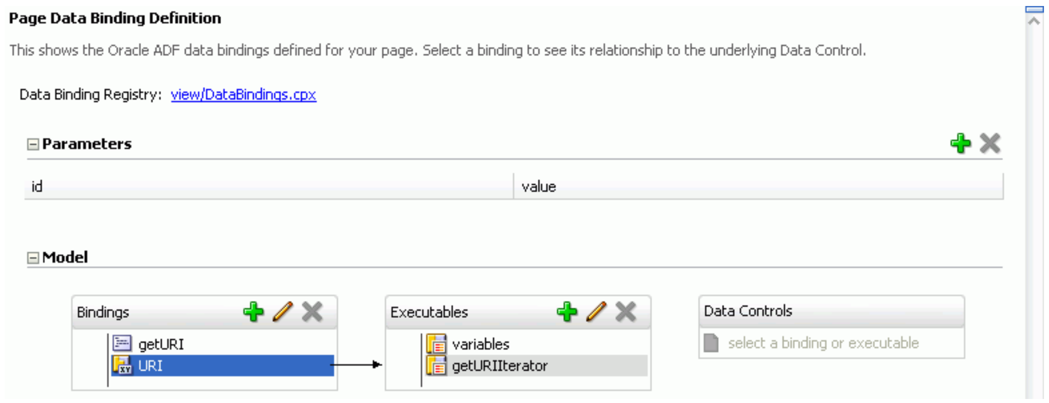
10. Click the link to check that the correct file is displayed.

In Section 8.5.1.2, "Creating a Clickable Image to Link to a Document", you will extend this textual link into an image link.

To add a link to another item, in the same JSF page:

1. In the Application Navigator, right-click the `.jspx` page, in which you created the ADF Go Link, and choose **Go to Page Definition**. The Page Data Binding Definition displays in the design view, as shown in Figure 8–21.

Figure 8–21 Page Data Binding Definition



Page Data Binding Definition

This shows the Oracle ADF data bindings defined for your page. Select a binding to see its relationship to the underlying Data Control.

Data Binding Registry: view/DataBindings.cpx

2. Go to the source view of the page definition.

3. In the `executables` element, add another `methodIterator` and change the `methodIterator id` and value of `Binds`, as shown in **Bold** in the following example:

```
<executables>
  <variableIterator id="variables"/>
  <methodIterator Binds="getURI.result" DataControl="MyDataControl"
     RangeSize="25" BeanClass="model.MyDataControl.getURI_return"
     id="getURIIterator"/>
  <methodIterator Binds="getURI1.result" DataControl="MyDataControl"
    RangeSize="25" BeanClass="model.MyDataControl.getURI_return"
    id="getURIIterator1"/>
  </executables>
```

4. In the `Bindings` element, add another `methodAction` and change the `methodAction id`, `ReturnName`, and `NDValue`, as shown in **Bold** in the following example:

```
<methodAction id="getURI"
  RequiresUpdateModel="true" Action="invokeMethod" MethodName="getURI"
  IsViewObjectMethod="false" DataControl="MyDataControl"
  InstanceName="MyDataControl"
  ReturnName="MyDataControl.methodResults.getURI_MyDataControl_getURI_result">
  <NamedData NDName="path" NDValue="/PlasmaNews.html"
    NDType="java.lang.String"/>
</methodAction>
<methodAction id="getURI1"
  RequiresUpdateModel="true" Action="invokeMethod MethodName="getURI"
  IsViewObjectMethod="false" DataControl="MyDataControl"
  InstanceName="MyDataControl"
  ReturnName="MyDataControl.methodResults.getURI_MyDataControl_getURI1_result">
  <NamedData NDName="path" NDValue="/FusionOrderDemoLogo.jpg"
    NDType="java.lang.String"/>
</methodAction>
```

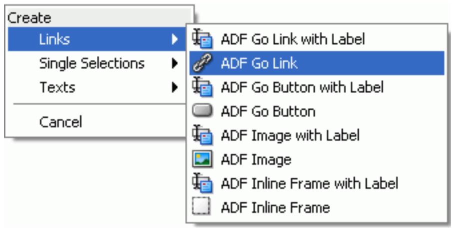
5. In the `Bindings` element, add another `attributeValues` tag to specify the new Id, as shown in **bold** in the following example:

```
<attributeValues
      IterBinding="getURIIterator"
      id="URI">
   <AttrNames>
      <Item Value="URI"/>
   </AttrNames>
</attributeValues>
<attributeValues
        IterBinding="getURIIterator1"
        id="URI1">
  <AttrNames>
        <Item Value="URI"/>
  </AttrNames>
</attributeValues>
```

6. In the Data Controls panel, under the repository connection, expand the **getURI (String)** method and expand **Return**. You should see the `URI` attribute.

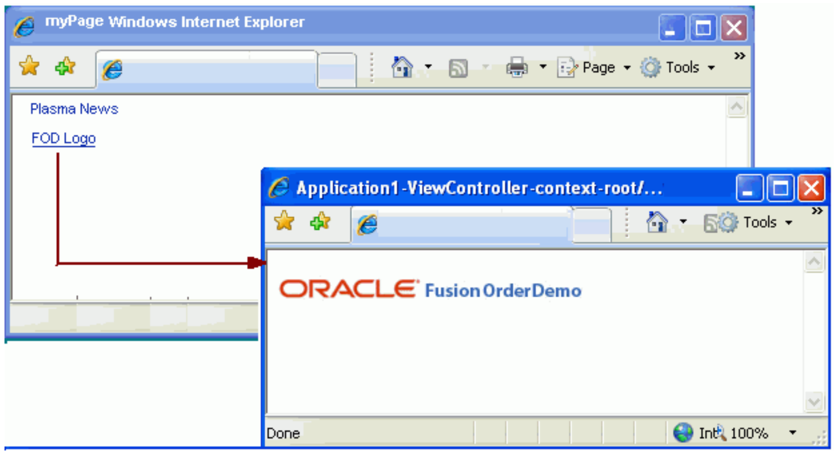
7. To create a textual link, select the **URI** attribute and drop it on to the page, or in the Structure window under `af:form`. From the **Create** menu, choose **Links** and then **ADF Go Link**, as shown in Figure 8–22.

Figure 8–22 Oracle JDeveloper Context Menu for the getURI method



8. In the Structure window, double-click the new goLink, for example, **af:goLink2** to display the Go Link Properties window.
9. In the **Text** field, enter a display name for your new link, for example, `FOD Logo`.
10. In the **Destination** field, click the down arrow to display the Expression Builder dialog. Then, expand **ADF Bindings**, **bindings**, and **URI1**.
11. Double-click **inputValue** variable to create the `#{bindings.URI1.inputValue}` expression, and click **OK**. This expression is based on new elements that you added in `executables` and `bindings`.
12. Run your page. The new link should display content from the new path (`NDValue`) that you specified in step 4. Figure 8–23 shows the new link, `FOD Logo`, in addition to the `Plasma News` link that was added in the first part of Section 8.5.1.1, "Publishing Content As a Textual Link".

Figure 8–23 New Textual Link


myPage Windows Internet Explorer
Page
Tools
Plasma News
FOD Logo
Application1-ViewController-context-root/...
Tools
ORACLE Fusion OrderDemo
Done
Int 100%

You can add as many links as required by following these steps.

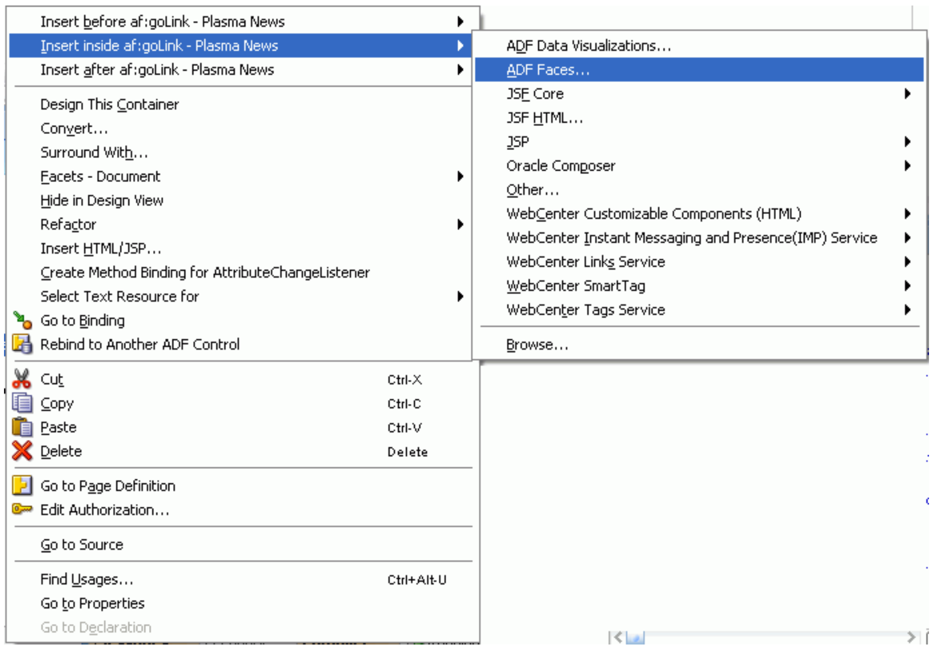
8.5.1.2 Creating a Clickable Image to Link to a Document

In this section, you will use the Image option of ADF Faces to publish a document as a clickable image, that is, clicking the image object will display your document.

To publish content as a clickable image object:

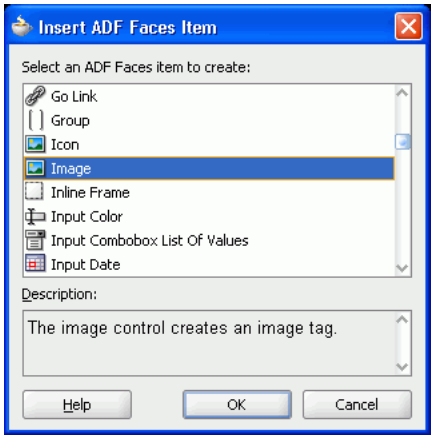
1. In the Application Navigator, open the `.jspx` page, in which you created the ADF Go Link, by double-clicking it.
2. To convert the textual link that you created in the first part of Section 8.5.1.1, "Publishing Content As a Textual Link", in the Structure window, right-click **af:goLink**, choose **Insert Inside af:goLink**, and then **ADF Faces**, as shown in Figure 8–24. The Insert ADF Faces Item dialog displays.

Figure 8–24 Insert Inside af:goLink - ADF Faces



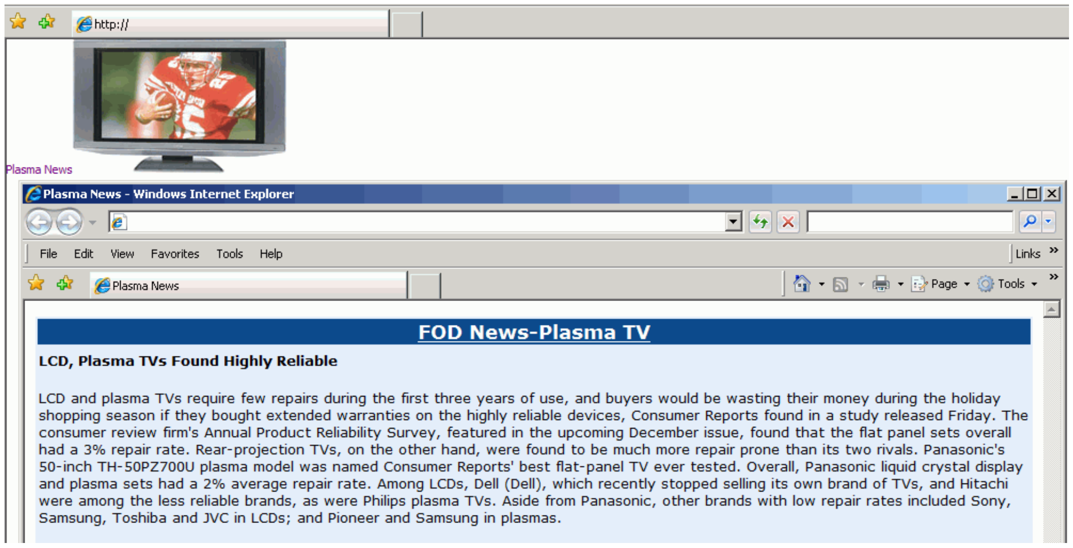
3. From the **Select an ADF Faces item to create** list, select **Image**, as shown in Figure 8–25.

Figure 8–25 Insert ADF Faces Item


Insert ADF Faces Item
Select an ADF Faces item to create:
Go Link
Group
Icon
Image
Inline Frame
Input Color
Input Combobox List Of Values
Input Date
Description:
The image control creates an image tag.
Help
OK
Cancel

4. Click **OK**. The Insert Image dialog displays.
5. Browse to the image file you intend to display as the image link, for example, `plasma.jpg`.
6. Click **Finish**.
7. Right-click your page and choose **Run**. Figure 8–26 shows a sample output.

Figure 8–26 ADF Object Image Link in a Browser


http://
Plasma News
Plasma News - Windows Internet Explorer
File Edit View Favorites Tools Help
Links
Plasma News
Page Tools

FOD News-Plasma TV

LCD, Plasma TVs Found Highly Reliable

LCD and plasma TVs require few repairs during the first three years of use, and buyers would be wasting their money during the holiday shopping season if they bought extended warranties on the highly reliable devices, Consumer Reports found in a study released Friday. The consumer review firm's Annual Product Reliability Survey, featured in the upcoming December issue, found that the flat panel sets overall had a 3% repair rate. Rear-projection TVs, on the other hand, were found to be much more repair prone than its two rivals. Panasonic's 50-inch TH-50PZ700U plasma model was named Consumer Reports' best flat-panel TV ever tested. Overall, Panasonic liquid crystal display and plasma sets had a 2% average repair rate. Among LCDs, Dell (Dell), which recently stopped selling its own brand of TVs, and Hitachi were among the less reliable brands, as were Philips plasma TVs. Aside from Panasonic, other brands with low repair rates included Sony, Samsung, Toshiba and JVC in LCDs; and Pioneer and Samsung in plasmas.

8.5.2 What Happens at Runtime

In the preceding examples, you created hyperlinks to files that are stored in a file system using **ADF Go Link** and converted them into textual and image links using the `getURI` method and the **Image** component of **ADF Faces**.

At runtime the ADF framework uses the information from the page definition file to invoke the data control methods required by the JSPX page. For information about the page lifecycle, see the chapter "Understanding the Fusion Page Lifecycle" of *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

At runtime, the `getURI` data control method is invoked to convert the given path into a valid HTTP URI for the content repository for which the data control is configured. The returned URL is syntactically correct for the target repository, but is not guaranteed to find a resource, that is, the `getURI` method does not validate that the path corresponds to an existing JCR node. When the page is rendered and the clickable image or link is clicked, the application's get handler converts the HTTP URL into a JCR path and attempts to retrieve the content for the JCR path. The get handler also supplies `mimeType` information in the response `Content-Type` header, if such information is available from the repository. This is because in JCR, `jcr:mimeType` is an optional property of the `nt:resource` node type.

8.5.3 How to Publish Content in a Table

In this section, the `getItems` data control method is used to publish file and folder information in a table. This section describes the following procedures:

- Section 8.5.3.1, "Displaying Files and Folders in Read-Only Format"
- Section 8.5.3.2, "Displaying the Name Attribute As a Go Link"
- Section 8.5.3.3, "Configuring a Table to Show Only Files"

Before you begin:

1. Configure a repository connection as described in Section 8.2, "Configuring Content Repository Connections".

2. Configure a data control as described in Section 8.4, "Configuring Content Data Controls for JCR Adapters".
3. Create a JSPX page as described in Section 3.3, "Creating WebCenter Application-Enabled Pages."

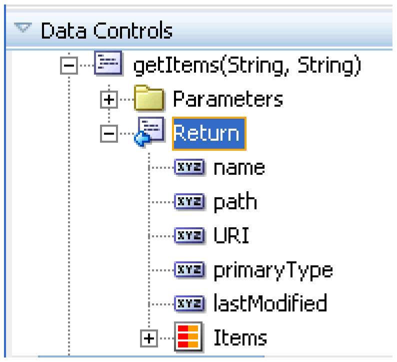
8.5.3.1 Displaying Files and Folders in Read-Only Format

Here you will create a read-only ADF table using the `getItems` method.

To display folder content in a read-only table:

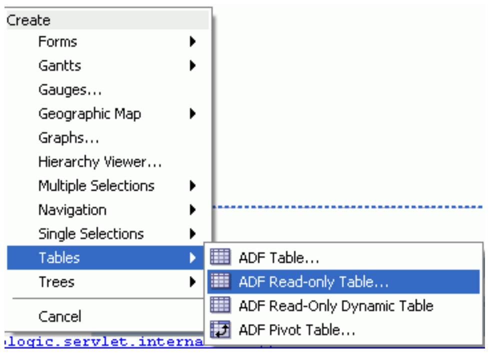
1. To open the page in the visual editor, double-click it in the Application Navigator.
2. In the Data Controls panel, under your data control, expand the **getItems** method and the **Return** node, as shown in Figure 8–27.

Figure 8–27 The Return node of the getItems method



3. To create a table that lists every file and folder available through this data control, drop the **Return** node on to the page or under `af:form` in the Structure window. From the **Create** menu, choose **Tables** and then **ADF Read-only Table** , as shown in Figure 8–28.

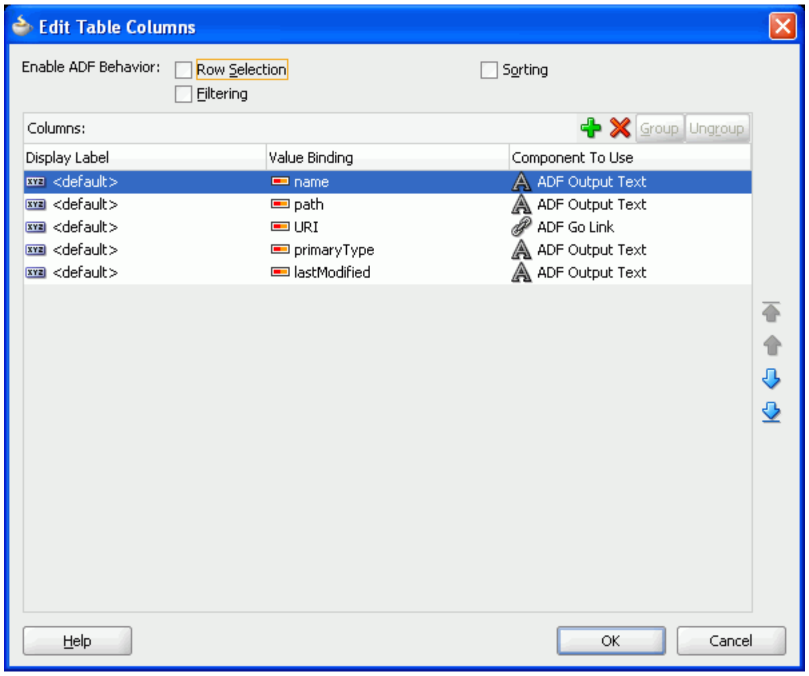
Figure 8–28 The Create Context Menu for getItems Method



4. In the Edit Table Columns dialog, as shown in Figure 8–29, Select the row for the getItems.name **Value Binding** and enter an appropriate value for the **Display Label**, for example, `Name`.

   Repeat this step for the **path**, **URI**, **primaryType**, and **lastModified** attributes. Enter new display labels such as `name`, `Location`, `URL`, and so on, then click **OK.**

Figure 8–29 Edit Table Columns



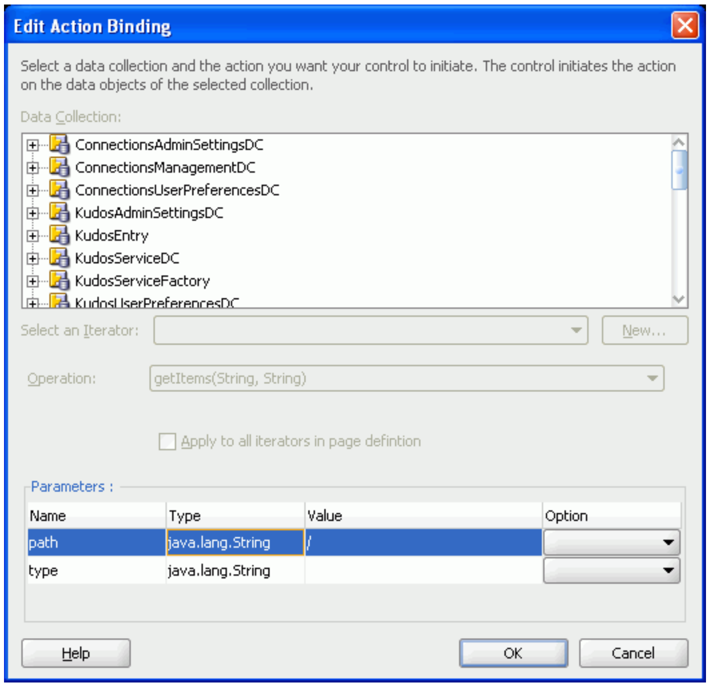
Display Label	Value Binding	Component To Use
<default>	name	ADF Output Text
<default>	path	ADF Output Text
<default>	URI	ADF Go Link
<default>	primaryType	ADF Output Text
<default>	lastModified	ADF Output Text

5. If this is the first time you have dropped a node onto the page, the Edit Action Binding dialog displays.

In the Edit Action Binding dialog, enter the path of the content directory as the `path` parameter, as shown in Figure 8–30. You must enter a leading slash (`/`). To modify or delete this path later, click the arrow icon next to the node in design mode and choose **Go to Binding** from the context menu.

Leave the `type` parameter blank. This implies that the table displays both files and folders.

Figure 8–30 Edit Action Binding



Edit Action Binding

Select a data collection and the action you want your control to initiate. The control initiates the action on the data objects of the selected collection.

Data Collection:
- ConnectionsAdminSettingsDC
- ConnectionsManagementDC
- ConnectionsUserPreferencesDC
- KudosAdminSettingsDC
- KudosEntry
- KudosServiceDC
- KudosServiceFactory
- KudosUserPreferencesDC

Select an Iterator:

Operation: getItems(String, String)

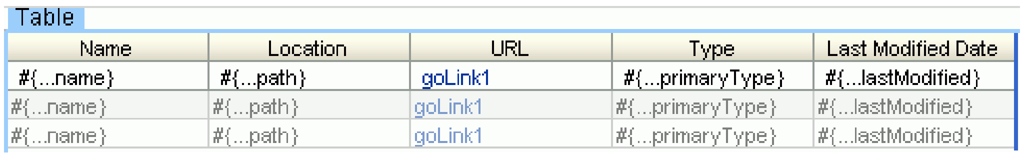
Apply to all iterators in page defintion

Parameters :

Name	Type	Value	Option
path	java.lang.String	/	
type	java.lang.String		

6. Click **OK.** You should now see a table on the JSPX page that looks like Figure 8–31.

Figure 8–31 Read-Only Table for Publishing Folder Content



Table

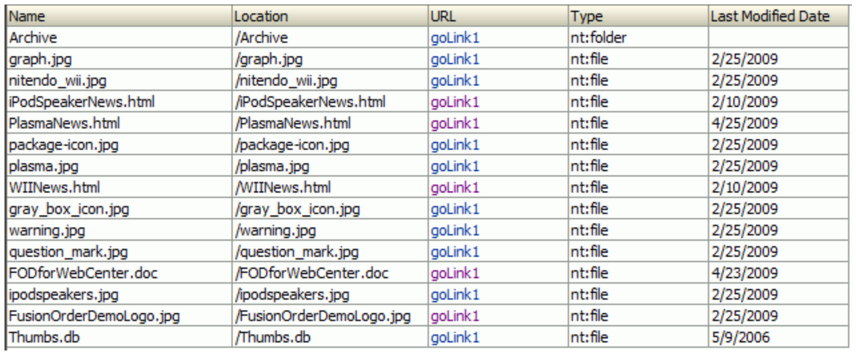
Name	Location	URL	Type	Last Modified Date
#{...name}	#{...path}	goLink1	#{...primaryType}	#{...lastModified}
#{...name}	#{...path}	goLink1	#{...primaryType}	#{...lastModified}
#{...name}	#{...path}	goLink1	#{...primaryType}	#{...lastModified}

Note: You can turn the page caching on or off. To do so, open the page definition and expand **executables**, and select **getItemsIterator** in the Structure window. Then, in the Property Inspector, set **CacheResults** to **true** or **false**, as required.

---

7. Run the page. You should see a list of all files and folders available in your content directory. Figure 8–32 shows a read-only table displaying both files and folders at runtime.

Figure 8–32 Files and Folders Displayed in a Read-Only Table



Name	Location	URL	Type	Last Modified Date
Archive	/Archive	goLink1	nt:folder	
graph.jpg	/graph.jpg	goLink1	nt:file	2/25/2009
nitendo_wii.jpg	/nitendo_wii.jpg	goLink1	nt:file	2/25/2009
iPodSpeakerNews.html	/iPodSpeakerNews.html	goLink1	nt:file	2/10/2009
PlasmaNews.html	/PlasmaNews.html	goLink1	nt:file	4/25/2009
package-icon.jpg	/package-icon.jpg	goLink1	nt:file	2/25/2009
plasma.jpg	/plasma.jpg	goLink1	nt:file	2/25/2009
WIINews.html	/WIINews.html	goLink1	nt:file	2/10/2009
gray_box_icon.jpg	/gray_box_icon.jpg	goLink1	nt:file	2/25/2009
warning.jpg	/warning.jpg	goLink1	nt:file	2/25/2009
question_mark.jpg	/question_mark.jpg	goLink1	nt:file	2/25/2009
FODforWebCenter.doc	/FODforWebCenter.doc	goLink1	nt:file	4/23/2009
ipodspeakers.jpg	/ipodspeakers.jpg	goLink1	nt:file	2/25/2009
FusionOrderDemoLogo.jpg	/FusionOrderDemoLogo.jpg	goLink1	nt:file	2/25/2009
Thumbs.db	/Thumbs.db	goLink1	nt:file	5/9/2006

By default, the table displays file or folder attributes as read-only text (`af:outputText`). The next section describes how to display the `Name` attribute (`name`) as a Go Link (`af:goLink`).

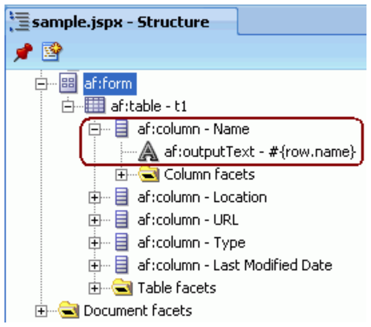
8.5.3.2 Displaying the Name Attribute As a Go Link

In this section, you will convert the `Name` attribute of the table that you created in Section 8.5.3.1, "Displaying Files and Folders in Read-Only Format" into a link using an **ADF Go Link** component. You will also configure the table to show only the **Name** column.

To display the Name attribute as a Go Link:

1. In the Structure window (Figure 8–33), expand the first column of the table (`af:column - Name`) to show the default display format `af:outputText - #{row.name}`.

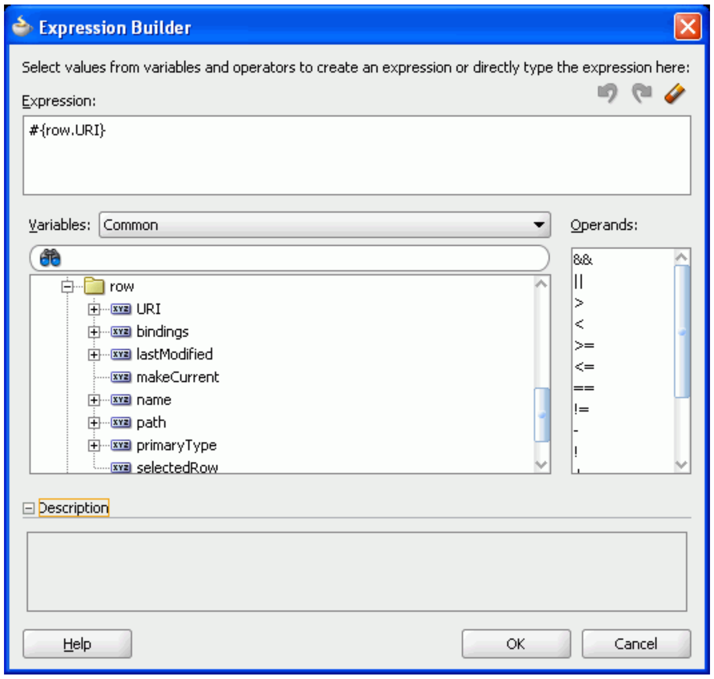
Figure 8–33 Default Formatting for the Name Column



2. Right-click **af:outputText - #{row.name}** and click **Convert**.
3. In the Convert Output Text dialog, select **ADF Faces** as the component category.
4. To convert the **Name** column into a link, in the **Select the item to be created** box, select **Go Link**, then click **OK**. The Confirm Convert dialog displays. Click **OK**.
5. In the Structure window, double-click **af:goLink - goLink2** to display the Property Inspector dialog.
6. To build the `#{row.name}` expression that displays the file or folder name, click down arrow next to the **Text** field, and select **Expression Builder**. The Expression Builder dialog displays.

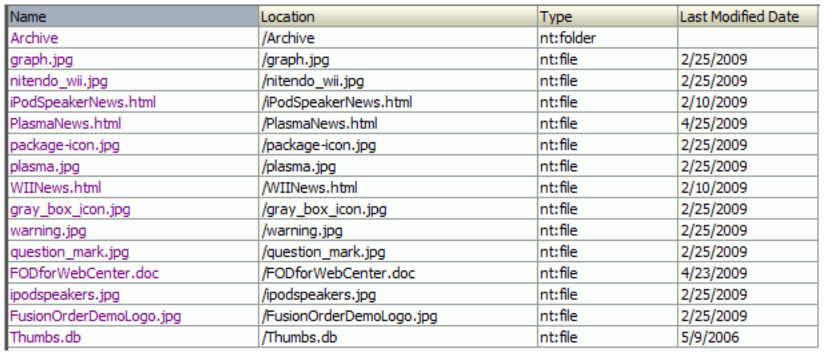
7. Clear the existing expression shown. Then expand **JSP Objects**, **row** and double-click **name**. The `#{row.name}` expression displays under the **Expression** box.
8. In the Property Inspector, under **Link**, click the down arrow next to the **Destination** field, and select **Expression Builder**. The Expression Builder dialog displays.
9. Expand **JSP Objects** and **row** and then double-click **URI**. The `#{row.URI}` expression displays under the Expression box, as shown in Figure 8–34. Click **OK**.

Figure 8–34 Expression Builder



10. In the Structure window, right-click `af:column - URL` and select **Delete**.
11. Right-click the page in the Application Navigator and choose **Run**. You should see a list of hyperlinked file and folder names like the one shown in Figure 8–35. This figure shows the `name` attribute as a link. Clicking a link in the **URI** column opens the respective file or folder and shows its contents.

Figure 8–35 Folder Content Displayed as Hyperlinks

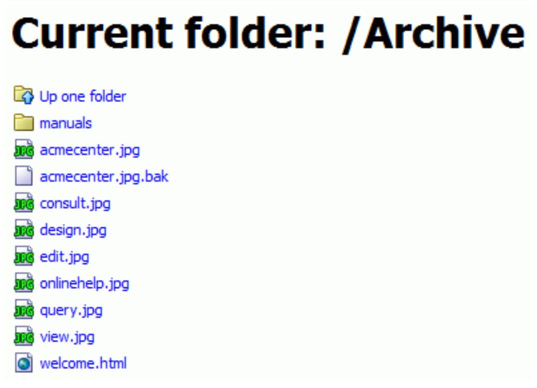


Name	Location	Type	Last Modified Date
Archive	/Archive	nt:folder	
graph.jpg	/graph.jpg	nt:file	2/25/2009
nitendo_wii.jpg	/nitendo_wii.jpg	nt:file	2/25/2009
iPodSpeakerNews.html	/iPodSpeakerNews.html	nt:file	2/10/2009
PlasmaNews.html	/PlasmaNews.html	nt:file	4/25/2009
package-icon.jpg	/package-icon.jpg	nt:file	2/25/2009
plasma.jpg	/plasma.jpg	nt:file	2/25/2009
WIINews.html	/WIINews.html	nt:file	2/10/2009
gray_box_icon.jpg	/gray_box_icon.jpg	nt:file	2/25/2009
warning.jpg	/warning.jpg	nt:file	2/25/2009
question_mark.jpg	/question_mark.jpg	nt:file	2/25/2009
FODforWebCenter.doc	/FODforWebCenter.doc	nt:file	4/23/2009
ipodspeakers.jpg	/ipodspeakers.jpg	nt:file	2/25/2009
FusionOrderDemoLogo.jpg	/FusionOrderDemoLogo.jpg	nt:file	2/25/2009
Thumbs.db	/Thumbs.db	nt:file	5/9/2006

12. Click a file name. The file you pick should appear in a browser window.

13. Click the name of a folder, the contents of the folder display, as shown in Figure 8–36.

Figure 8–36 Folder Contents



Current folder: /Archive

- Up one folder
- manuals
- acmecenter.jpg
- acmecenter.jpg.bak
- consult.jpg
- design.jpg
- edit.jpg
- onlinehelp.jpg
- query.jpg
- view.jpg
- welcome.html

To configure the table to show only the Name column:

1. In the Structure window, under the **af:table - t1** node, delete all but the **Name** column.
2. Double-click the **af:table - t1** node to display the Property Inspector.
3. Under the **Common** tab, in the **Id** field, enter a name, for example `myFiles`, as shown in Figure 8–37, and click **OK**.

Figure 8–37 Table Properties - Common Tab

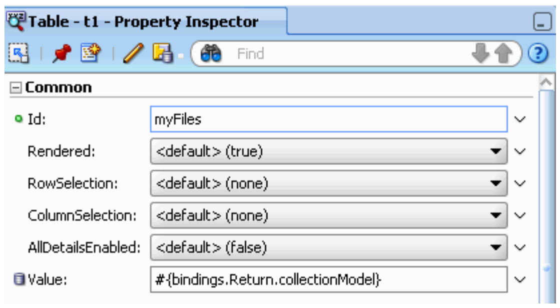


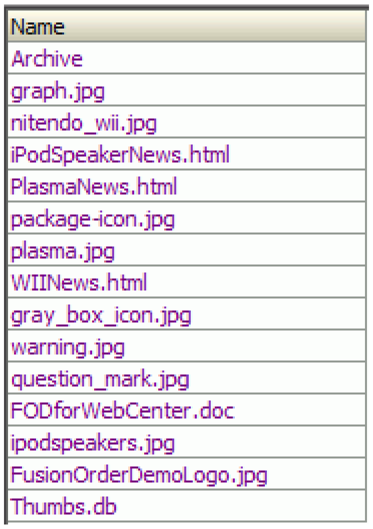
Table - t1 - Property Inspector

Common

Id:	myFiles
Rendered:	<default> (true)
RowSelection:	<default> (none)
ColumnSelection:	<default> (none)
AllDetailsEnabled:	<default> (false)
Value:	#{bindings.Return.collectionModel}

4. Run the page to view the output. Figure 8–38 shows only one column of the table since other rows were removed from the Structure window at design time. This table shows both files and folders, because you left the `type` parameter blank when creating the table.

Figure 8–38 Files and Folders Displayed in a Single-Column Table



Name
Archive
graph.jpg
nitendo_wii.jpg
iPodSpeakerNews.html
PlasmaNews.html
package-icon.jpg
plasma.jpg
WIINews.html
gray_box_icon.jpg
warning.jpg
question_mark.jpg
FODforWebCenter.doc
ipodspeakers.jpg
FusionOrderDemoLogo.jpg
Thumbs.db

In the next section, the **Name** column will be configured to show only files.

8.5.3.3 Configuring a Table to Show Only Files

The `type` attribute is used to configure a table to show only files and not folders.

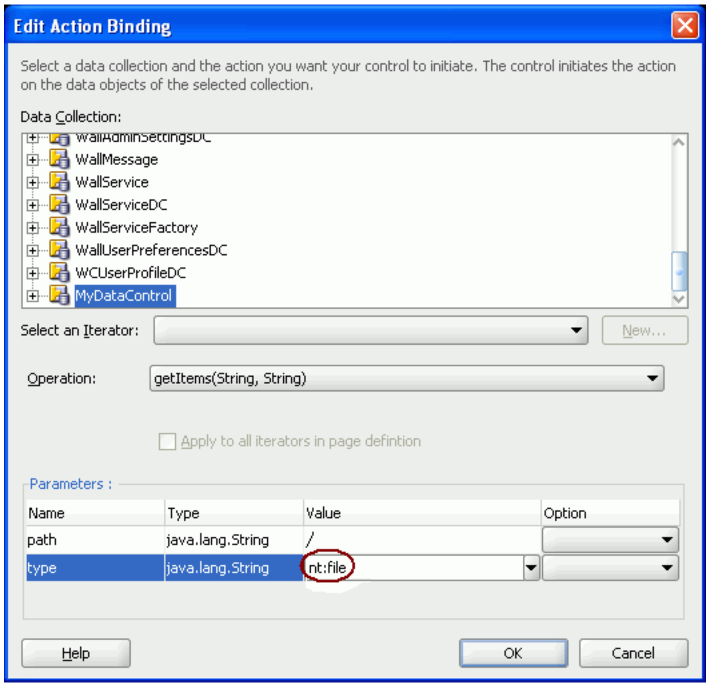
To configure the table to show files:

1. Right-click your `.jspx` page and choose **Go to Page Definition**.

> **Note:** The `RangeSize` binding setting, which is used to control the number of items displayed on a page, is set to `10` by default in the page definition file. You can change it, as required, in the Property Inspector.

2. In the **Overview** tab, double-click **getItems** under **Bindings**. The Edit Action Binding dialog displays.
3. The **type** options are `nt:file` and `nt:folder`. To specify the display of only files, enter `nt:file` under the **Value** column, as shown in Figure 8–39, and click **OK**.

Figure 8–39 Display Files Only


Edit Action Binding

Select a data collection and the action you want your control to initiate. The control initiates the action on the data objects of the selected collection.

Data Collection:

- WallAdminSettingsDC
- WallMessage
- WallService
- WallServiceDC
- WallServiceFactory
- WallUserPreferencesDC
- WCUserProfileDC
- MyDataControl

Select an Iterator:

Operation: getItems(String, String)

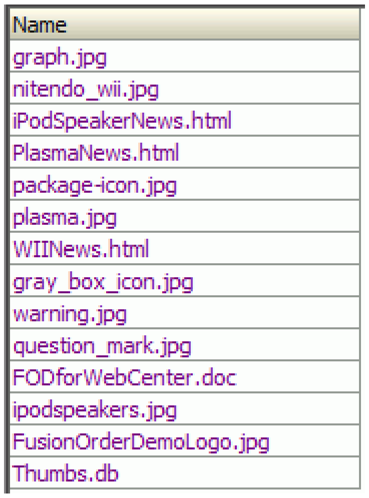
Apply to all iterators in page defintion

Parameters :

Name	Type	Value	Option
path	java.lang.String	/	
type	java.lang.String	nt:file	

4. Now run the page. Figure 8–40 shows only files in the single-column table because the column type is `nt:file`.

Figure 8–40 Files Displayed in a Single-Column Table



Name
graph.jpg
nitendo_wii.jpg
iPodSpeakerNews.html
PlasmaNews.html
package-icon.jpg
plasma.jpg
WIINews.html
gray_box_icon.jpg
warning.jpg
question_mark.jpg
FODforWebCenter.doc
ipodspeakers.jpg
FusionOrderDemoLogo.jpg
Thumbs.db

8.5.4 What Happens at Runtime

The `getItems` method of the JCR data control retrieves the child items of a JCR folder (type `nt:folder`). This method is called with a path to a folder and optionally a type to which the returned child nodes are restricted.

The `path` parameter must be the path of a folder. An exception to this rule is that the root of the repository does not have to be a folder. Hence, the `getItems` method can retrieve the child items for a path that corresponds to a folder, or that is the root of the repository. Otherwise the `getItems` method does not attempt to retrieve the child items and therefore the result set is never populated.

At runtime, the JCR data control calls the `Session.getItem` method. This method returns a JCR node. The data control then calls the `node.getNodes` to retrieve child nodes. The child nodes are filtered according to the specified type, for example, the `nt:file` type. The data control passes on these child nodes to ADF. The data control ensures that the JCR node object is adapted to ADF so that JCR properties are available as data control item attributes that can be consumed through the bindings container.

In the example, each row in the collection returned by the data control is stored as a row in the `af:table` table. However, a column is only displayed for each `af:column` defined in the `af:table`, and not for every possible ADF item attributes returned by the data control. In the first part of the example, the drag and drop action creates an `af:column` for every ADF attribute available for each ADF item returned by the data control.

If the `af:table` is modified to include only the `af:column` for the name, then at runtime it only requests the name of the ADF item. That is, the data control runs a method to fetch the name of the JCR node. At design time, when the name column is converted to an `af:goLink`, the destination of **Go Link** destination is set to the **URI** value of the item, as shown in the following syntax:

```
<af:goLink text="#{row.name}" destination="#{row.URI}"/>
```

At runtime, for each ADF item in the table, the data control runs methods to return both the JCR name of the item and its HTTP URL.

8.5.5 How to Publish Folder Content in a Tree

In this section, you will use the `getItems` method to publish content in a hierarchal tree format. This section describes the following procedures:

- Section 8.5.5.1, "Displaying Files and Folders in Read-Only Format"
- Section 8.5.5.2, "Displaying File Names As Hyperlinks"

Before you begin:

1. Configure a repository connection as described in Section 8.2, "Configuring Content Repository Connections".
2. Configure a data control as described in Section 8.4, "Configuring Content Data Controls for JCR Adapters".
3. Create a JSPX page as described in Section 3.3, "Creating WebCenter Application-Enabled Pages."

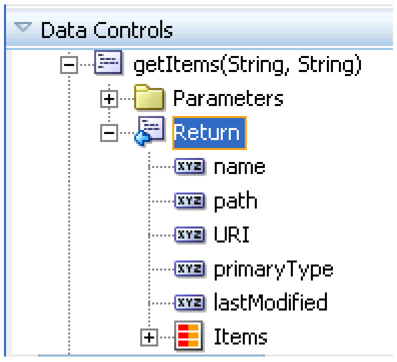
8.5.5.1 Displaying Files and Folders in Read-Only Format

In this section, you will display your content in the tree format.

To display your content in the tree format:

1. In the Application Navigator, double-click your page to open it in the design view.
2. In the Data Controls panel, under your data control, expand the **getItems** method as shown in Figure 8–41.

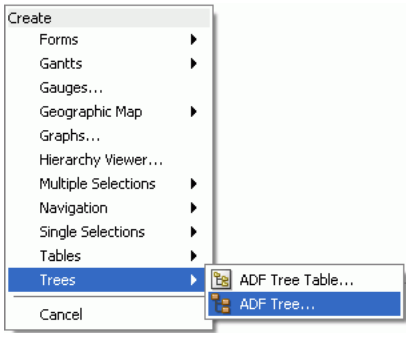
Figure 8–41 Parameters of the getItems Method



3. To display your content as an **ADF Tree**, select the **Return** node and drag it onto the page. From the Create menu, choose **Trees** and then **ADF Tree**, as shown in Figure 8–42.

   If this is the first time you have dropped a node onto the page, the Edit Action Binding dialog displays.

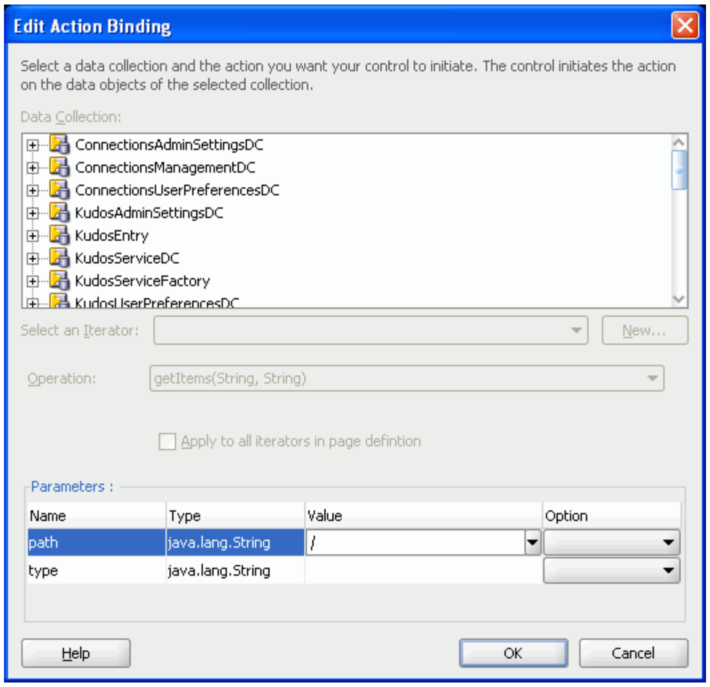
Figure 8–42 Oracle JDeveloper Create Menu for getItems



4. To create a tree that displays everything under the base path, enter the slash (/) for the `path` parameter, as shown in Figure 8–43. To modify or delete this path later, click the arrow icon next to the node in the design mode and choose **Go to Binding** from the context menu.

   Leave the `type` parameter blank to show both files and folders.

Figure 8–43 The Edit Action Binding Dialog



Select a data collection and the action you want your control to initiate. The control initiates the action on the data objects of the selected collection.

- ConnectionsAdminSettingsDC
- ConnectionsManagementDC
- ConnectionsUserPreferencesDC
- KudosAdminSettingsDC
- KudosEntry
- KudosServiceDC
- KudosServiceFactory
- KudosUserPreferencesDC

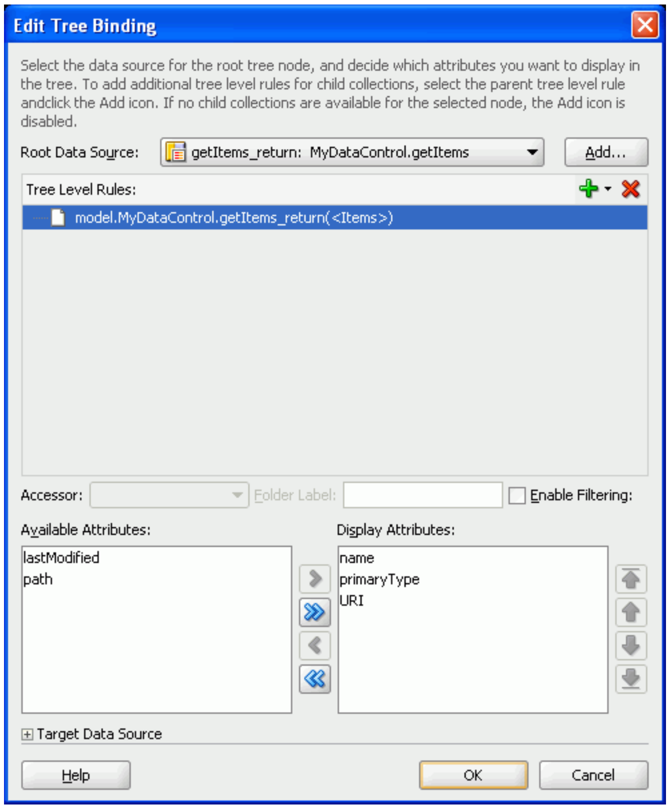
Operation: getItems(String, String)

Apply to all iterators in page defintion

Name	Type	Value	Option
path	java.lang.String	/	
type	java.lang.String		

5. Click **OK**. The Edit Tree Binding dialog displays.
6. To show item names, paths, and types at runtime, under **Available Attributes**, select **URI** and **primary type**, and move them to the **Display Attributes** list.
7. In the **Tree Level Rules** box, click **Add Rule** icon and select **Items**. It creates a rule, as shown in Figure 8–44, which enables the tree to find its child items.

Figure 8–44 Edit Tree Binding



Select the data source for the root tree node, and decide which attributes you want to display in the tree. To add additional tree level rules for child collections, select the parent tree level rule andclick the Add icon. If no child collections are available for the selected node, the Add icon is disabled.

Root Data Source: getItems_return: MyDataControl.getItems

Tree Level Rules: model.MyDataControl.getItems_return(<Items>)

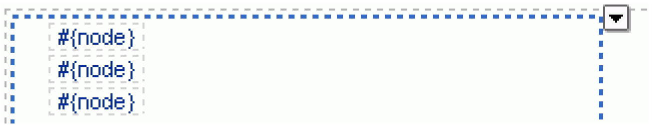
Accessor: Folder Label: Enable Filtering:

Available Attributes: lastModified, path

Display Attributes: name, primaryType, URI

8. Click **OK.** A tree displays in the JSPX page that looks like Figure 8–45.

Figure 8–45 Tree for Navigating Folder Content

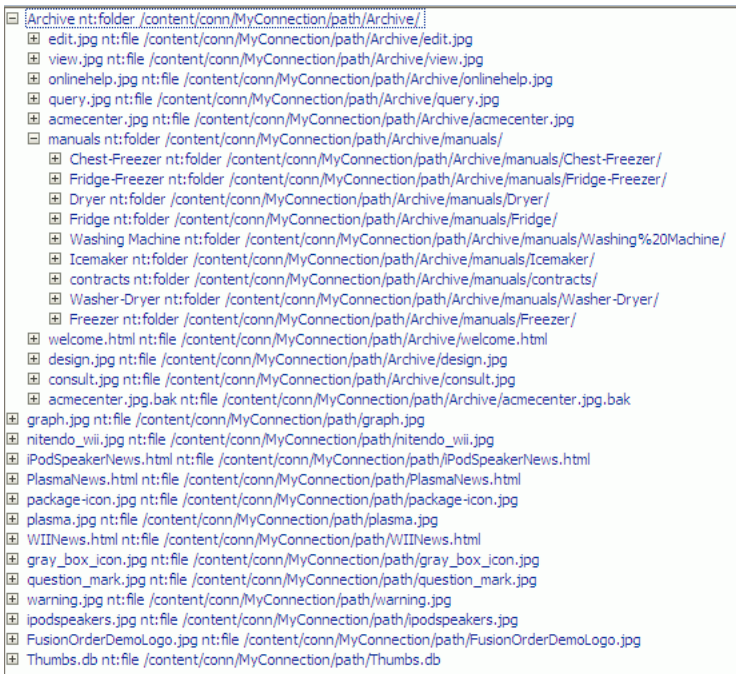


9. Run your page to display the results.

   When the page appears in your browser window, you should see a list of files and folders available through your data control. Figure 8–46 displays a tree of files and folders in the read-only format based on **ADF tree** dropped on the JSF page. Expand a branch to see the content in this subdirectory.

> **Note:** By default, the range size is `10`. To change the number of items displayed in the tree, edit the **RangeSize** property for the data control in the page definition file (`*name*PageDef.xml`).

Figure 8–46 Folder Content Displayed in a Tree



```
⊟ Archive nt:folder /content/conn/MyConnection/path/Archive/
   ⊞ edit.jpg nt:file /content/conn/MyConnection/path/Archive/edit.jpg
   ⊞ view.jpg nt:file /content/conn/MyConnection/path/Archive/view.jpg
   ⊞ onlinehelp.jpg nt:file /content/conn/MyConnection/path/Archive/onlinehelp.jpg
   ⊞ query.jpg nt:file /content/conn/MyConnection/path/Archive/query.jpg
   ⊞ acmecenter.jpg nt:file /content/conn/MyConnection/path/Archive/acmecenter.jpg
   ⊟ manuals nt:folder /content/conn/MyConnection/path/Archive/manuals/
      ⊞ Chest-Freezer nt:folder /content/conn/MyConnection/path/Archive/manuals/Chest-Freezer/
      ⊞ Fridge-Freezer nt:folder /content/conn/MyConnection/path/Archive/manuals/Fridge-Freezer/
      ⊞ Dryer nt:folder /content/conn/MyConnection/path/Archive/manuals/Dryer/
      ⊞ Fridge nt:folder /content/conn/MyConnection/path/Archive/manuals/Fridge/
      ⊞ Washing Machine nt:folder /content/conn/MyConnection/path/Archive/manuals/Washing%20Machine/
      ⊞ Icemaker nt:folder /content/conn/MyConnection/path/Archive/manuals/Icemaker/
      ⊞ contracts nt:folder /content/conn/MyConnection/path/Archive/manuals/contracts/
      ⊞ Washer-Dryer nt:folder /content/conn/MyConnection/path/Archive/manuals/Washer-Dryer/
      ⊞ Freezer nt:folder /content/conn/MyConnection/path/Archive/manuals/Freezer/
   ⊞ welcome.html nt:file /content/conn/MyConnection/path/Archive/welcome.html
   ⊞ design.jpg nt:file /content/conn/MyConnection/path/Archive/design.jpg
   ⊞ consult.jpg nt:file /content/conn/MyConnection/path/Archive/consult.jpg
   ⊞ acmecenter.jpg.bak nt:file /content/conn/MyConnection/path/Archive/acmecenter.jpg.bak
⊞ graph.jpg nt:file /content/conn/MyConnection/path/graph.jpg
⊞ nitendo_wii.jpg nt:file /content/conn/MyConnection/path/nitendo_wii.jpg
⊞ iPodSpeakerNews.html nt:file /content/conn/MyConnection/path/iPodSpeakerNews.html
⊞ PlasmaNews.html nt:file /content/conn/MyConnection/path/PlasmaNews.html
⊞ package-icon.jpg nt:file /content/conn/MyConnection/path/package-icon.jpg
⊞ plasma.jpg nt:file /content/conn/MyConnection/path/plasma.jpg
⊞ WIINews.html nt:file /content/conn/MyConnection/path/WIINews.html
⊞ gray_box_icon.jpg nt:file /content/conn/MyConnection/path/gray_box_icon.jpg
⊞ question_mark.jpg nt:file /content/conn/MyConnection/path/question_mark.jpg
⊞ warning.jpg nt:file /content/conn/MyConnection/path/warning.jpg
⊞ ipodspeakers.jpg nt:file /content/conn/MyConnection/path/ipodspeakers.jpg
⊞ FusionOrderDemoLogo.jpg nt:file /content/conn/MyConnection/path/FusionOrderDemoLogo.jpg
⊞ Thumbs.db nt:file /content/conn/MyConnection/path/Thumbs.db
```

By default, the tree displays file and folder names as read-only text. The next section describes how to create hyperlinks to file names. In the following section, folder names will remain read-only text because they are required for navigation through the tree.

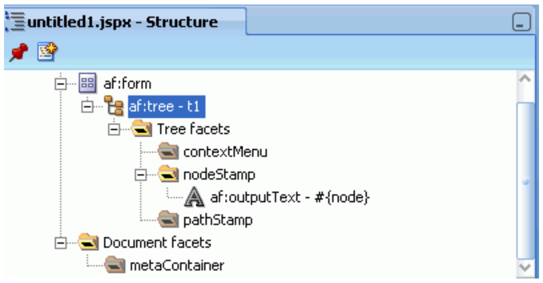
8.5.5.2 Displaying File Names As Hyperlinks

To create hyperlinks to file names and to keep folder names read-only, you need the `af:switcher` component with two facets: one for folders and one for files.

To use the Switcher component for folders and files:

1. In the Structure window, navigate to `nodeStamp` to show the default display format `af:outputText-#{node}`, as shown in Figure 8–47.

Figure 8–47 Default Display Format for Trees



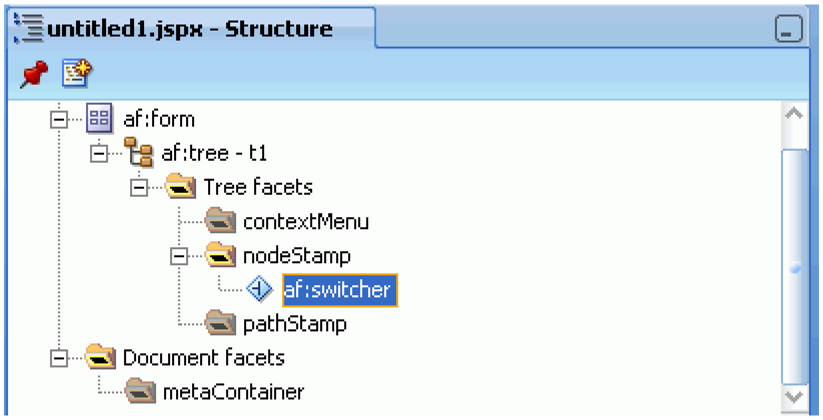
```
untitled1.jspx - Structure
⊟ af:form
   ⊟ af:tree - t1
      ⊟ Tree facets
            contextMenu
         ⊟ nodeStamp
               af:outputText - #{node}
            pathStamp
   ⊟ Document facets
         metaContainer
```

2. Right-click **af:outputText - #{node**} and click **Convert**. The Convert Output Text dialog displays.

3. Select **ADF Faces** as the category of the component. From the **Select an ADF Faces item to create** list, select **Switcher**, and click **OK**. The Confirm Convert dialog displays.

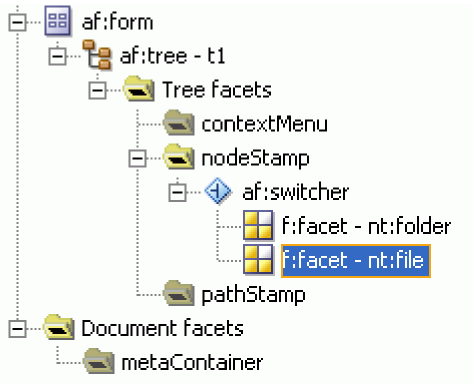
4. Click **OK** to complete the conversion and display the switcher in the Structure window, as shown in Figure 8–48.

Figure 8–48 Output Text Converted to a Switcher Component



5. Double-click **af:switcher** to display the Property Inspector, if it is not displayed.
6. Under the **Common** tab, in the **FacetName** field, enter the expression `#{node.primaryType}`.
7. In the Structure window, insert two facets for the switcher. Right-click **af:switcher**, choose **Insert Inside af:switcher** and then **Facet**. The Insert Facet dialog displays.
8. Name the first facet `nt:folder` and click **OK**. Folder names require no additional formatting, so you can display the node names as plain text. Name the second facet `nt:file`. The facets look like Figure 8–49.

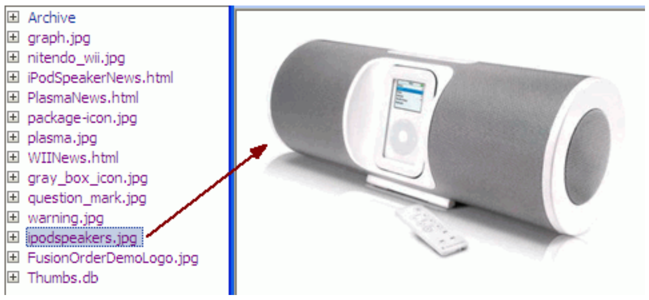
Figure 8–49 Switcher Component with Two Facets



9. Right-click **f:facet - nt:folder**, choose **Insert Inside f:facet - nt:folder**, and then choose **ADF Faces**. The Insert ADF Faces Item dialog displays.
10. Select **Output Text** and click **OK**.
11. Double-click **af:outputText - outputText1** to display the Property Inspector, if it is not displayed already. Under the **Common** tab, in the **Value** field, enter the expression `#{node.name}`.
12. Right-click **f:facet - nt:file** and choose **Insert Inside f:facet - nt:file.** The Insert ADF Faces Item dialog displays.

13. Select **ADF Faces** to display the Insert ADF Item dialog. Then, select **Go Link** from the **Select the item to be created** box, and click **OK**.
14. In the Structure window, double-click **af:goLink - goLink** to display the Property Inspector, if it is not already displayed.
15. Under the **Common** tab, in the **Text** field, enter the `#{node.name}` expression.
16. In the **Destination** field, enter the expression `#{node.URI}`.
17. Run the page. Figure 8–50 displays files names as hyperlinks, because the `nt:file` facet of the switcher under `af:tree` was converted to a link (`node.URI`). Clicking a link displays the respective item.

Figure 8–50 Tree with File Names as Hyperlinks



8.5.6 What Happens at Runtime

The JCR data control `getItems` method is designed to retrieve the child items of a JCR folder (type `nt:folder`). The method is invoked with a path to a folder and optionally a type to which the returned child nodes are restricted. The `path` parameter must be the path of a folder for the `getItems` method to retrieve the child items. An exception to this rule is that the root of the repository does not have to be a folder. Hence, the `getItems` method can retrieve the child items for a path that corresponds to a folder, or that is the root of the repository. Otherwise the `getItems` method does not attempt to retrieve the child items and therefore the result set is never populated. If the path is valid, then the data control invokes `JCR Session.getItem()` on this path which returns a JCR node, and then it invokes `node.getNodes()` to retrieve all child nodes. The child nodes are filtered according to the type supplied, for example, to return only the `nt:file` type child node. This is the result that is provided by the data control to ADF. The data control ensures that the JCR node object is adapted to ADF such that JCR properties are available as data control item attributes that can be consumed by way of the bindings container.

The `af:tree` renders each node in the collection returned by the data control as a node in its tree. In the first part of the example, it displays the `name`, `type`, and `URI` for each node. Each of these values is retrieved through the data control. When the switcher is added to the `af:tree` the node's `primaryType` value is used to differentiate how a node is rendered. If the node is of primary type `nt:folder`, then only its name is shown in the tree node. However if the node is of type `nt:file`, then the node renders `go:Link`, the destination of which is the node's URI. The data control `getItems` method is invoked again to retrieve the child nodes of any folder node in the tree.

8.5.7 How to Add Search Capabilities to Content Repositories

With the help of two examples, this section describes how to add simple and advanced search capabilities for the integrated content. The simple search enables users to search for the content based on name or content fragments in specific locations. The advanced search enables users to search by attribute values of the content.

This section contains the following:

- Section 8.5.7.1, "Adding Simple Search Capabilities"
- Section 8.5.7.2, "Adding Advanced Search Capabilities"

Before you begin:

1. Configure a repository connection as described in Section 8.2, "Configuring Content Repository Connections".
2. Configure a data control as described in Section 8.4, "Configuring Content Data Controls for JCR Adapters".
3. Create a JSPX page as described in Section 3.3, "Creating WebCenter Application-Enabled Pages."

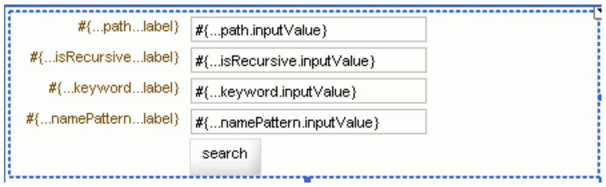
8.5.7.1 Adding Simple Search Capabilities

In this section, you will enable simple search capabilities in your page. This will let you perform wildcard (`%`) search.

To enable the search function:

1. In the Application Navigator, double-click your `.jspx` page to open it.
2. In the Data Controls panel, select the **search** node.
3. To enable users to perform a search by clicking a button, drag and drop the **search** node on your `.jspx` page. From the Create menu, choose **Parameters** and then **ADF Parameter Form**. The Edit Form Fields dialog displays.
4. Click **OK**. The ADF parameter form is added to the page, as shown in Figure 8–51

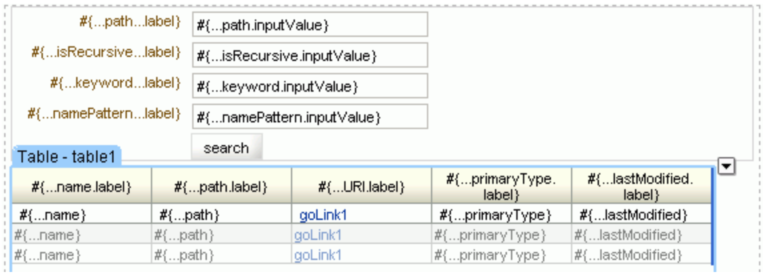
Figure 8–51 ADF Parameter Form in the Design View



#{...path...label}	#{...path.inputValue}
#{...isRecursive...label}	#{...isRecursive.inputValue}
#{...keyword...label}	#{...keyword.inputValue}
#{...namePattern...label}	#{...namePattern.inputValue}
	search

5. To enable the display of search results in a read-only table, drag and drop the **Return** node onto the page. From the **Create** menu, choose **Tables** and then **ADF Read-Only Table**. The Edit Table Columns dialog displays.
6. Click **OK**. A table similar to Figure 8–52 displays.

Figure 8–52 Table with Four Columns - search



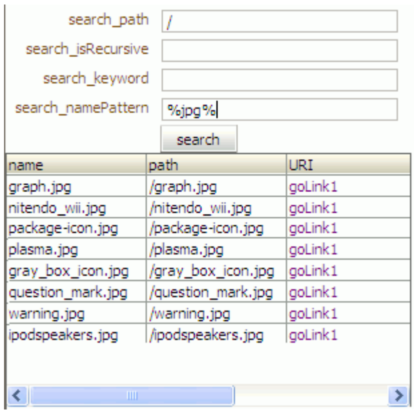
#{...path...label}	#{...path.inputValue}
#{...isRecursive...label}	#{...isRecursive.inputValue}
#{...keyword...label}	#{...keyword.inputValue}
#{...namePattern...label}	#{...namePattern.inputValue}
	search

Table - table1

#{...name.label}	#{...path.label}	#{...URI.label}	#{...primaryType. label}	#{...lastModified. label}
#{...name}	#{...path}	goLink1	#{...primaryType}	#{...lastModified}
#{...name}	#{...path}	goLink1	#{...primaryType}	#{...lastModified}
#{...name}	#{...path}	goLink1	#{...primaryType}	#{...lastModified}

7. Run this page and specify / for **search_path** and %jpg% for **namePattern**. All .jpg files stored in the root of your repository display, as shown in Figure 8–53.

Figure 8–53 Search Results for .jpg Files



search_path	/
search_isRecursive	
search_keyword	
search_namePattern	%jpg%

name	path	URI
graph.jpg	/graph.jpg	goLink1
nitendo_wii.jpg	/nitendo_wii.jpg	goLink1
package-icon.jpg	/package-icon.jpg	goLink1
plasma.jpg	/plasma.jpg	goLink1
gray_box_icon.jpg	/gray_box_icon.jpg	goLink1
question_mark.jpg	/question_mark.jpg	goLink1
warning.jpg	/warning.jpg	goLink1
ipodspeakers.jpg	/ipodspeakers.jpg	goLink1

8.5.7.2 Adding Advanced Search Capabilities

In this section, you will add advanced search capabilities to your page that will enable you to perform search based on the last modified dates of items located in your file system repository.

To enable the advanced search function:

1. In the Application Navigator, open your .jspx page in which you intend to create the advanced search function. The page must be automatically exposed in new managed bean, where name=advancedSearch, class=AdvancedSearch, package=view.

   In this example the page is called advancedSearch.jspx.

2. Double-click the .jspx page to open it.
3. In the Component Palette, select **ADF Faces**.
4. From the list of **ADF Faces** components, drag **Panel Form Layout** onto the page.
5. From the Component Palette, drag **Input Date** into the **Panel Form Layout**.
6. In the Property Inspector, under the **Common** tab, set the **label** to **Modified after**.
7. Under **view**, double-click **AdvancedSearch.java** to open it.
8. Add the following import declarations. These declarations are required for the getPredicates method, which will be added in the next step.

```
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import oracle.vcr.datacontrol.search.Predicate;
import oracle.vcr.datacontrol.search.Operator;
```

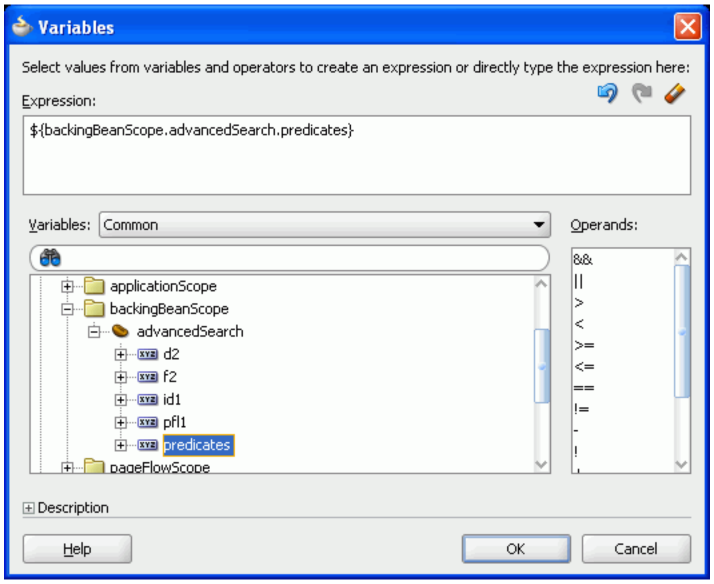
9. Add the following method to enable the advanced search based on last modified dates of the items stored in the repository:

```
public List<Predicate> getPredicates() {
        ArrayList<Predicate> predicates = new ArrayList<Predicate>();
        if (id1.getValue() != null && !id1.getValue().equals(""))
{
```

```
                Calendar cal = Calendar.getInstance();
                cal.setTime((Date)id1.getValue());
                predicates.add(new Predicate("jcr:content/jcr:lastModified"
                                             , Operator.GREATER_THAN
                                             , cal));
            }
            // ... other predicates
            if (predicates.size()>0)
                return predicates;
            return null;
        }
```

10. From the Data Controls panel, drag the **advancedSearch** node onto the `.jspx` page. From the Create menu, choose **Methods** and then **ADF Button**. The Edit Action Binding dialog displays.
11. For **Path**, specify the path to the directory in which the search will be performed, for example, `/`. For **isRecursive** specify `true`, and **matchAny** specify `false`.
12. Select the arrow next to the **Value** field for predicates and then select **Show El Expression Builder**. The Variables dialog displays.
13. Expand **ADF Managed Beans**, **backingBeanScope**, **advancedSearch** and select **predicates**. This adds the expression `${backingBeanScope.advancedSearch.predicates}`, as shown in Figure 8–54, and click **OK**.

Figure 8–54 Variables Dialog - predicates


Variables

Select values from variables and operators to create an expression or directly type the expression here:

Expression:

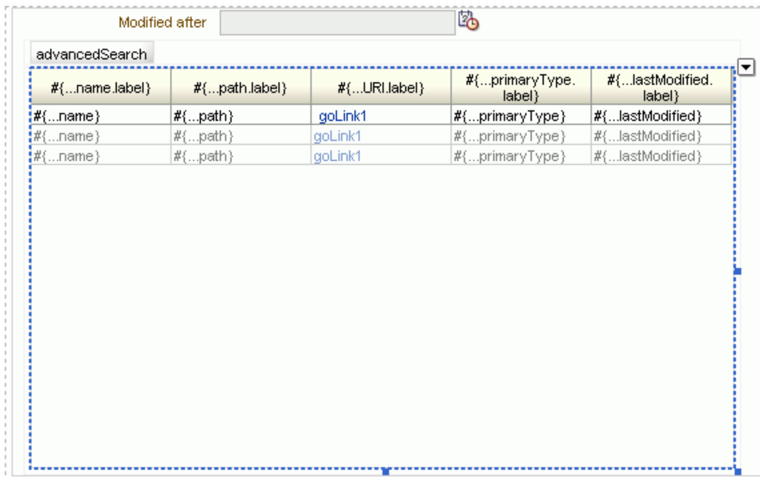
${backingBeanScope.advancedSearch.predicates}

Variables: Common

Operands:

14. Click **OK** in the Edit Action Binding dialog.
15. In the Data Controls panel, expand the **advancedSearch** node. Then drag **Return** and drop it onto the page, after the advancedSearch ADF button. From the Create menu, choose **Table** and then **ADF Read-only Table**.
16. In the Edit Table Columns dialog, edit labels of the columns, if needed. Then click **OK**. The Design view looks like Figure 8–55.

Figure 8–55 Advanced Search - Design View



Modified after

advancedSearch

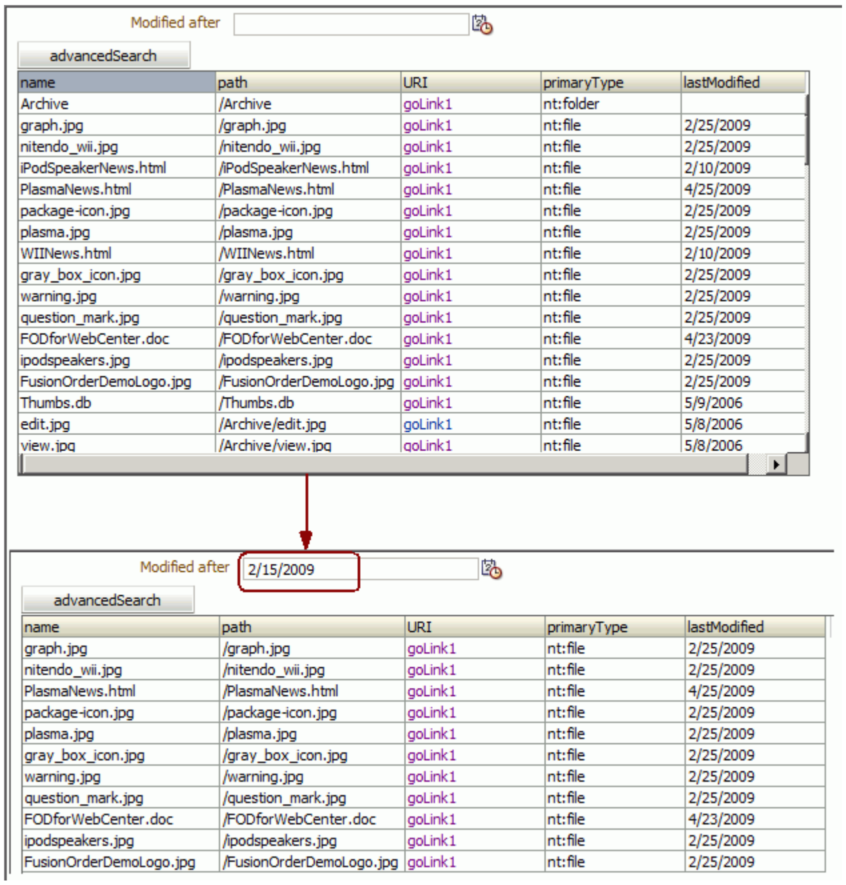
#{...name.label}	#{...path.label}	#{...URI.label}	#{...primaryType.label}	#{...lastModified.label}
#{...name}	#{...path}	goLink1	#{...primaryType}	#{...lastModified}
#{...name}	#{...path}	goLink1	#{...primaryType}	#{...lastModified}
#{...name}	#{...path}	goLink1	#{...primaryType}	#{...lastModified}

17. In the Property Inspector, under the **Behavior** tab, set the `ContentDelivery` property to `Immediate`.

18. To view the page in a browser, under the Application Navigator, right-click the `.jspx` and choose **Run**. The `advancedSearch` page displays in the browser.

19. Enter a *last modified* date and click the **advancedSearch** button. The files modified after that date are displayed, as shown in Figure 8–56.

Figure 8–56 Advanced Search Results



Modified after

advancedSearch

name	path	URI	primaryType	lastModified
Archive	/Archive	goLink1	nt:folder	
graph.jpg	/graph.jpg	goLink1	nt:file	2/25/2009
nitendo_wii.jpg	/nitendo_wii.jpg	goLink1	nt:file	2/25/2009
iPodSpeakerNews.html	/iPodSpeakerNews.html	goLink1	nt:file	2/10/2009
PlasmaNews.html	/PlasmaNews.html	goLink1	nt:file	4/25/2009
package-icon.jpg	/package-icon.jpg	goLink1	nt:file	2/25/2009
plasma.jpg	/plasma.jpg	goLink1	nt:file	2/25/2009
WIINews.html	/WIINews.html	goLink1	nt:file	2/10/2009
gray_box_icon.jpg	/gray_box_icon.jpg	goLink1	nt:file	2/25/2009
warning.jpg	/warning.jpg	goLink1	nt:file	2/25/2009
question_mark.jpg	/question_mark.jpg	goLink1	nt:file	2/25/2009
FODforWebCenter.doc	/FODforWebCenter.doc	goLink1	nt:file	4/23/2009
ipodspeakers.jpg	/ipodspeakers.jpg	goLink1	nt:file	2/25/2009
FusionOrderDemoLogo.jpg	/FusionOrderDemoLogo.jpg	goLink1	nt:file	2/25/2009
Thumbs.db	/Thumbs.db	goLink1	nt:file	5/9/2006
edit.jpg	/Archive/edit.jpg	goLink1	nt:file	5/8/2006
view.jpg	/Archive/view.jpg	goLink1	nt:file	5/8/2006

Modified after 2/15/2009

advancedSearch

name	path	URI	primaryType	lastModified
graph.jpg	/graph.jpg	goLink1	nt:file	2/25/2009
nitendo_wii.jpg	/nitendo_wii.jpg	goLink1	nt:file	2/25/2009
PlasmaNews.html	/PlasmaNews.html	goLink1	nt:file	4/25/2009
package-icon.jpg	/package-icon.jpg	goLink1	nt:file	2/25/2009
plasma.jpg	/plasma.jpg	goLink1	nt:file	2/25/2009
gray_box_icon.jpg	/gray_box_icon.jpg	goLink1	nt:file	2/25/2009
warning.jpg	/warning.jpg	goLink1	nt:file	2/25/2009
question_mark.jpg	/question_mark.jpg	goLink1	nt:file	2/25/2009
FODforWebCenter.doc	/FODforWebCenter.doc	goLink1	nt:file	4/23/2009
ipodspeakers.jpg	/ipodspeakers.jpg	goLink1	nt:file	2/25/2009
FusionOrderDemoLogo.jpg	/FusionOrderDemoLogo.jpg	goLink1	nt:file	2/25/2009

8.5.8 What Happens at Runtime

Clicking the **search** button at runtime invokes the `search` method and the values provided through the UI are used as the parameters for the search. At runtime, you can perform the wild (`%`) card search. For example, to search for files that have `.jpg` extension, enter `/` in the **search_path** field, enter `%jpg` in the namePattern field and then click **Search**. All files with the `.jpg` extension will display in the read-only table.

At runtime the JCR data control uses the given parameters to construct an `XPath` query for the given parameters. For the simple search example, the query is:

```
Non-recursive:
XPath query /jcr:root/element(*, nt:file)[jcr:like(ojcr:local-name(), '%jpg%')]

Recursive:
XPath query /jcr:root//element(*, nt:file)[jcr:like(ojcr:local-name(), '%jpg%')]
```

In the advanced search example, first a backing bean is added to the JSPX page, because the backing bean is required to construct the `predicate` parameter value of the `advancedSearch` method. Then the **Panel Form Layout UI** component is dropped onto the JSPX page. In this component, the `InputDate` component is dropped, which is used at runtime to supply the date-based search criterion. The `advancedSearch` method predicates parameter allows for a combination of predicates to be supplied to the `search` method. Each can specify the value of an item's properties that must apply for the search. In this example, only a modification date is tested in the predicates, but potentially multiple tests could be included, for example a modification date and a `mimeType`.

The backing bean's `getPredicates` method is handcoded to construct the `predicates` method from the date provided by the page at runtime. At design time, the return value of the `predicates` method is bound into the predicates `advancedSearch` method parameter. At runtime this invokes the `getPredicates` method before invoking the `advancedSearch` method to construct the correct `predicate` value.

At runtime the JCR data control uses the given parameters to construct an `XPath` query for the given parameters. For this example the query is:

```
/jcr:root//element(*, nt:hierarchyNode)[jcr:content/@jcr:lastModified >
xs:dateTime('2009-02-15T00:00:00.000+05:30')]
```

8.5.9 What You May Need to Know When Using Search Capabilities

Consider the following points while adding search capabilities:

- How certain operations work depends on the implementation of the adapter and the underlying repository. While read and query operations are similar, full text search works differently. Another example is, the file system and Oracle Portal adapters do not support search based on the `primaryType` attribute. The only supported way to search based on type is through the element (`*`, type) construct.
- If you use the Oracle Portal adapter, then the behavior of search functionality varies depending on whether Oracle Text is enabled or not. If Oracle Text is disabled, then the search is performed in the Oracle Portal content metadata. With Oracle Text turned on, all indexed content is searched, which includes the contents of files. This also applies to Oracle Content Server. That is, Oracle Content Server-based adapter performs full text index operation on its documents using Oracle Text.

- The Oracle Portal adapter does not support translations and only returns content in the base language of a page group. Searching across multiple page groups with different base languages is not supported.

9 Consuming Portlets

This chapter describes how to add portlets to the pages of your custom WebCenter application and the options that accompany this process.

This chapter includes the following sections:

This chapter does not cover Oracle JDeveloper or Oracle ADF page creation basics. It covers only those aspects of page creation that are specific to custom WebCenter application pages. Therefore, you should familiarize yourself with the information covered in the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* before reading this chapter.

For information about creating portlets, which can then be consumed by custom WebCenter application pages, see the following chapters:

- Chapter 27, "Overview of Portlets"
- Chapter 28, "Creating Portlets with the Oracle JSF Portlet Bridge"
- Chapter 29, "Creating Portlets with the Portlet Wizard"
- Chapter 31, "Creating Portlets with OmniPortlet"
- Chapter 32, "Creating Content-Based Portlets with Web Clipping"

9.1 Introduction to Consuming Portlets

Oracle WebCenter Framework enables you to consume a portlet by registering its producer either with an application or with the Resource Palette from where you can add it to any application. After you register the producer, its portlets appear under the registered producer's name under the Connections node in the Application Resources panel or in the Resource Palette.

Your application can consume portlets that you build and portlets that you receive from a third party, such as a packaged-application vendor.

There are many options associated with portlet consumption. For example, you can choose to place portlets straight onto a page or nest them in an Oracle Composer component, you can adjust many attributes of the portlet tag, and you can wire portlets together.

9.2 Registering Portlet Producers with a Custom WebCenter Application

Before you can add a portlet to a custom WebCenter application page, you must register the portlet's producer with the application. You can register portlet producers in two ways:

- Register the portlet producer with a specific application. Use this option if you are not likely to want to register the producer with other applications.
- Register the portlet producer using the Resource Palette. Use this option to use the producer's portlets in multiple applications.

A portlet that is available in the Resource Palette can be added to any of your custom WebCenter applications by dropping it on the page as you would any other component. When you add a portlet from the Resource Palette, its producer gets registered with the application if you have not already done so. You can drag and drop a whole producer connection from the Resource Palette into the Application Resources panel of the Application Navigator. This registers the producer with the application. Alternatively, you can right-click a producer in the Resource Palette and choose **Add to Application** from the context menu to register the producer with the currently open application.

JDeveloper provides wizards for registering both WSRP producers and Oracle PDK-Java producers.

Note: If your application is source controlled, you must manually create elements in the source control system for any new files created during producer registration. Any files that are already source controlled are checked out automatically by the producer registration process.

For more information about producers, see Section 27.2.3, "Deployment Type." For information about obtaining prebuilt portlets through Oracle, see Section 3.8, "Using Integrated WLS." For information about using JDeveloper's portlet creation wizards, see Chapter 29, "Creating Portlets with the Portlet Wizard." For more information about portlets, see Chapter 27, "Overview of Portlets."

9.2.1 How to Register a WSRP Portlet Producer

When you register a WSRP portlet producer, you provide basic information that describes the producer's operational parameters. This information is used by the portlet-consuming application to communicate with the producer and with the portlets through the producer.

Oracle WebCenter Framework supports both WSRP 1.0 and WSRP 2.0 producers. The WSRP 2.0 standard, among others, provides support for inter-portlet communication and export and import of portlet customizations. You can leverage the benefits of WSRP 2.0 while building standards-based JSR 168 portlets. To take advantage of more advanced features of WSRP 2.0, use the Oracle-specific `oracle-portlet.xml` metadata extensions.

The Register WSRP Portlet Producer wizard is the entry point for registering both WSRP 1.0 and 2.0 producers. When registration is successful, the newly registered producer displays in JDeveloper either in the Application Resources panel of the Application Navigator, or in the Resource Palette, depending on where you created the connection. You can then select portlets from the producer for placement on your application (`.jspx`) pages.

Note: If you are registering the PeopleSoft portlet producer, you must set the `require-iframe` element to `true` in `oracle-portlet.xml`, for example:

```
<portlet-extension>
    <portlet-name>myPortlet</portlet-name>
    <portlet-id>18</portlet-id>
    <require-iframe>true</require-iframe>
    <minimum-wsrp-version>2</minimum-wsrp-version>
  </portlet-extension>
```

You also use the Register WSRP Portlet Producer wizard to register JSF portlets, which are portletized JSF applications or portletized ADF task flows. Once you create a portlet from a JSF application, you can deploy the portlet to a WLS instance and register the JSF portlet producer as you would register any WSRP portlet producer. The Oracle Portlet Bridge exposes JSF applications and task flows as JSR 168 portlets. For more information, see Chapter 28, "Creating Portlets with the Oracle JSF Portlet Bridge."

Note: In the Register WSRP Portlet Producer wizard, if you click **Cancel** after you have clicked **Finish,** the registration is not canceled.

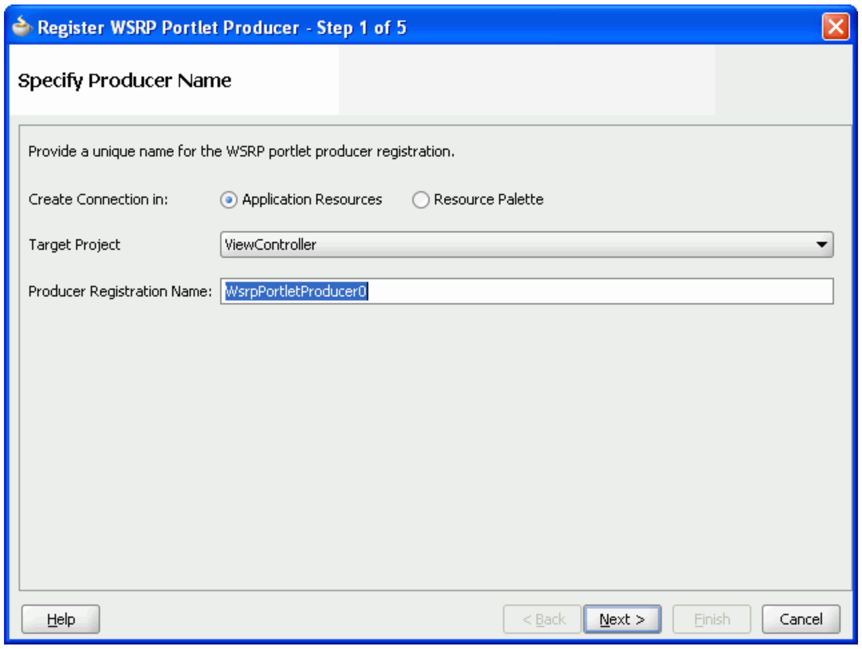
To register a WSRP portlet producer:

1. In the Application Resources panel of the Application Navigator, right-click **Connections,** choose **New Connection** and then choose **WSRP Producer.**

   For other methods of invoking the wizard, such as from the Resource Palette, see Section 3.9, "Accessing Connection Wizards."

2. In the Specify Producer Name page (Figure 9–1) of the Register WSRP Portlet Producer wizard, the **Create Connection in** option is set depending on how you accessed the wizard. The default selection is **Application Resources** if you invoked the wizard from an application, and **Resource Palette** if you invoked the wizard from the Resource Palette. You can change this option at this point.

Figure 9–1 The Specify Producer Name Page (WSRP Producer)



Register WSRP Portlet Producer - Step 1 of 5

Specify Producer Name

Provide a unique name for the WSRP portlet producer registration.

Create Connection in: [✓ Application Resources] Resource Palette

Target Project: ViewController

Producer Registration Name: WsrpPortletProducer0

3. From the **Target Project** list, select the project to be configured for the WSRP producer connection.

   This should be the same project as the one in which you intend to consume the portlets.

   You can change this option only if you invoked the wizard from the Application Navigator.

4. In the **Producer Registration Name** field, enter a name for the producer registration that is unique among all connections and then click **Next.**

5. In the Specify Connection Details page (Figure 9–2), in the **WSDL URL** field, enter the producer's URL.

   The syntax varies according to your WSRP implementation, for example, the sample WSRP producer uses the following syntax:

   - `http://`*`host`*`:`*`port`*`/`*`context-root`*`/portlets/wsrp1?WSDL`
   - `http://`*`host`*`:`*`port`*`/`*`context-root`*`/portlets/wsrp2?WSDL`
   - `http://`*`host`*`:`*`port`*`/`*`context-root`*`/portlets?WSDL` (WSRP 1.0 for backward compatibility)

   Where:

   - *`host`* is the server to which your producer has been deployed.
   - *`port`* is the port to which the server is listening for HTTP requests.
   - *`context-root`* is the Web application's context root.
   - `portlets[/wsrp(1|2)]?WSDL` is static text. The text entered here depends on how the producer is deployed.

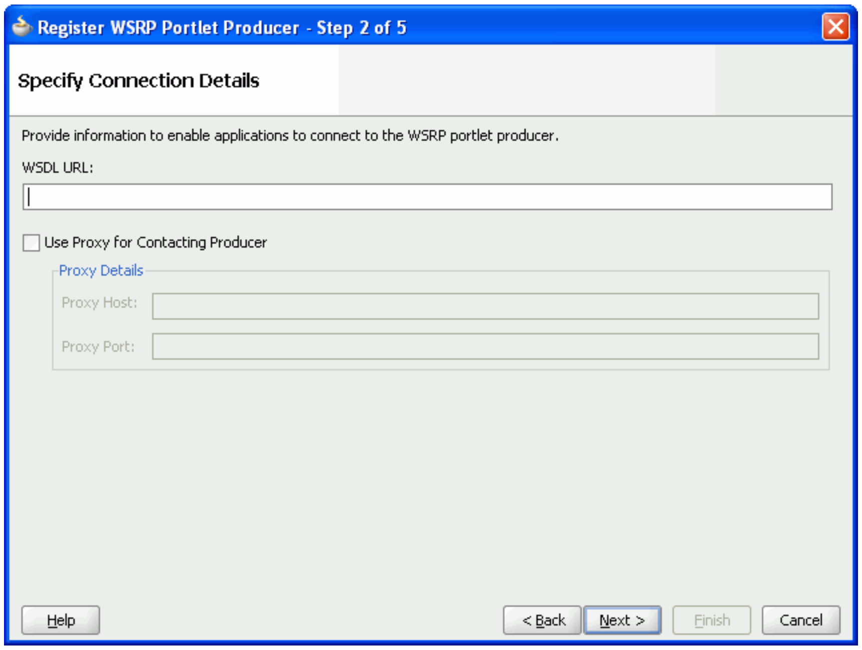
   For example:

   ```
   http://myhost.example.com:7101/portletapp/portlets/wsrp2?WSDL
   ```

You can access the producer test page through the URL:

`http://host:port/context-root/info`

Figure 9–2 The Specify Connection Details Page (WSRP Producer)


Register WSRP Portlet Producer - Step 2 of 5

Specify Connection Details

Provide information to enable applications to connect to the WSRP portlet producer.

WSDL URL:

Use Proxy for Contacting Producer

Proxy Details

Proxy Host:

Proxy Port:

6. If the application and the producer are separated by a firewall, an HTTP proxy is needed for communication between the application and the producer. If this is the case for your application, select **Use Proxy for Contacting Producer** and specify the URL and port number of the proxy.

   Note: The proxy fields in this step default to the proxy preferences set in JDeveloper Preferences (from the main menu, choose **Tools > Preferences,** and then select **Web Browser and Proxy**).

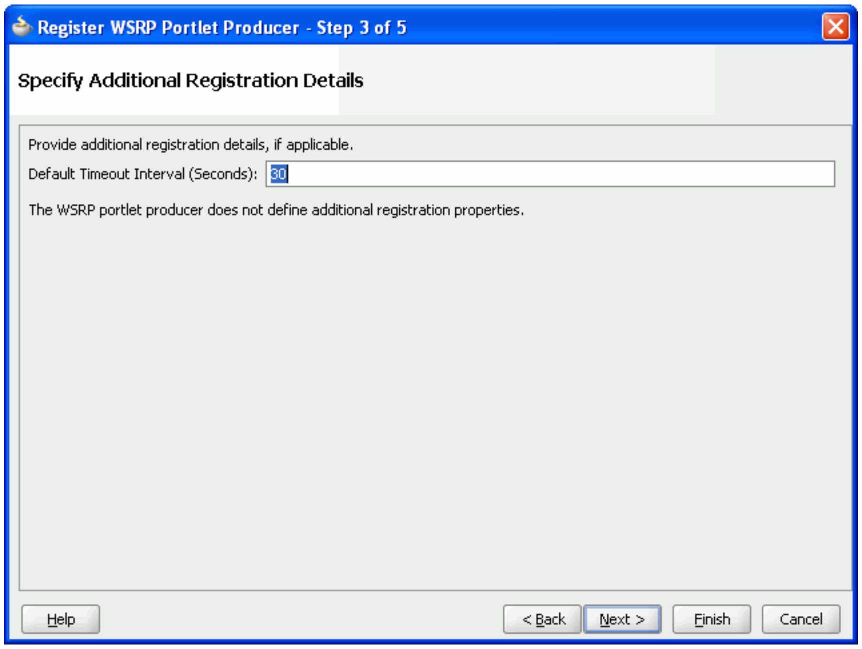
7. Click **Next.**

   The connection to the producer is tested. If there are any problems, an error message displays. You must resolve any problems before you can continue.

8. In the Specify Additional Registration Details page (Figure 9–3), in the **Default Timeout Interval (Seconds)** field, enter the number of seconds to wait for the producer to respond during design time operations.

   Some producers define additional registration properties. In such cases, the properties are displayed in a table on this page of the wizard. You can enter values for these additional properties in the table. These properties are producer-specific and are used only at registration time. That is, they collect information that consumer applications send to producers at registration time; the producers store this information against the consumers and use it subsequently.

Figure 9–3 The Specify Additional Registration Details Page (WSRP Producer)


Register WSRP Portlet Producer - Step 3 of 5
Specify Additional Registration Details

Provide additional registration details, if applicable.

Default Timeout Interval (Seconds): 30

The WSRP portlet producer does not define additional registration properties.

Help
< Back
Next >
Finish
Cancel

9. If you are registering the producer in the Resource Palette, click **Finish** to complete the registration.

   If you are registering the producer in the Application Resources panel, and plan to request authentication whenever the producer (and consequently, its portlet) is accessed, click **Next** and follow the remaining steps. If you do not want to configure security, click **Finish.**

   If the producer declares user categories, when you click **Finish,** the Register WSRP Portlet Producer dialog displays. Click **Yes** and see Section 9.2.2, "How to Map a Producer's Declared User Categories to an Application's Defined Java EE Security Roles." Click **No** to decline this opportunity and complete the registration process.

10. In the Configure Security Attributes page (Figure 9–4), from the **Token Profile** list, select the type of token profile to use for authentication with the WSRP producer:

    - **None**—No token; no WS-Security header is attached to the SOAP message.

      If you select this option, you do not need to complete the rest of the wizard. Click **Finish.**

    - **SAML Token with Message Integrity**——SAML (Security Assertion Markup Language) Token Profile-based identity propagation with certificate based message integrity. SAML is an XML-based approach for passing security tokens defining authentication and authorization rights. An attesting entity (that has trust relationship with the receiver) vouches for the verification of the subject by method called sender-vouches.

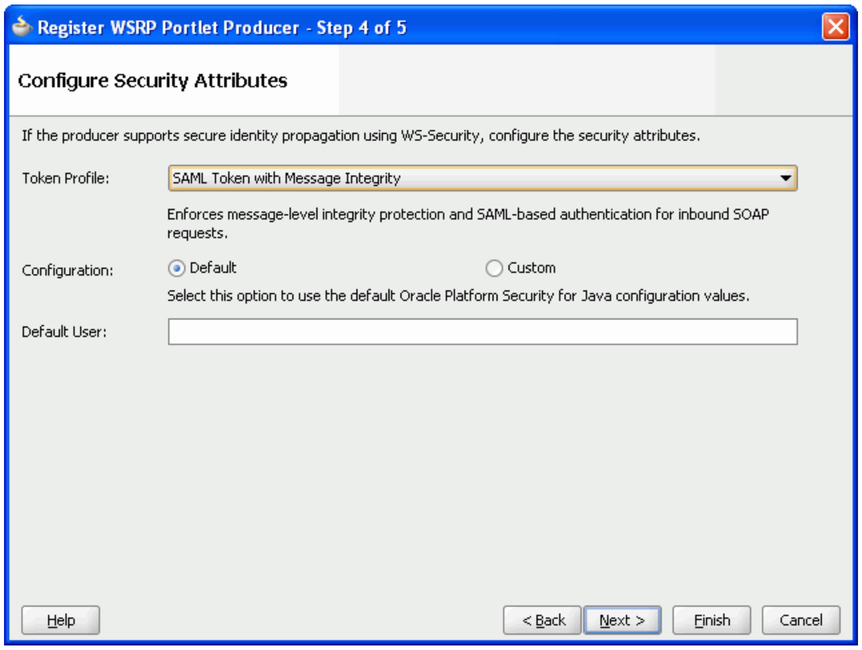
    - **SAML Token with Message Protection**—SAML (Security Assertion Markup Language) Token based identity propagation with certificate based message integrity and confidentiality.

    - **User Name Token without Password**—A Web service consumer can supply a user name token to identify the requester by user name only to authenticate that identity to the Web service producer.

- **User Name Token with Password**—A Web service consumer can supply a user name token to identify the requester by user name and password to authenticate that identity to the Web service producer.

  This token profile is used when you want to consume WSRP producers that have a different identity store. For this token profile case, ensure that you define an external application pertaining to the producer and associate this external application with the producer being registered.

Figure 9–4 The Configure Security Attributes Page (WSRP Producer)



Configure Security Attributes

If the producer supports secure identity propagation using WS-Security, configure the security attributes.

Token Profile: SAML Token with Message Integrity

Enforces message-level integrity protection and SAML-based authentication for inbound SOAP requests.

Configuration: [✓ Default] Custom

Select this option to use the default Oracle Platform Security for Java configuration values.

Default User:

11. Select the configuration type:

    - **Default**—If you choose default, then all the default keystore attributes, that is location, password, keystore type, signature key and alias, encryption key and alias are picked up from the JPS (Java Platform Security) configuration. The value for recipient alias is used from the policy being used. The WebLogic Server where the application is deployed must be configured for WS-Security. For more information, see "Securing a WSRP Producer with WS-Security" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.
    - **Custom**—If you select this option, then you must enter the appropriate keystore attributes in the next page of the wizard.

12. In the **Default User** field, enter a user name to assert to the remote producer when the user has not authenticated to the custom WebCenter application.

    When unauthenticated, the identity *anonymous* is associated with the application user. The value *anonymous* may be inappropriate for the remote producer, so you may need to specify an alternative identity here. Keep in mind though, that in this case, the custom WebCenter application has not authenticated the user so the default user you specify should be a low privileged user in the remote producer. If the user has authenticated to the application, then the user's identity is asserted rather than the default user.

Note: If you specify a **Default User**, the remote producer must be set up to accept this information. This is done by setting the `<strict-authentication>` flag in `oracle-portlet.xml` to `true`. For more information about this flag, see Section A.3.2, "oracle-portlet.xml."

The **Default User** field does not appear if you selected **User Name Token with Password.**

13. In the **Issuer Name** field, enter the name of the issuer of the SAML Token, for example `www.oracle.com.`

    This field appears only if you selected an **SAML Token** option from the **Token Profile** list, and **Custom** from the **Configuration** options. The issuer name is the attesting entity that vouches for the verification of the subject.

14. Select **Associate Producer with External Application**, then select the application, if this producer must provide authentication to an external application.

    For more information, see Section 24.2.3, "Managing External Applications."

    This option is available only if you selected **User Name Token with Password.**

15. Click **Next.**

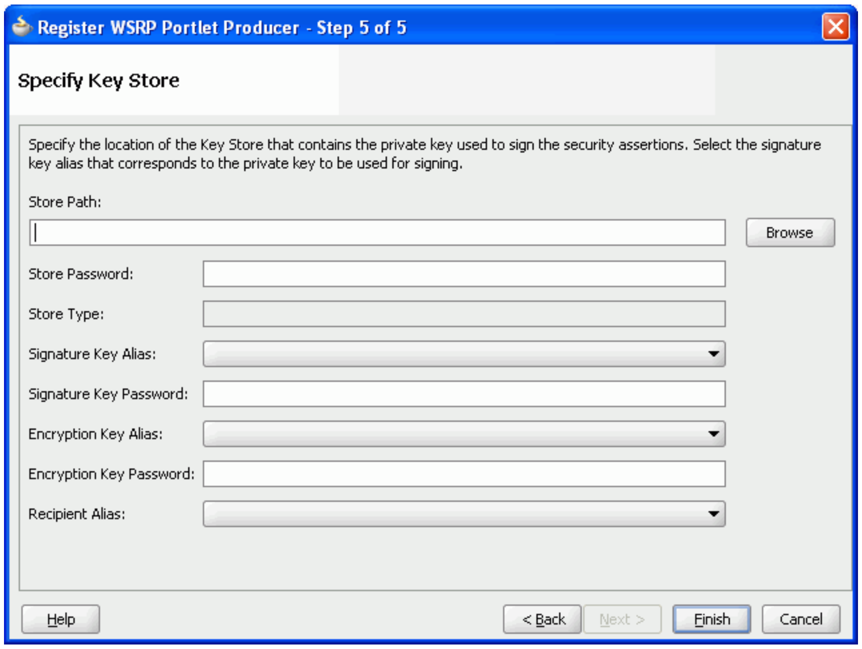
    If you selected **Default** as the configuration option, then the fields on the Specify Key Store page are disabled. Click **Finish** to complete the registration.

    If the producer declares user categories, when you click **Finish,** the Register WSRP Portlet Producer dialog displays. Click **Yes** and see Section 9.2.2, "How to Map a Producer's Declared User Categories to an Application's Defined Java EE Security Roles." Click **No** to decline this opportunity and complete the registration process.

16. In the Specify Key Store page, in the **Store Path** field, provide the full path to the keystore that contains the certificate and the private key that is used for signing some parts (security token and SOAP message body) of the SOAP message.

    If you are not sure of the full path, click **Browse** to navigate to and select the file. The selected file should be a keystore created with the Java keytool.

Figure 9–5 The Specify Key Store Page (WSRP Producer)


Register WSRP Portlet Producer - Step 5 of 5

Specify Key Store

Specify the location of the Key Store that contains the private key used to sign the security assertions. Select the signature key alias that corresponds to the private key to be used for signing.

Store Path:

Store Password:

Store Type:

Signature Key Alias:

Signature Key Password:

Encryption Key Alias:

Encryption Key Password:

Recipient Alias:

17. In the **Store Password** field, provide the password to the keystore that was set when the keystore was created.

    The keystore password must be correct for the **Store Type** field and the **Signature Key Alias** list to populate.

    If an incorrect keystore path or password is entered, then an error message appears stating that the password is invalid and must be corrected. All fields on this screen except for **Store Path** and **Store Password** are disabled until you specify the correct values.

18. After you provide the correct keystore path and password, press the Tab key to move to another active field (for example, the **Store Path** field). This ensures that the **Store Type** field and the **Signature Key Alias** list are properly populated.

19. The **Store Type** value is read from the keystore and is never editable. The store type is always **JKS** (Java Key Store).

20. From the **Signature Key Alias** list, select the signature key alias.

    This list populates automatically when the correct password is entered in the **Store Password** field. The **Signature Key Alias** is the identifier for the certificate associated with the private key that is used for signing. The key aliases found in the specified keystore are available in the list. Select the one to be used for signing.

21. In the **Signature Key Password** field, specify the password for accessing the key identified by the alias specified in **Signature Key Alias.**

22. Optionally, from the **Encryption Key Alias** list, select the encryption key alias.

    This list populates automatically when the correct password is entered in the **Store Password** field. The key aliases found in the specified keystore are available in the list. Select the one to be used for encryption.

23. Optionally, in the **Encryption Key Password** field, specify the password for accessing the key identified by the alias specified in **Encryption Key Alias**.

24. From the **Recipient Alias** field, select the keystore alias that is associated with the producer's certificate and then click **Finish.**

    This certificate is used to encrypt the message to the producer.

    This field is not displayed if you selected **SAML Token with Message Integrity** as the **Token Profile** in the Configure Security Attributes page of the wizard.

    If the producer declares user categories, when you click **Finish,** a dialog displays asking if you want to map the user categories to Java EE roles. Click **Yes** and see Section 9.2.2, "How to Map a Producer's Declared User Categories to an Application's Defined Java EE Security Roles." Click **No** to decline this opportunity and complete the registration process.

9.2.2 How to Map a Producer's Declared User Categories to an Application's Defined Java EE Security Roles

The user categories the producer declares come from the portlets it contains. For example, if the producer contains one or more JSR 168 portlets created with the Standards-based Java Portlet (JSR 168) Wizard, then any security roles added during portlet creation are included in the user categories the producer declares. Java EE Security Roles can be specified through the custom WebCenter application's `web.xml` file properties.

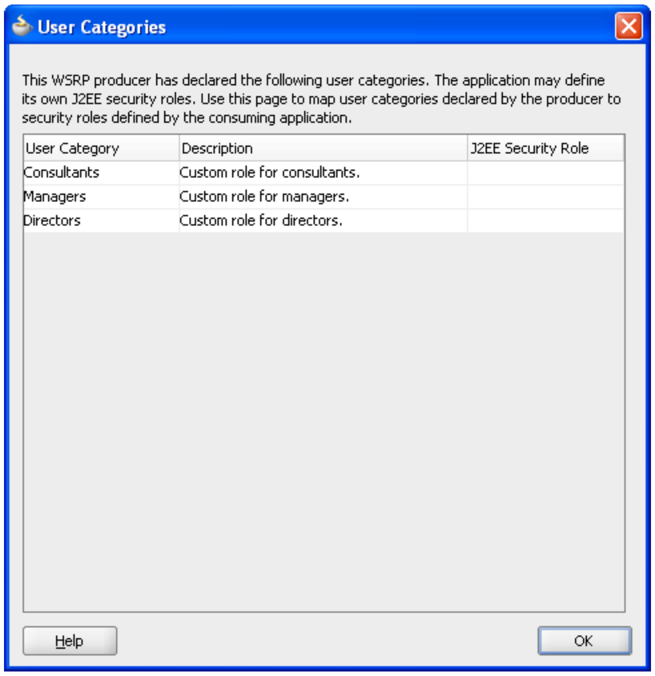
This procedure continues forward from Section 9.2.1, "How to Register a WSRP Portlet Producer."

To map producer-declared user categories with application-defined Java EE security roles:

1. After clicking **Finish** in the Register WSRP Portlet Producer wizard, click **Yes** in the resulting dialog.
2. In the User Categories dialog (Figure 9–6), for each **User Category**, click the corresponding field in the **J2EE Security Role** column.

   The User Categories dialog is also accessible when you edit producer registration settings. For more information see Section 9.2.4, "How to Edit Portlet Producer Registration Settings."

Figure 9–6 The User Categories Dialog



This WSRP producer has declared the following user categories. The application may define its own J2EE security roles. Use this page to map user categories declared by the producer to security roles defined by the consuming application.

User Category	Description	J2EE Security Role
Consultants	Custom role for consultants.	
Managers	Custom role for managers.	
Directors	Custom role for directors.	

3. From the resulting list, select the security role to map to the producer user category.
4. Click **OK** when all user categories are mapped.

9.2.3 How to Register an Oracle PDK-Java Portlet Producer

When you register a PDK-Java portlet producer, you provide basic information that describes the producer's operational parameters. This information is used by the portlet-consuming application to communicate with the producer and with the portlets through the producer.

When registration is successful, the newly registered producer is displayed in JDeveloper either in the Application Resources panel of the Application Navigator, or in the Resource Palette, depending on where you created the connection. You can then select portlets from the producer for placement on your application (`.jspx`) page.

Note: In the Register Oracle PDK-Java Portlet Producer wizard, if you click **Cancel** after you have clicked **Finish,** the registration is not canceled.

To register a PDK-Java portlet producer:

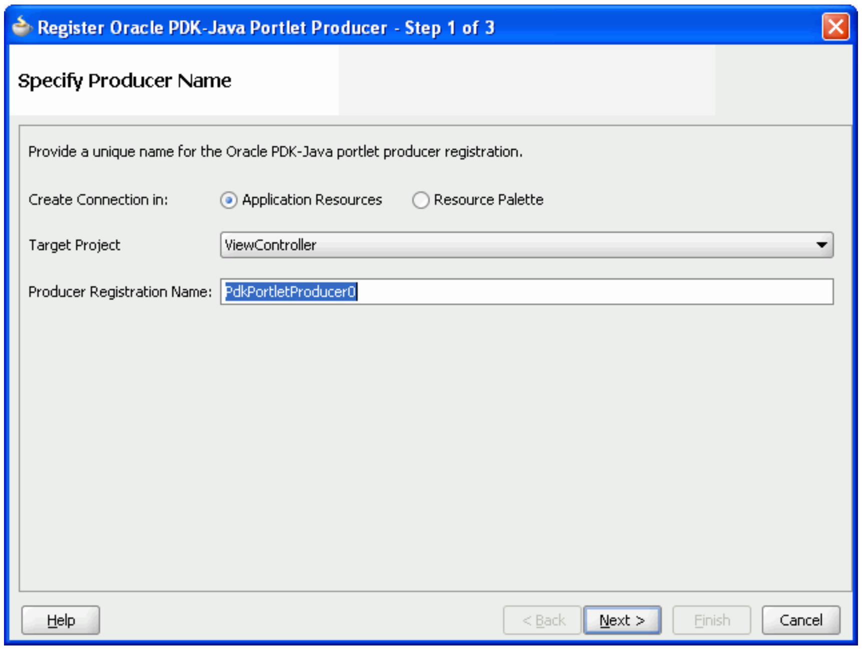
1. In the Application Resources panel of the Application Navigator, right-click **Connections,** choose **New Connection** and then choose **Oracle PDK-Java Producer.**

   For other methods of invoking the wizard, such as from the Resource Palette, see Section 3.9, "Accessing Connection Wizards."

2. In the Specify Producer Name page (Figure 9–7) of the Register Oracle PDK-Java Portlet Producer wizard, the **Create Connection in** option is set depending on how you accessed the wizard. The default selection is **Application Resources** if

you invoked the wizard from an application, and **Resource Palette** if you invoked the wizard from the Resource Palette. You can change this option at this point.

Figure 9–7 The Specify Producer Name Page (PDK-Java Producer)


Register Oracle PDK-Java Portlet Producer - Step 1 of 3

Specify Producer Name

Provide a unique name for the Oracle PDK-Java portlet producer registration.

Create Connection in: [✓ Application Resources] Resource Palette

Target Project: ViewController

Producer Registration Name: PdkPortletProducer0

3. From the **Target Project** list, select the project to be configured for the PDK-Java producer connection.

   This should be the same project as the one in which you intend to consume the portlets.

   You can change this option only if you invoked the wizard from the Application Navigator.

4. In the **Producer Registration Name** field, enter a name for the producer registration that is unique among all connections and then click **Next.**

5. In the Specify Connection Details page (Figure 9–8), in the **URL Endpoint** field, enter the producer's URL using the following syntax:

   ```
   http://host:port/context-root/providers
   ```

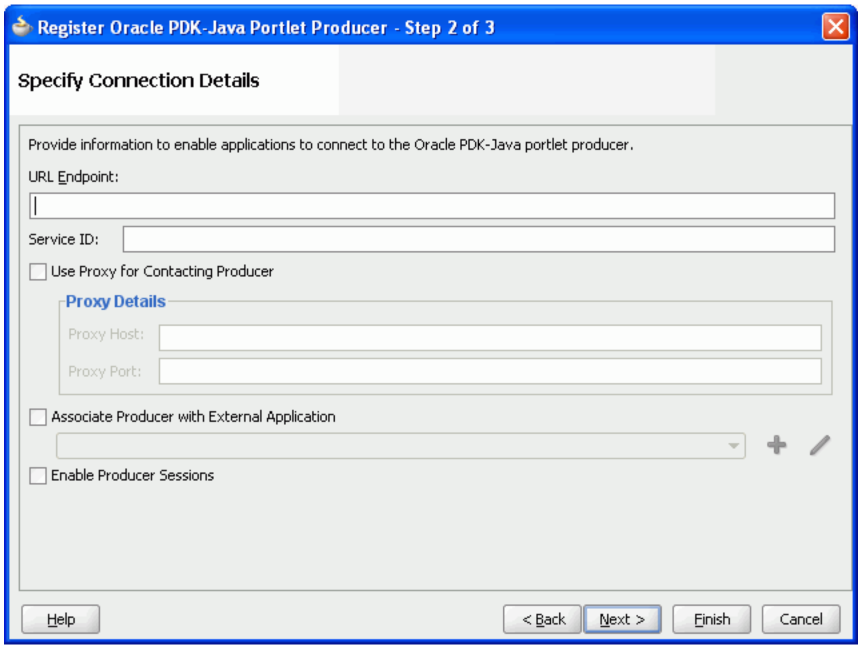
   Where:

   - `host` is the server to which your producer has been deployed.
   - `port` is the port to which the server is listening for HTTP requests.
   - `context-root` is the Web application's context root.
   - `providers` is static text. The text entered here depends on how the producer is deployed.

   For example:

   ```
   http://myhost.example.com:7101/myEnterprisePortlets/providers
   ```

Figure 9–8 The Specify Connection Details Page (PDK-Java Producer)


Register Oracle PDK-Java Portlet Producer - Step 2 of 3
Specify Connection Details
Help
< Back
Next >
Finish
Cancel

Provide information to enable applications to connect to the Oracle PDK-Java portlet producer.

URL Endpoint:

Service ID:

Use Proxy for Contacting Producer

Proxy Details

Proxy Host:

Proxy Port:

Associate Producer with External Application

Enable Producer Sessions

6. In the **Service ID** field, enter the unique identifier for this producer.

   PDK-Java enables you to deploy multiple producers under a single adapter servlet. The producers are identified by their unique service IDs. A service ID is required only when a service ID or producer name is not appended to the URL endpoint. For example the following URL endpoint requires the service ID, `sample`:

   ```
   http://myhost:7101/myEnterprisePortlets/providers
   ```

   However, the following URL endpoint, does not require a service ID:

   ```
   http://myhost:7101/myEnterprisePortlets/providers/sample
   ```

7. If the application and the producer are separated by a firewall, an HTTP proxy is needed for communication between the application and the producer. If this is the case for your application, select **Use Proxy for Contacting Producer** and specify the URL and port number of the proxy.

   Note: The proxy fields in this step default to the proxy preferences set in JDeveloper Preferences (from the main menu, choose **Tools > Preferences,** and then select **Web Browser and Proxy.**)

8. Select **Associate Producer with External Application**, then select the application, if this producer must provide authentication to an external application.

   For more information, see Section 24.2.3, "Managing External Applications."

   This option is available only if you invoked the wizard from the Application Navigator, as external applications are scoped to individual applications.

9. Select **Enable Producer Sessions** to enable sessions between the producer and the consuming application.

When sessions are enabled, the server maintains session-specific information, such as user name. Message authentication uses sessions, so if the shared key is set, then this option should also be selected.

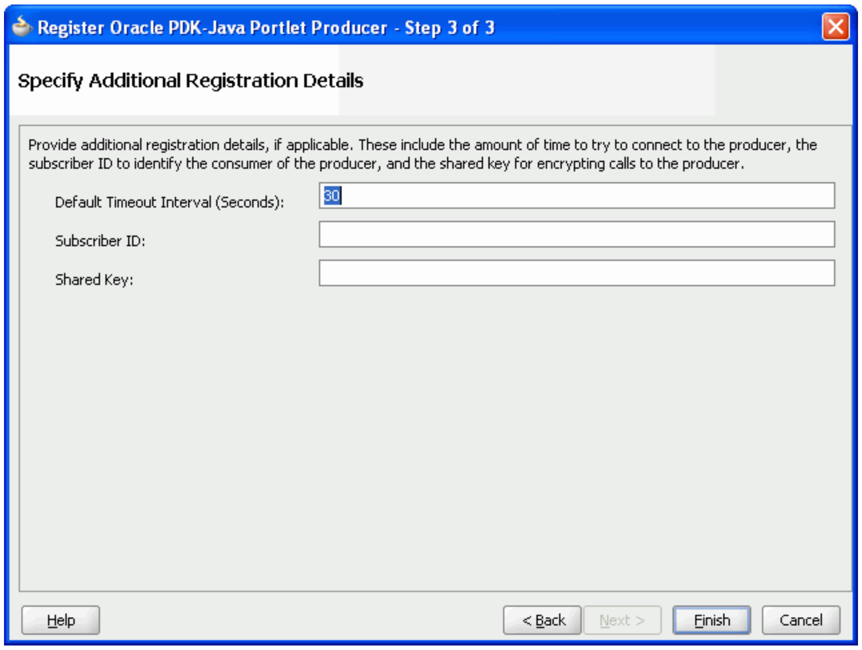
For sessionless communication between the producer and the server, deselect this option.

10. At this point in the wizard, you can click **Finish** to complete the registration, using the default values for all remaining steps.

    To provide additional details, click **Next** and follow the remaining steps.

11. In the Specify Additional Details page (Figure 9–9), in the **Default Timeout Interval (Seconds)** field, enter the number of seconds to wait for the producer to respond during design time operations.

Figure 9–9 The Specify Additional Registration Details Page (PDK-Java Producer)



Register Oracle PDK-Java Portlet Producer - Step 3 of 3

Specify Additional Registration Details

Provide additional registration details, if applicable. These include the amount of time to try to connect to the producer, the subscriber ID to identify the consumer of the producer, and the shared key for encrypting calls to the producer.

Default Timeout Interval (Seconds): 30

Subscriber ID:

Shared Key:

12. In the **Subscriber ID** field, enter a string to identify the consumer of the producer being registered.

    When a producer is registered, a call is made to the producer. During the call, the consumer passes the value for Subscriber ID to the producer. This value for Subscriber ID is also passed every time a portlet call is made. If the producer does not see the expected value for Subscriber ID, then it might reject the registration call.

13. In the **Shared Key** field, enter a shared key to use for producers that are set up to handle encryption and then click **Finish.**

    The shared key is used by the encryption algorithm to generate a message signature for message authentication. Producer registration fails if the producer is set up with a shared key and you enter an incorrect shared key here. The shared key can contain between 10 and 20 alphanumeric characters.

9.2.4 How to Edit Portlet Producer Registration Settings

Both the WSRP and PDK-Java portlet producer registration wizards enable you to access and revise many of the values you entered when you registered the producer.

To edit portlet producer registration settings:

1. Navigate to the producer in the Application Resources panel of the Application Navigator or in the Resource Palette.
2. Right-click the producer to edit, and choose **Properties.**
3. Edit the producer registration properties as required, clicking **Next** to step through the pages of the wizard.

   You can also go directly to a specific step by clicking the links in the navigation panel on the left side of the wizard.

   - For information about WSRP Producer settings, see Section 9.2.1, "How to Register a WSRP Portlet Producer."
   - For information about Oracle PDK-Java Producer settings, see Section 9.2.3, "How to Register an Oracle PDK-Java Portlet Producer."

   You cannot edit the **Producer Registration Name.**
4. When you have changed all the necessary settings, click **Finish.**
5. Editing some properties, such as the **Endpoint URL,** requires a producer refresh. When you make such a change, a dialog displays allowing you to refresh the producer immediately, save the changes but do not refresh the producer, or return to the edit producer registration wizard. Click **Yes, No,** or **Cancel** as appropriate.

   Note: While you can edit the value of the **URL Endpoint** field, for WSRP producers only update the host name, port, or IP address. Do not point to a new producer. Switching from one producer to another (even when the portlets are identical) is not supported by the WSRP specification.

6. Click **OK** when the producer has been successfully refreshed, if necessary.
7. Once you have completed your edits, consider testing the producer connection to be sure the connection information is valid. For more information, see Section 9.2.5, "How to Test a Portlet Producer Connection."

9.2.5 How to Test a Portlet Producer Connection

The connection testing feature provides a means of testing the validity of a portlet producer connection.

To test a portlet producer connection:

1. Navigate to the producer in the Application Resources panel of the Application Navigator or in the Resource Palette.
2. Right-click the producer to test, and choose **Test Producer Connection.**
3. A progress bar appears while the test is underway. A success or failure dialog displays when the test is complete. Click **OK** to close this dialog.

   If the failure dialog displays, consider editing the producer registration details and retesting the producer connection. Additionally, ensure that the producer is available. For example, if the producer is provided through the Integrated WLS, ensure that the Integrated WLS is running, and then retest the connection.

9.2.6 How to Refresh a Portlet Producer

When you refresh a portlet producer, the portlets from that producer are also refreshed. Newly added portlets and any updates to existing portlets become available to any applications that are consuming portlets from this producer.

Tip: When a portlet is removed from a producer, be sure to manually delete the portlet from all application pages on which it has been placed. For more information, see Section 9.6, "Deleting Portlets from Application Pages."

To refresh a portlet producer:

1. Navigate to the producer in the Application Resources panel of the Application Navigator or in the Resource Palette.
2. Right-click the producer to refresh, and choose **Refresh Producer Registration.**
3. In the Portlet Producer Refresh dialog, click **Yes.**

9.2.7 How to Delete a Portlet Producer

If you no longer want to use a particular producer with your application, you can delete the producer. For information about deleting portlets and relevant page variables, see Section 9.6, "Deleting Portlets from Application Pages."

To delete a portlet producer:

1. Navigate to the producer in the Application Resources panel of the Application Navigator or in the Resource Palette.
2. Right-click the producer to delete, and choose **Delete.**
3. In the Delete Confirmation dialog, click **Yes.**

9.3 Adding Portlets to a Page

Placing a portlet on a custom WebCenter application page is a simple matter of dragging the portlet from the Application Resources panel or Resource Palette and dropping it on the page.

Before You Begin

Before you can place a portlet on a page, there are a few preparatory steps you must perform before you can take this simple action. These include:

1. Creating a custom WebCenter application. For more information, see Section 3.2, "Creating a WebCenter Application."
2. Creating an application page. For more information, see Section 3.3, "Creating WebCenter Application-Enabled Pages."
3. Registering the portlet's producer with the application. For more information, see Section 9.2, "Registering Portlet Producers with a Custom WebCenter Application."
4. Some of the portlets you plan to consume may come from applications that handle their own authentication. In such cases, you must register the application as an

external application and identify it to the portlet producer that provides it. For more information, see Chapter 24, "Securing Your WebCenter Application."

5. Some of the portlets you plan to consume may come from producers that are Secure Sockets Layer (SSL) enabled. When you try to access an SSL-enabled producer, you may be presented with a Security Alert dialog, prompting you to view the producer's certificate and add it to the list of trusted certificates. The Security Alert dialog displays only if the producer uses a security certificate issued by a certificate authority that is not widely accepted. To consume portlets from such a producer, you must first add the producer's security certificate to the keystore. For more information, see Section 24.9, "Registering Custom Certificates with the Keystore."

9.3.1 How to Add a Portlet to a Page

You can add a portlet to a page by dragging and dropping it from the Application Resources panel of the Application Navigator or from the Resource Palette.

To add a portlet to a page:

1. In the Application Navigator, open the application that contains the page (`.jspx` file) to which you want to add the portlet.
2. Expand the project that contains the page.
3. Locate the page, right-click it, and then choose **Open.**

   Tip: You can also double-click the page to open it.

4. Under the **Connections** node in the Application Resources panel of the Application Navigator, or in the Resource Palette:
   - If the producer is a WSRP producer, expand the **WSRP Producer** node.
   - If the producer is a PDK-Java producer, expand the **Oracle PDK-Java Producer** node.
5. Expand the node for the portlet producer that contains the portlet to add to the page.

   Under the selected producer, all portlets contained by that producer are listed.
6. Drag the portlet from the producer node directly onto the page.

   You should drag the portlet onto a form on the page. If you do not, a dialog displays prompting you to create a form to contain the portlet. Select:
   - **ADF Faces - Form** if the page contains rich client components. This adds an `af:form` component to the page.
   - **JSF HTML - Form** if the page contains HTML components (for example, if it is an upgraded 10.1.3.2 page). This adds an `h:form` or `tr:form` component to the page, depending on the surrounding document tag.

   Do not select **HTML - Form** as this is not valid for portlets.

9.3.2 What Happens When You Add a Portlet to a Page

When you add a portlet to a page, a portlet tag (`adfp:portlet` or `adfph:portlet`) is added to the page source. This is the tag that represents the portlet component. This tag includes attributes that you can edit using the Property Inspector, or in the page source, to further control the behavior and appearance of the portlet. For information about these attributes, see Section 9.4, "Setting Attribute Values for the Portlet Tag."

The type of portlet tag used is determined by the following:

- If the project is configured for rich client components alone, the `adfp:portlet` tag is used.
- If the project is configured for Trinidad components alone, the `adfph:portlet` tag is used.
- If the project is configured for both rich client and Trinidad components, a dialog displays where you can choose which portlet tag to use.
- If the project is not configured for rich client or Trinidad components, the adfp:portlet tag is used and the project is automatically configured for rich client components.

This is so that the look and feel of the portlet matches that of other components on the page. For example, if you created your page as described in Section 3.3, "Creating WebCenter Application-Enabled Pages," the page is a rich client page. In this case, the portlet is added using the `adfp:portlet` tag.

Example 9–1   A Rich Client Page Containing a Portlet

```
<?xml version='1.0' encoding='US-ASCII'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
          xmlns:h="http://java.sun.com/jsf/html"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
          xmlns:adfp="http://xmlns.oracle.com/adf/faces/portlet">
  <jsp:directive.page contentType="text/html;charset=US-ASCII"/>
  <f:view>
    <af:document>
      <af:form>
        <adfp:portlet value="#{bindings.Portlet11_1}"
                      id="portlet1"
                      renderPortletInIFrame="true"
                      partialTriggers="com1"/>
      </af:form>
    </af:document>
  </f:view>
</jsp:root>
```

If you are working with an upgraded 10.1.3.2 application or an application that contains Trinidad components, the application uses HTML components, rather than rich client components. In this case, when you drag a portlet onto a page, the `adfph:portlet` tag is used.

Example 9–2   A Trinidad Page Containing a Portlet

```
<?xml version='1.0' encoding='US-ASCII'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
          xmlns:h="http://java.sun.com/jsf/html"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
          xmlns:tr="http://myfaces.apache.org/trinidad"
          xmlns:adfph="http://xmlns.oracle.com/adf/faces/portlet/html">
  <jsp:directive.page contentType="text/html;charset=US-ASCII"/>
  <f:view>
    <tr:document title="Title 1">
      <tr:form>
```

```
            <adfph:portlet value="#{bindings.Portlet12_1}"
                           id="portlet1"/>
          </tr:form>
        </tr:document>
      </f:view>
    </jsp:root>
```

If the application page includes one or more Oracle Composer components, this may influence where the portlet is placed. For example, in the Structure panel, a portlet placed on a page with a `cust:panelCustomizable` tag, would be placed as illustrated in Example 9–3:

Example 9–3 Hierarchical Placement of the Portlet Tag

```
af:document
  af:form
    cust:panelCustomizable
       adfp:portlet
```

> **Note:** We recommend that you do not mix ADF Faces rich client components with HTML or Trinidad components on the same page. Doing so may produce unexpected results at runtime. Therefore, do not place a rich client portlet inside an Oracle Composer HTML component or an HTML portlet inside an Oracle Composer rich client component.

For information about Composer tags, see Chapter 4, "Enabling Runtime Editing of Pages Using Oracle Composer."

> **Note:** When you drop an instance of OmniPortlet onto your page, open the Property Inspector and ensure that the **AllModesSharedScreen**, under the Display Mode category, is set to `false`, the default value. Setting this property to `true` may prevent you from editing certain sections of your OmniPortlet in the OmniPortlet wizard.

9.3.3 What Happens at Runtime

Once you place a portlet on a page, right-click the page and choose **Run.** This displays the page and runs the portlet in your default browser using JDeveloper's Integrated WLS. Different portlets may require additional runtime configuration. Notably, the content of an OmniPortlet or Web Clipping portlet instance is defined at runtime. For more information about OmniPortlet, see Chapter 31, "Creating Portlets with OmniPortlet." For more information about the Web Clipping portlet, see Chapter 32, "Creating Content-Based Portlets with Web Clipping." For more information about portlets generally, see Chapter 27, "Overview of Portlets."

When running a portlet that has an Edit mode (in a custom WebCenter application, this renders as a Personalize command on the portlet's Actions menu), the Personalize option displays in the portlet's Actions menu only to authenticated users (that is, users who have logged in). Anonymous or public users do not see the option to personalize the portlet. Some form of security must be implemented for the portlet-consuming application before users can personalize their view of a portlet. If you are a developer creating portlets and pages, you may want to test your portlet's Edit mode without creating a complete security model for your application. For information about how to add security to enable testing of a portlet's Edit mode, see Section 24.8, "Configuring Basic Authentication for Testing Portlet Personalization."

Note: To be able to add portlets to your page at runtime, you must add at least one portlet to that page at design time. Adding a portlet at design time ensures that the following is added to the `<definitionFactories>` element of the `DataBindings.cpx` file:

```
<factory nameSpace="http://xmlns.oracle.com/portlet/bindings"
className="oracle.adf.model.portlet.binding.PortletBindingDefFactor
yImpl"/>
<dtfactory
className="oracle.adfdtinternal.view.faces.portlet.PortletDefinitio
nDTFactory"/>
```

This entry is required to enable consumption of portlets at runtime.

When running a portlet from a producer associated with an external application, a link to update login information is displayed. Clicking the link displays a credential provisioning page for entering external application credentials. After specifying valid credentials the portlet displays content appropriately. For more information about external applications, see Section 24.2, "Working with External Applications."

9.4 Setting Attribute Values for the Portlet Tag

In the source code view of a page, each portlet is represented by an `adfp:portlet` tag (or `adfph:portlet` tag), which includes a set of required and optional attributes. Required attributes, `value` and `portletType`, are provided automatically by the framework, and must not be altered. Optional attribute values are relevant when support for the attribute is built into the portlet. For example, you can set `isAboutModeAvailable` to `true`, but if no About mode has been defined for the portlet, then the attribute setting does not affect the portlet.

Portlets also support a set of style-related attributes, which are discussed more fully in Section 4.2.16, "How to Apply Styles to Components."

The portlet tag uses many attributes, which you can set at design time either through the JDeveloper Property Inspector or in the source code as attributes of the tag.

9.4.1 How to Set Attribute Values for the Portlet Tag Using the Property Inspector

The Property Inspector provides a quick and easy way to set attribute values for the portlet tag without having to edit the source code yourself.

To set attribute values for the portlet tag using the Property Inspector:

1. In the Application Navigator, open the application that contains the page on which the portlet appears.
2. Expand the project that contains the page.
3. Locate the page, right-click it, and then choose **Open.**

   Tip: You can also double-click the page to open it.

4. In the design view, select the portlet whose attributes you want to set.
5. In the Property Inspector, click the appropriate tab and set the desired attribute.

Repeat this step as often as required.

6. Save your changes.
7. To test the changes you have made, right-click the page and choose **Run.**

9.4.2 How to Set Attribute Values for the Portlet Tag in Source Code

If you prefer working in source code, you can set attribute values for the portlet tag directly there.

To set attribute values for the portlet tag in source code:

1. In the Application Navigator, open the application that contains the page on which the portlet appears.
2. Expand the project that contains the page.
3. Locate the page, right-click it, and then choose **Open.**

   Tip: You can also double-click the page to open it.

4. In the design view, select the portlet whose attributes you want to set.
5. Click the **Source** tab. The portlet that you selected is highlighted in the source code.
6. Make your changes directly to the source code. Example 9–4 shows an edited portlet tag.

Example 9–4 An Edited Portlet Tag

```
<adfp:portlet value="#{bindings.portlet1}"
               portletType="/oracle/adf/portlet/WsrpPortletProducer1/
              applicationPortlets/E0default_b452f828_010a_1000_8002_82235f57eaa8"
              allModesSharedScreen="true"
              isMaximizable="true"
              isMinimizable="true"/>
```

7. Save your changes.
8. To test the changes you have made, right-click the page and choose **Run.**

9.4.3 Common Attributes of the Portlet Tag

Table 9–1 describes the common attributes of the portlet tag.

Table 9–1 Common Attributes of the Portlet Tag

Attribute	Value	Description
`id`	Text string. For example: `id="newsBrief"` The value must follow a subset of the syntax allowed in HTML: ■ Must not be a zero-length String. ■ First character must be an ASCII letter (A-Z a-z) or an underscore (_). ■ Subsequent characters must be ASCII letter or digits (a-Z a-z 0-9), underscores (_), or dashes (-).	The unique identifier of the portlet. This attribute is populated with a unique value by default when you add the portlet to a page.
`title`	Text string. For example: `title="Announcements"`	The portlet title, which is displayed in the portlet header. The value specified here takes precedence over any title specified elsewhere (for example, in the portlet markup). If no value is specified here, the portlet extracts its title from the portlet markup (response). If no value is specified either here or in the portlet markup, the portlet extracts its title from the portlet definition. **Note:** Supplying a value to the `title` attribute at design time means that any change made to the title at runtime in Edit or Edit Defaults mode is ignored.

`width`	Number expressed in pixels or as a percentage of available area:
	■ For pixels, enter *n*px, for example:
	`width = 300px`
	■ For percentage, enter *n*%, for example:
	`width = 50%`
		The width of the area to allow for portlet display.
		If the actual portlet width is larger than the `width` value entered here, a scrollbar appears, provided `displayScrollBar` is set to `auto` or `true`. If `displayScrollBar` is set to `false`, and the actual portlet width exceeds the value expressed for the `width` attribute, the `width` attribute value is considered and the portlet content is truncated.

Table 9–1 (Cont.) Common Attributes of the Portlet Tag

Attribute	Value	Description
`height`	Number expressed in pixels, for example: `height = 300px`	The height of the area to allow for portlet display. If the actual portlet height is larger than the `height` value entered here, a scrollbar appears, provided `displayScrollBar` is set to `auto` or `true`. If `displayScrollBar` is set to `false`, and the actual portlet height exceeds the value expressed for the `height` attribute, the `height` attribute value is considered and the portlet content is truncated.
`icon`	URI to an image. For example: `icon="coffee.png"` In the Property Inspector, click the Property Menu icon next to the field and then choose **Edit** to locate and select the required image. The value must be an absolute URI or a URI that is resolvable relative to the current page or the application context root. The URI provided in the preceding example is stored at the application context root, therefore a full path is not required.	A URI specifying the location of an image to use as an icon, displayed to the left of the portlet title in the portlet header. This can be used to indicate the portlet's purpose, to reinforce branding, as a content indicator, or for some other reason.
`partialTriggers`	One or more component IDs. For example: `partialTriggers="_id1 _id2 componentID5"` Separate component IDs with spaces.	The IDs of the components that trigger a partial update. The portlet listens on the specified trigger components. If a trigger component receives a trigger event that causes it to update in some way, this portlet also requests to be updated.

9.4.4 Appearance Attributes of the Portlet Tag

Table 9–2 describes the appearance attributes of the portlet tag.

Table 9–2 Appearance Attributes of the Portlet Tag

Attribute	Value	Description

`expansionMode`	`minimized`
	`normal`
	Default: `normal`
		The default state of the portlet:
		■ `minimized`—The portlet's default display mode is collapsed (minimized).
		■ `normal`—The portlet's default display made is neither collapsed nor expanded to the width of the page.

`allModesSharedScreen`	`auto`
	`false`
	`true`
	Default: `auto`
		Whether a change in portlet mode renders the new mode on a new page, rather than the page on which the portlet resides.
		■ `auto`—All portlet modes are displayed inline if the remote portlet is configured, through its `oracle-portlet.xml`, to require an iframe. This ensures that Oracle Bridge portlets are displayed inline.
		■ `false`—All portlet modes, except View (JSR 168) or Show (PDK-Java), are rendered each on their own page. This is useful for portlets such as OmniPortlet and the Web Clipping portlet, which require that modes other than Show mode display on pages other than the page on which the portlet resides.
		■ `true`—All portlet modes are displayed inline. One mode is swapped out for another on the same page. In other words, this attribute enables all portlet modes to display without leaving the context of a given page.

`renderPortletInIFrame`	`auto`
	`false`
	`true`
	Default: `auto`
		Whether the portlet is rendered in an iframe:
		■ `auto`—The portlet is rendered in an iframe if:
		- parsing of the content fails
		- a form in the content contains an input of type `file`
		- the remote portlet is configured, through its `oracle-portlet.xml`, to require an iframe. This ensures that Oracle JSF Portlet Bridge portlets are displayed in an iframe.
		For more information, see Section 9.4.13, "What You May Need to Know About Iframes."
		■ `false`—The portlet is rendered inline. HTML markup from a portlet that is not rendered in an iframe may interfere with other components on the Oracle ADF page.
		■ `true`—The portlet is rendered in an iframe.

Table 9–2 (Cont.) Appearance Attributes of the Portlet Tag

Attribute	Value	Description

`displayScrollBar`	`auto`
	`false`
	`true`
	Default: `auto`
		Whether a scroll bar is displayed:
		■ `auto`—Render a scroll bar if the portlet content does not fit the `width` and `height` specified.
		■ `false`—Never render a scroll bar. If the portlet content does not fit the `height` and `width` specified, the portlet renders in its actual size.
		■ `true`—Always render a scroll bar.

`displayHeader`	`false`
	`true`
	Default: `true`
		Whether the portlet header is displayed:
		■ `false`—The portlet header is not displayed. Icons and links normally displayed in the header are hidden. If `isSeededInteractionAvailable` is set to `true`, users can access portlet menus and icons by rolling the mouse over the portlet. A fade-in/fade-out toolbar appears, from which users can select Actions menu options.
		■ `true`—The portlet header is displayed. Consequently, header-based icons and links are displayed.

`displayShadow`	`false`
	`true`
	Default: `true`
		Whether to display a shadow decoration around the portlet:
		■ `false`—Do not display a shadow decoration.
		■ `true`—Display a shadow decoration.

`rendered`	`false`
	`true`
	Default: `true`
		Whether the portlet is rendered.
		■ `false`—Do not render the portlet. No output is rendered.
		■ `true`—Render the portlet. This is the recommended setting. Setting this attribute to `false` causes problems when you run the page.

`background`	`dark`
	`light`
	`medium`
	Default: `medium`
		The style selector to apply to the skin used by the portlet:
		■ `dark`—Apply the dark style selector to the skin.
		■ `light`—Apply the light style selector to the skin.
		■ `medium`—Apply the medium style selector to the skin.
		This provides a way for you to apply a different look and feel to each portlet on an page.

`shortDesc`	Text string. For example: `shortDesc="Portlet for entering display text in place."`	A short description of the portlet.

Table 9–2 (Cont.) Appearance Attributes of the Portlet Tag

Attribute	Value	Description

`displayActions`	`always`
	`onHover`
	Default: `always`
		Whether seeded interactions for the portlet are shown:
		▪ `always`—Always show seeded interactions.
		▪ `onHover`—Show seeded interactions when users move the mouse over the portlet.

`showMoveAction`	`menu`
	`none`
	Default: `menu`
		Whether to display the **Move** command in the portlet's **Action** menu:
		▪ `menu`—Display the **Move** command on the portlet's **Action** menu.
		▪ `none`—Do not display the **Move** command.
		There is a difference in the way that the **Move** command behaves at design time and at runtime. For more information, see Section 9.4.12, "What You May Need to Know About Maximize, Minimize, Restore, and Move."

`showRemoveAction`	`menu`
	`none`
	Default: `menu`
		Whether to display the Remove icon on the portlet chrome:
		▪ `menu`—Display the **Remove** command on the portlet's **Action** menu.
		▪ `none`—Do not display the Remove icon.
		There is a difference in the way that the Remove icon behaves at design time and at runtime. For more information, see Section 9.4.12, "What You May Need to Know About Maximize, Minimize, Restore, and Move."
		Note: This attribute is available only for the `adfp:portlet` tag and not for the `adfph:portlet` tag.

`showResizer`	`always`
	`never`
	Default: `always`
		Whether to display the resize handle at the bottom right corner of the portlet.
		▪ `always`—Always display the resize handle.
		▪ `never`—Never display the resize handle.
		Note: This attribute is available only for the `adfp:portlet` tag and not for the `adfph:portlet` tag.

`showMinimizeAction`	`chrome`
	`none`
	Default: `chrome`
		Whether to display the Minimize icon on the portlet chrome:
		▪ `chrome`—Display the Minimize icon on the portlet chrome.
		▪ `none`—Do not display the Minimize icon.
		There is a difference in the way that the Minimize icon behaves at design time and at runtime. For more information, see Section 9.4.12, "What You May Need to Know About Maximize, Minimize, Restore, and Move."

9.4.5 Behavior Attributes of the Portlet Tag

Table 9–3 describes the behavior attributes of the portlet tag.

Table 9–3 Behavior Attributes of Portlet Tag

Attribute	Value	Description
`partialTriggers`	One or more component IDs. For example: `partialTriggers="_id1 _id2 componentID5"` Separate component IDs with spaces.	The IDs of the components that trigger a partial update. The portlet listens on the specified trigger components. If a trigger component receives a trigger event that causes it to update in some way, this portlet also requests to be updated.

`submitUrlParamters`	`false`
	`true`
	Default: `false`
		Whether parameters in portlet links that point to the page on which the portlet is placed are made available to the page:
		■ `false`—Parameters are not made available to the page. Rather, they are available only inside the portlet initiating the request.
		■ `true`—Parameters available on the container page.

9.4.6 Portlet Modes Attributes of the Portlet Tag

Portlet Modes attributes control the rendering of mode-switching UI actions, such as entering edit mode. The ability to render a portlet in a particular mode depends on the modes supported by the portlet and the user authorization. For example, if the `isCustomizeModeAvailable` attribute is set to `true`, but the action is not supported in the portlet, then the attribute setting does not affect the portlet.

Portlet Modes attributes, described in Table 9–4, are value binding expressions that evaluate to `true` or `false`:

- `true` means the portlet is allowed to render in the named mode.
- `false` means the portlet is not allowed to render in the named mode.

Table 9–4 Portlet Modes Attributes of the Portlet Tag

Attribute	Value	Description

`isAboutModeAvailable`	`false`
	`true`
	Default: `true`
		Whether to render an **About** command on the portlet's **Actions** menu.
		Users choose **About** to invoke the portlet's About mode.

`isConfigModeAvailable`	`false`
	`true`
	Default: `true`
		Whether to render a **Configure** command on a JSR 168 portlet's **Actions** menu.
		Users choose **Configure** to open the portlet's Configuration settings.

`isCustomizeModeAvailable`	`false`
	`true`
	Default: `true`
		Whether to render a **Customize** command on the portlet's **Actions** menu.
		Site administrators choose **Customize** to edit a portlet's default personalization data.

Table 9–4 (Cont.) Portlet Modes Attributes of the Portlet Tag

Attribute	Value	Description

`isDetailModeAvailable`	`false`
	`true`
	Default: `true`
		Whether to render a **Details** command on a PDK-Java portlet's **Actions** menu.
		Users choose **Details** to open the portlet in Full Screen mode.

`isHelpModeAvailable`	`false`
	`true`
	Default: `true`
		Whether to render a **Help** command on the portlet's **Actions** menu.
		Users choose **Help** to open the portlet's Help page.

`isPrintModeAvailable`	`false`
	`true`
	Default: `true`
		Whether to render a **Print** command on a JSR 168 portlet's **Actions** menu.
		Users choose **Print** to displays a printer-friendly version of the portlet.

`isNormalModeAvailable`	`false`
	`true`
	Default: `true`
		Whether to render a **Refresh** command on the portlet's **Actions** menu.
		Users choose **Refresh** to redraw the portlet independent of any other content on the page (also known as a partial-page refresh).

`isPersonalizeModeAvailable`	`false`
	`true`
	Default: `true`
		Whether to render a **Personalize** command on the portlet's **Actions** menu.
		Users choose **Personalize** to alter their personal view of the portlet. This mode is equivalent to the Edit mode selection in the Standards-based Java Portlet (JSR168) Wizard.
		The **Personalize** command displays on the **Actions** menu only to authenticated users (that is, users who are logged in). It does not display to public or unauthenticated users. You must implement some form of application security for users to be able to personalize their portlet views.
		If you are a developer creating portlets, and you want to test the Personalize mode without creating a complete security model for your application, then see Section 24.8, "Configuring Basic Authentication for Testing Portlet Personalization."
		Note: A typical personalization setting is Portlet Title. You can set Portlet Title at design time, by providing a value for the `title` attribute. Consider however that supplying a value to the `title` attribute at design time prevents personalization and customization of the portlet title at runtime.

`isPreviewModeAvailable`	`false`
	`true`
	Default: `false`
		Whether to enable previewing of portlet content.
		This mode has no particular application in custom WebCenter applications, but it is used in Oracle Portal's Portlet Repository, where it renders as a magnifying glass icon, which users click to preview a portlet.

9.4.7 Style Attributes of the Portlet Tag

Table 9–5 describes the style attributes of the portlet tag.

Table 9–5 Style Attributes of the Portlet Tag

Attribute	Value	Description
`contentStyle`	One or more CSS styles. These should be in compliance with, at least, CSS 2.0 and take the following format: `contentStyle="color:rgb(255,0,255); font-family:Arial Helvetica Geneva sans-serif;font-size:large;"`	The CSS style to apply to the portlet content. Values entered here take precedence over styles specified in the `inlineStyle` attribute and those included in a CSS or skin on the specific portlet instance. For more information, see Understanding contentStyle and inlineStyle Properties.
`inlineStyle`	One or more CSS styles. These should be in compliance with, at least, CSS 2.0 and take the following format: `inlineStyle="color:rgb(255,0,255); font-family:Arial Helvetica Geneva sans-serif;font-size:large;"`	The CSS style to apply to the whole portlet. Values entered here take precedence over styles included in a CSS or skin on the specific portlet instance. For more information, see Understanding contentStyle and inlineStyle Properties.

9.4.8 Binding Attributes of the Portlet Tag

Table 9–6 describes the binding attributes of the portlet tag.

Table 9–6 Binding Attributes of the Portlet Tag

Attribute	Value	Description
`binding`	Name of a managed bean. For example: `binding="#{frameActionsBean.Binding}"` In the Property Inspector, click the Property Menu icon next to the field and then choose **Edit** to select a managed bean and specify the relevant managed bean property.	A binding reference to store the component instance. The binding reference binds an instance of the portlet to a managed bean property. Managed beans are any JavaBeans used by the application that are registered in the JSF `faces-config.xml` file.

9.4.9 Customization Attributes of the Portlet Tag

Table 9–7 describes the customization attributes of the portlet tag.

Table 9–7 Customization Attributes of the Portlet Tag

Attribute	Value	Description

`customizationAllowed`	`false`
	`true`
	Default: `true`
		Whether design time customizations of the portlet tag are allowed on this portlet.

`customizationAllowedBy`	Text string	The roles for which design time customizations are allowed. This enables you to allow customizations, but restrict who can actually perform them.

9.4.10 Annotations Attributes of the Portlet Tag

Table 9–8 describes the annotations attributes of the portlet tag.

Table 9–8 Annotations Attributes of the Portlet Tag

Attribute	Value	Description
`Title`	Text string	The title of the portlet.
`Creator`	Text string	The entity primarily responsible for creating the portlet.
`Subject`	Text string	The topic of the portlet. Typically, the subject is represented using keywords, key phrases, or classification codes.
`Description`	Text string	The purpose of the portlet.
`Publisher`	Text string	The entity responsible for making the portlet available.
`Contributor`	Text string	The entity responsible for making contributions to the portlet. Examples of a contributor include a person, an organization, or a service.
`Date`	Text string	The date of portlet creation.
`Type`	Text string	The type of portlet.
`Format`	Text string	The file format, physical medium, or dimensions of the portlet. Examples of dimensions include size and duration.
`Identifier`	Text string	An unambiguous reference to the portlet.
`Source`	Text string	The related portlet from which the described portlet is derived.
`Language`	Text string	The language of the portlet.
`Relation`	Text string	A related portlet.
`Coverage`	Text string	The jurisdiction under which the portlet is relevant.
`Rights`	Text string	Information about property rights associated with the portlet, including intellectual property rights.

9.4.11 Other Attributes of the Portlet Tag

Table 9–9 describes the other attributes of the portlet tag.

Table 9–9 Other Attributes of the Portlet Tag

Attribute	Value	Description

`iframeDtd`	`loose`
	`none`
	`strict`
	Default: `loose`
		Which DTD, if any, is specified in the `doctype` declaration that is created when portlet content is rendered inside a `iframe`:
		■ `none`—No DTD is specified. This relaxes the restrictions on the HTML content being technically conformant HTML. Browsers usually handle such HTML acceptably, however, because some CSS style sheet from the ADF Faces page consuming the portlet is also imported into the `iframe` document, for that style sheet to work correctly, it may be necessary to declare the content conformant to the loose or strict DTDs.
		■ `loose`—The DTD `http://www.w3.org/TR/html/loose.dtd` is used.
		■ `strict`—The DTD `http://www.w3.org/TR/html/strict.dtd` is used

9.4.12 What You May Need to Know About Maximize, Minimize, Restore, and Move

To accommodate the needs of the development environment, the behavior of the actions Minimize, Maximize, Restore, and Move for `Show Detail Frame` and `portlet` components differs between design time and runtime. At design time, these actions persist in a given WLS session, but do not persist over sessions (*session* means the time between starting and stopping WLS). At runtime, these actions persist both during a given WLS session and across sessions.

This difference has been introduced to enable an automatic reset of an application page at design time.

If persisting across sessions is not required at runtime, then a simple modification to the application's `web.xml` file can turn it off. Go to the following parameter setting in the application's `web.xml` file (Example 9–5):

Example 9–5 Persistence Setting in the Application's web.xml File

```
<context-param>
   <param-name>oracle.adf.view.faces.CHANGE_PERSISTENCE</param-name>
   <param-value>oracle.adfinternal.view.faces.change.HybridChangeManager</param-va
lue>
</context-param>
```

Replace it with the following (Example 9–6):

Example 9–6 Turning Runtime Persistence Off in the Application's web.xml File

```
<context-param>
   <param-name>oracle.adf.view.faces.CHANGE_PERSISTENCE</param-name>
   <param-value>oracle.adf.view.faces.change.SessionChangeManager</param-value>
</context-param>
```

If security has been implemented on the application, then the Minimize, Maximize, Restore, and Move actions display only to users with Customize privileges. They do not display to users with Personalize privileges. Customize users can test the effect of these actions by following these steps at design time:

1. Run the application page using JDeveloper's Integrated WLS.
2. Log in as the administrator.
3. Maximize a portlet. Move portlets around. Make whatever changes you want using the relevant actions commands.
4. Log out, then log in as a user and check the effects of your actions.

9.4.13 What You May Need to Know About Iframes

Placing a portlet inline on a page provides a better user experience as compared to placing it in an iframe (or inline frame). However, at times, it may be required to include the portlet markup inside an iframe. Setting the `renderPortletInIFrame` attribute to `auto` causes Oracle WebCenter Framework to render the portlet in an iframe if:

- The portlet is a JSF Portlet.

  When you add a JSF Portlet to your page, `renderPortletInIFrame` is set to `true` by default as ADF pages within portlets are too complex to render inline due to Javascript issues.

- The portlet contains a file upload element.
- The parser throws an exception as it is not able to parse the markup.
- You have specified that the portlet must be rendered in an iframe.

  To ensure that the portlet is always included in an iframe, open the `oracle-portlet.xml` file and in the `<portlet>` section, set the `<prefer-iframe>` element to `true`.

> **Note:** If you render a portlet within an iframe, then manipulating `window.location` may give unexpected results. If your portlet uses `window.location`, then you should ensure that your JavaScript is robust enough to handle the case where the portlet renders itself inside of an iframe.

Using iframes also has accessibility implications. For more information, see Section 1.3, "Accessibility Features."

9.5 Copying Portlets

When you copy portlets, the portlets and their copies must reside within the same application. For example, you can copy a portlet from one page in an application to another page in the same project in that application, or from one place on a page to another place on the same page. You can also copy a portlet from one project to another project in the same application, if the target project is configured for consuming portlets. The copies are references to the same portlet instance, so customizations or personalizations made to any instance of the portlet (original or copy) affect all the other instances.

Copying a portlet is more than a matter of copying and pasting the portlet view tag. It involves copying portlet-related entries from the application page's source. It may also involve copying portlet-related entries from the page definition file and removing duplicate portlet binding information or creating a new method in the copied portlet's binding bean.

When a portlet is copied, the target page must be an Oracle ADF Faces page. Any preexisting code on the target page must reflect that. This is quite easy to accomplish. When JDeveloper creates a new JSF page, it contains pure JSF tags. The first time you drop an Oracle ADF Faces component onto the page, tags are automatically updated to be Oracle ADF Faces tags. For example, an `<html>` tag becomes `<afh:html>`, `<head>` and `<title="title">` tags become `<afh:head title="title">`, and so on. Therefore, a simple way to ensure the conversion of the target page to an Oracle ADF Faces page is to place any Oracle ADF Faces component on the target page. This performs any required code conversion for you automatically.

This section describes how to copy portlets from one application page to another and how to copy a portlet from one part of a page to another part of the same page.

9.5.1 How to Copy a Portlet and Place it on the Same Page

Because all of the page's resources are available to both portlet instances when you copy a portlet to the same page, there is no need to copy portlet-related information from the page's Page Definition file. It is just a matter of copying and pasting the portlet's view tag, and assigning a unique identifier to the copy.

To copy and place a portlet on the same page:

1. In the Application Navigator, open the application that contains the page on which the portlet to copy appears.
2. Expand the project that contains the page.
3. Locate the page, right-click it, and then choose **Open.**

   Tip: You can also double-click the page to open it.

4. In the design view, select the portlet to copy.
5. Click the **Source** tab. The portlet that you selected is highlighted in the source code.
6. Copy the portlet tag (Example 9–7).

Example 9–7 Code Fragment to be Copied When Copying a Portlet

```
<f:view>
   <afh:html binding="#{backing_portlet_page.html1}" id="html1">
      <afh:head title="portlet_page" binding="#{backing_portlet_page.head1}"
         id="head1">
         <meta http-equiv="Content-Type"
            content="text/html;charset=windows-1252"/>
      </afh:head>
      <afh:body binding="#{backing_portlet_page.body1}" id="body1">
         <h:form binding="#{backing_portlet_page.form1}" id="form1">
            <adfp:portlet value="#{bindings.portlet1}"
               portletType="/oracle/adf/portlet/
               pdksampleproducer_1153245807295/applicationPortlets/
               Portlet2_82d49b79_010c_1000_8006_82235ffc4e2b"
               binding="#{backing_portlet_page.portlet1}"
               id="portlet1"
               isCustomModesAvailable="true"/>
         </h:form>
      </afh:body>
   </afh:html>
</f:view>
```

7. Paste the copied code fragment into the desired location of the page source.
8. Provide a unique value for the copy's `id` attribute (Example 9–8).

Example 9–8 Changing the Portlet ID

```
<adfp:portlet value="#{bindings.portlet1}"
   portletType="/oracle/adf/portlet/
   pdksampleproducer_1153245807295/applicationPortlets/
   Portlet2_82d49b79_010c_1000_8006_82235ffc4e2b"
   binding="#{backing_portlet_page.portlet1}"
   id="portlet2"
   isCustomModesAvailable="true"/>
```

Note: On a given page, each portlet must have a unique ID.

9. In the page source, if the copied portlet's `adfp:portlet` tag has a binding attribute, for example:

   ```
   binding="#{backing_untitled2.portlet1}"
   ```

   Then either remove this binding, or create a new method in the binding bean by opening the managed bean class for this managed bean and defining the new method.

   For example, if portlet1 is copied, the pasted copy becomes portet2 in the managed bean class, as shown in Example 9–9.

Example 9–9 Creating a New Method for a Managed Bean in the Managed Bean Class

```
.
private PortletBase portlet2;
public void setPortlet2(PortletBase portet2)  {
     this.portlet2 = portlet2;
}
.
public PortletBase getPortlet2()  {
     return portlet2;
}
```

10. From the main menu, choose **File > Save All.**

9.5.2 How to Copy a Portlet from One Application Page to Another

When you copy a portlet from one page to another in an application, portlet-related code must also be copied from the source page's Page Definition file. This section describes the steps related to both copying from one application page to another and from one application project to another.

To copy a portlet from one application page to another:

1. In the Application Navigator, open the application that contains the page on which the portlet to copy appears.
2. Expand the project that contains the page.

3. Locate the page, right-click it, and then choose **Open.**

   Tip: You can also double-click the page to open it.

4. In the design view, select the portlet to copy.
5. Click the **Source** tab. The portlet that you selected is highlighted in the source code.
6. Copy the portlet tag (Example 9–10).

Example 9–10 Source Page Code Fragment to Be Copied When Copying a Portlet

```
<f:view>
   <afh:html binding="#{backing_portlet_page.html1}" id="html1">
      <afh:head title="portlet_page" binding="#{backing_portlet_page.head1}"
         id="head1">
         <meta http-equiv="Content-Type"
            content="text/html;charset=windows-1252"/>
      </afh:head>
      <afh:body binding="#{backing_portlet_page.body1}" id="body1">
         <h:form binding="#{backing_portlet_page.form1}" id="form1">
            <adfp:portlet value="#{bindings.portlet1}"
               portletType="/oracle/adf/portlet/
               pdksampleproducer_1153245807295/applicationPortlets/
               Portlet2_82d49b79_010c_1000_8006_82235ffc4e2b"
               binding="#{backing_portlet_page.portlet1}"
               id="portlet1"
               isCustomModesAvailable="true"/>
         </h:form>
      </afh:body>
   </afh:html>
</f:view>
```

7. Go to the application page to which to copy the portlet (the target page).

   Portlets can reside only on Oracle ADF Faces pages. If the target page does not contain Oracle ADF Faces components, then ensure that the container objects—that is, any tags the portlet tag is nested in—use Oracle ADF tags.

   If you are copying the portlet to a page in a different project, the target project must be configured for consuming portlets. To configure the project, you must register a portlet producer with the project. For information, see Section 9.2, "Registering Portlet Producers with a Custom WebCenter Application."

8. Click the **Source** tab and paste the copied code fragment into the desired location of the page source.
9. In the Application Navigator, right-click the source page (the page from which the portlet was copied), and choose **Go to Page Definition.**
10. Click the **Source** tab and copy the portlet binding from the source page's page definition file (Example 9–11).

Example 9–11 Code Fragment to Be Copied From a Page Definition File

```
<portlet id="portlet1"
  portletInstance="/oracle/adf/portlet/pdksampleproducer_
1153245/applicationPortlets/Portlet2_82d49_010c_1000_8006_82235"
  class="oracle.adf.model.portlet.binding.PortletBinding"
  xmlns="http://xmlns.oracle.com/portlet/bindings"/>
```

Note: When the portlet being copied includes parameters, be sure to include the copied portlet's portlet parameters and the page variables linked to the portlet parameters in the copy.

---

11. In the Application Navigator, right-click the target page and choose **Go to Page Definition.**

    If a page definition does not already exist for the page, you are asked if you want to create one. Click **Yes.**

12. Click the **Source** tab and paste the portlet binding you copied from the source (along with relevant portlet parameters and the page variables associated with those parameters).

    Paste the code between the `<executables>` tags. You may need to add a closing `</executables>` tag and ensure that the opening tag does not contain a slash (`/`).

13. From the main menu, choose **File > Save All.**

9.6 Deleting Portlets from Application Pages

When you delete a portlet from an application page, if the portlet had parameters, then you should also delete page variables associated with those parameters from the application page's Page Definition file.

To delete a portlet and its related page variables from a page:

1. In the Application Navigator, open the application that contains the page on which the portlet to delete appears.
2. Expand the project that contains the page.
3. Locate the page, right-click it, and then choose **Open.**

    Tip: You can also double-click the page to open it.

4. In the design view, right-click the portlet to delete and choose Delete.

    This deletes the portlet from the page and the portlet binding from the Page Definition file.

5. If the portlet included variables, in the Application Navigator, right-click the page and choose **Go to Page Definition.**
6. Click the **Source** tab and locate the page variables associated with the deleted portlet, and delete them from the page definition file.

    For example, if you deleted portlet1, you would also delete the highlighted variables in Example 9–12:

Example 9–12 Deleting Portlet-Related Page Variables from a Page Definition File

```
<?xml version="1.0" encoding="UTF-8" ?>
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
       version="10.1.3.38.97" id="untitled1PageDef"
       Package="project1.pageDefs">
   <parameters/>
   <executables>
      <variableIterator id="variables">
```

```
                <variable Name="portlet1_param1" Type="java.lang.Object"/>
                <variable Name="portlet1_param2" Type="java.lang.Object"/>
                <variable Name="portlet2_param1" Type="java.lang.Object"/>
                <variable Name="portlet2_param2" Type="java.lang.Object"/>
        </variableIterator>
        <portlet id="portlet2" portletInstance="/oracle/adf/portlet/
                PdkPortletProducer2_1154100666247/applicationPortlets/
                Portlet1_b5e49696_010c_1000_8008_8c5707ef9c4f"
                class="oracle.adf.model.portlet.binding.PortletBinding"
                xmlns="http://xmlns.oracle.com/portlet/bindings">
            <parameters>
                  <parameter name="param1" pageVariable="portlet2_param1"/>
                  <parameter name="param2" pageVariable="portlet2_param2"/>
            </parameters>
        </portlet>
    </executables>
    <bindings/>
</pageDefinition>
```

7. From the main menu, choose, select **File > Save All**.

9.7 Contextually Linking WSRP 2.0 Portlets

One way to make your custom WebCenter application more interactive is by linking related components such that their contents are synchronized based upon the context. For example, suppose you have two stock portlets on a page, one provides data about a stock's price while the other provides headline news items for a stock. Both portlets are based upon the stock ticker symbol, hence it would make sense that, when the ticker symbol is changed in the stock price portlet, the stock headlines portlet picks up that change and refreshes itself with headlines pertaining to the same ticker symbol. Table 9–10 summarizes the types of components you may tie with this type of contextual behavior.

Table 9–10 Components You Can Synchronize

Component Initiating Parameter	Can Initiate Partial Page Refresh	Component Reading Parameter Value	Comments
Oracle ADF Faces components	Yes	Oracle ADF Faces components	For more information about linking Faces components, see *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.
	Yes	WSRP portlets	Faces components, such as options or lists, can be used to change the displayed content of portlets.
	Yes	PDK-Java portlets	Faces components, such as options or lists, can be used to change the displayed content of portlets.
WSRP 2.0 portlets[1]	Yes	Oracle ADF Faces components	Standards-based portlets can pass parameters to Faces components.
	Yes	WSRP 2.0 portlets	You can pass parameters between standards-based portlets.
	Yes	PDK-Java portlets	You can pass parameters from a standards-based portlet to a PDK-Java-based portlet.

Table 9–10 (Cont.) Components You Can Synchronize

Component Initiating Parameter	Can Initiate Partial Page Refresh	Component Reading Parameter Value	Comments
PDK-Java portlets	No	Oracle ADF Faces components	PDK-Java portlets can pass parameters to Faces components.
	No	WSRP portlets	You can pass parameters from a PDK-Java portlet to a standards-based portlet.
	No	PDK-Java portlets	You can pass parameters between PDK-Java-based portlets.
Oracle ADF Faces page variable/parameter	N/A	Oracle ADF Faces components	You can set parameters at the page level and have them picked up by Faces components on the page.
	N/A	WSRP portlets	Standards-based portlets can pick up page parameters.
	N/A	PDK-Java portlets	PDK-Java portlets can pick up page parameters.

[1] Note that support for navigational parameters exists only in WSRP 2.0. WSRP 1.0 does not have support for navigational parameters.

WSRP 2.0 introduced navigational parameters, which can be with page variables to link portlets on the page. When your goal is to link two components where at least one is a WSRP 2.0 portlet, you can use navigational parameters alone.

With navigational parameters, you publish page variables, and individual components can subscribe to those page variables and reflect the changes made to those values. When you want to link more than two components, such as a task flow to two or more WSRP 2.0 portlets, you can use ADFm events for the navigational parameters. The ADFm model enables one component to accept a value, process it, and then raise an event that another component can then process. This can continue until all events in the page are raised and consumed or discarded.

Note: An example of how to use ADFm events and navigation parameters with JSF portlets is provided in Section 28.3, "Creating JSF Portlets: Example."

Before You Begin

The following examples show how to link WSRP portlets in various ways. This is illustrated using the Parameter Form Portlet and Parameter Display Portlet provided by the sample WSRP producer. The Parameter Form Portlet has three output parameters that get set when values are submitted in the form inside the portlet. The Parameter Display Portlet has three input parameters, the values of which are displayed in the portlet when they are set.

Before you begin working through the examples, you must create an application and register the sample WSRP producer.

1. Start JDeveloper.
2. From the main menu, choose **File > New.**

3. In the New Gallery, expand **General,** select **Applications** and then **WebCenter Application,** and click **OK.**
4. In the Name your application page of the Create WebCenter Application wizard, in the **Application Name** field, enter `ContextualWiringApplication` and then click **Finish.**
5. From the main menu, choose **Run > Start Server Instance.**
6. From the main menu, choose **Help > WebCenter Preconfigured Server Readme.**
7. Under Preconfigured Portlet Producers, click the **Sample Portlets** link under WSRP Portlet Producers.
8. In your browser, note the URL of the **WSRP V2 WSDL.**
9. In the Application Resources panel of the Application Navigator, right-click **Connections** and choose **New Connection** and then **WSRP Producer.**
10. In the Specify Producer Name page of the Register WSRP Portlet Producer wizard, in the **Producer Registration Name** field, enter `MyWsrpSampleProducer` and then click **Next.**
11. In the Specify Connection Details page, in the **WSDL URL** field, enter the URL of your sample WSRP producer (as noted in step 8), and then click **Next.**

    For example:

    ```
    http://127.0.0.1:7101/portletapp/portlets/wsrp2?WSDL
    ```

12. In the Specify Additional Registration Details page, click **Finish.**

If you prefer to work through these examples using your own portlets, you must ensure that you have created portlets that support parameters. For information about how to create WSRP portlets, see Section 29.2.1, "How to Create a JSR 168 Java Portlet." For more specific information about how to implement parameters in your WSRP portlets, see Section 30.1.2, "How to Implement Navigational Parameters (WSRP 2.0)."

9.7.1 How to Link Portlets With Navigational Parameters

The simplest way to link WSRP 2.0 portlets, when you simply want to link two portlets, is to use navigational parameters.

To link your WSRP 2.0 portlets using navigational parameters:

1. In the Application Navigator, right-click the **ViewController** project of the ContextualWiringApplication and choose **New.**
2. In the New Gallery, expand **Web Tier,** select **JSF** and then **JSF page,** and click **OK.**
3. In the Create JSF Page dialog, in the **File Name** field, enter `PortletsWithNavParam.jspx.`
4. Select **Create as XML Document (*.jspx)** and click **OK.**
5. In the Component Palette, select **ADF Faces** from the dropdown list.
6. From the Layout panel, drag the **Panel Splitter** component onto the page you just created.
7. In the Property Inspector, for the Panel Splitter component, click or expand the **Style** tab and then click the **Box** tab.
8. In the **Width** field, enter `600` and select **px - Pixel** from the corresponding dropdown list.

9. Click or expand the **Appearance** tab, and in the **SplitterPosition** field, enter `300`.
10. If the Integrated WLS is not already running, from the main menu, choose **Run > Start Server Instance.**
11. In the Application Resources panel of the Application Navigator, under the **Connections** node, expand **WSRP Producer** and then expand **MyWSRPSampleProducer.**

    You should have registered the sample WSRP producer. If you have registered the producer, see the "Before You Begin" section.
12. Drag the **Parameter Form Portlet** onto the **first** facet of the PortletsWithNavParam page.
13. Drag the **Parameter Display Portlet** onto the **second** facet of the page.

    Adding the portlets to the page creates six page variables: three page variables for the three portlet parameters of each portlet. To link the portlets, you must make the two portlets use the same page variables for their parameters. In this example, the Parameter Display Portlet sets three page variables to the values entered. The Parameter Display Portlet then uses the values from these same page variables to determine the values of its parameters.
14. In the Projects panel of the Application Navigator, right-click **PortletsWithNavParam.jspx** and choose **Go to Page Definition.**
15. In the Structure panel of the Application Navigator, expand the **executables** node, then **ParameterFormPortlet1_1,** then **parameters.**
16. Select each parameter in turn and, from the Property Inspector, make a note of the page variable assigned to it. This should be the same as shown in Table 9–11:

Table 9–11 Parameter Settings for the Parameter Form Portlet

Portlet Parameter	pageVariable
ora_wsrp_navigparam_Parameter1	ParameterFormPortlet1_1_ora_wsrp_navigparam_Parameter1
ora_wsrp_navigparam_Parameter2	ParameterFormPortlet1_1_ora_wsrp_navigparam_Parameter2
ora_wsrp_navigparam_Parameter3	ParameterFormPortlet1_1_ora_wsrp_navigparam_Parameter3

17. Now to make the portlet parameters for the Parameter Display Portlet point to the same page variable as the Parameter Form Portlet so that the two portlets communicate correctly.

    In the Structure panel, expand **ParameterDisplayPortlet1_1,** then **parameters.**
18. Select each parameter in turn and, in the Property Inspector, set the **pageVariable** property to the same page variables as used by the Parameter Form Portlet, instead of those created for the Parameter Display Portlet. Use Table 9–12 for guidance:

Table 9–12 Parameter Settings for the Parameter Display Portlet

Portlet Parameter	pageVariable
ora_wsrp_navigparam_Parameter1	Parameter**Form**Portlet1_1_ora_wsrp_navigparam_Parameter1
ora_wsrp_navigparam_Parameter2	Parameter**Form**Portlet1_1_ora_wsrp_navigparam_Parameter2
ora_wsrp_navigparam_Parameter3	Parameter**Form**Portlet1_1_ora_wsrp_navigparam_Parameter3

19. Next, you must ensure that the Parameter Display Portlet refreshes itself whenever the Parameter Form Portlet is updated to display the correct values.

    In the PortletsWithNavParam page, select the Parameter Form Portlet.

20. In the Property Inspector, click or expand the **Common** tab and make a note of the value in the **Id** field (this should be `portlet1`).

21. In the PortletsWithNavParam page, select the Parameter Display Portlet.

22. In the Property Inspector, click or expand the **Common** tab and in the **PartialTriggers** field, enter the ID of the Parameter Form Portlet that you noted earlier (`portlet1`).

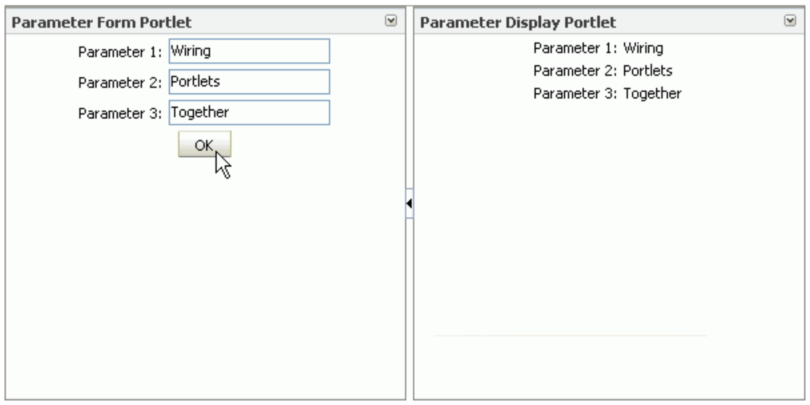
23. From the main menu, choose **File > Save All.**

24. In the Projects panel of the Application Navigator, right-click **PortletsWithNavParam.jspx** and choose **Run.**

9.7.2 What Happens at Runtime

When you run the page, you should see one portlet that enables you to change the value of the parameters (the Parameter Form Portlet) and another that displays the value of the parameters (the Parameter Display Portlet). When you change the value of a parameter in the Parameter Form Portlet and click **OK,** the corresponding parameter in the Parameter Display Portlet updates automatically to display the new value (see Figure 9–10).

Figure 9–10 Portlets Linked Using Navigational Parameters


Parameter Form Portlet

Parameter 1: Wiring
Parameter 2: Portlets
Parameter 3: Together

Parameter Display Portlet

Parameter 1: Wiring
Parameter 2: Portlets
Parameter 3: Together

9.7.3 How To Associate Navigational Parameters with ADFm Events

ADFm events provide a powerful way of linking many components, including WSRP 2.0 portlets. When a portlet is placed on a page, an event is raised by the portlet if the portlet has parameters. By subscribing to that event, many components can be updated when the parameters change.

The following procedure provides a simple example that shows how to update the parameters of a single portlet using ADFm events. For an example that shows how to update the parameters of several portlets, see Section 9.7.5, "How To Cascade Events Across Multiple Components."

To associate parameters with an ADFm event:

1. In the Application Navigator, right-click the **ViewController** project of the ContextualWiringApplication and choose **New.**

You created this application in Section 9.7.1, "How to Link Portlets With Navigational Parameters."

2. Create a JSPX page using the instructions provided in steps 1 through 13 in Section 9.7.1, "How to Link Portlets With Navigational Parameters," setting the **File Name** of the page to `PortletsWithAdfmEvents.jspx.`
3. In the Projects panel of the Application Navigator, right-click **PortletsWithAdfmEvents.jspx** and choose **Go to Page Definition.**
4. In the Structure panel of the Application Navigator, right-click **PortletsWithAdfmEventsPageDef** and choose **Edit Event Map.**
5. In the Event Map Editor dialog, click the Add a New Event Entry icon.
6. In the Add New EventMap Entry dialog, from the **Producer** list, select **ParameterFormPortlet2_1.**

   This identifies the Parameter Form Portlet as the component that triggers the event.
7. The **Event Name** field is automatically populated with the value **ParameterFormPortlet2_1_Event.**
8. From the **Consumer** list, select **ParameterDisplayPortlet2_1.**

   This identifies the Parameter Display Portlet as a component that receives values when the event is triggered.
9. In the Consumer Params table, add the parameters listed in Table 9–13 by clicking the Add Consumer Parameter icon and specifying the name and value for each parameter.

   This maps the portlet parameters in the consumer (the Parameter Display Portlet) to the payload of the event triggered by the producer (the Parameter Form Portlet).

Table 9–13 Parameter Values

Param Name (Consumer)	Param Value (Producer)
ora_wsrp_navigparam_Parameter1	${payLoad.ora_wsrp_navigparam_Parameter1}
ora_wsrp_navigparam_Parameter2	${payLoad.ora_wsrp_navigparam_Parameter2}
ora_wsrp_navigparam_Parameter3	${payLoad.ora_wsrp_navigparam_Parameter3}

10. Click **OK** to close the Add New EventMap Entry dialog, and then **OK** again to close the Event Map Editor dialog.
11. Next, you must ensure that the Parameter Display Portlet refreshes itself whenever the Parameter Form Portlet is updated to display the correct values.

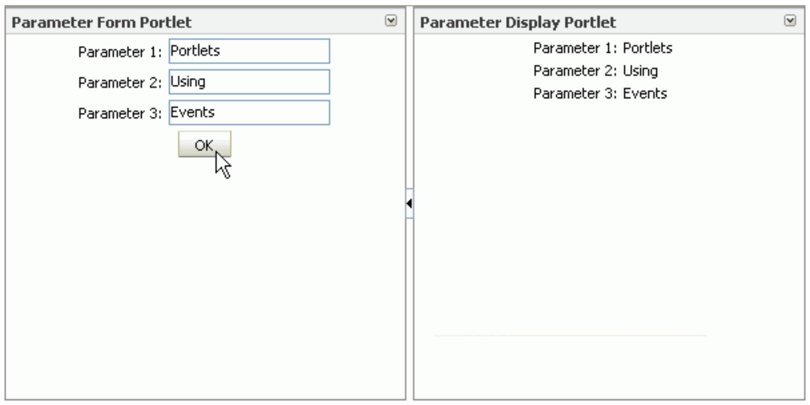
    In the PortletsWithAdfmEvents page, select the Parameter Form Portlet.
12. In the Property Inspector, click or expand the **Common** tab and make a note of the value in the **Id** field (this should be `portlet1`).
13. In the PortletsWithAdfmEvents page, select the Parameter Display Portlet.
14. In the Property Inspector, click or expand the **Common** tab and in the **PartialTriggers** field, enter the ID of the Parameter Form Portlet that you noted earlier (`portlet1`).
15. From the main menu, choose **File > Save All.**

16. In the Application Navigator, right-click **PortletsWithAdfmEvents.jspx** and choose **Run.**

9.7.4 What Happens at Runtime

When you run the page, you should see one portlet that enables you to change the value of the parameters (the Parameter Form Portlet) and another that displays the value of the parameters (the Parameter Display Portlet). When you change the value of a parameter in the Parameter Form Portlet and click **OK,** the corresponding parameter in the Parameter Display Portlet updates automatically to display the new value (see Figure 9–10).

Figure 9–11 Portlets Linked Using ADFm Events



When you run the page, you should see one portlet that enables you to change the value of the parameter and another that displays the value of the parameter. When you change the value of the parameter in one portlet and click **OK,** the other portlet should update automatically to display the new value.

9.7.5 How To Cascade Events Across Multiple Components

The main advantage of associating your navigational parameters with ADFm events is the capability to cascade the events across many components. In this way, you can link many more components than you could with just navigational parameters alone.

To cascade events across multiple components

1. In the Application Navigator, right-click the **ViewController** project of the ContextualWiringApplication and choose **New.**

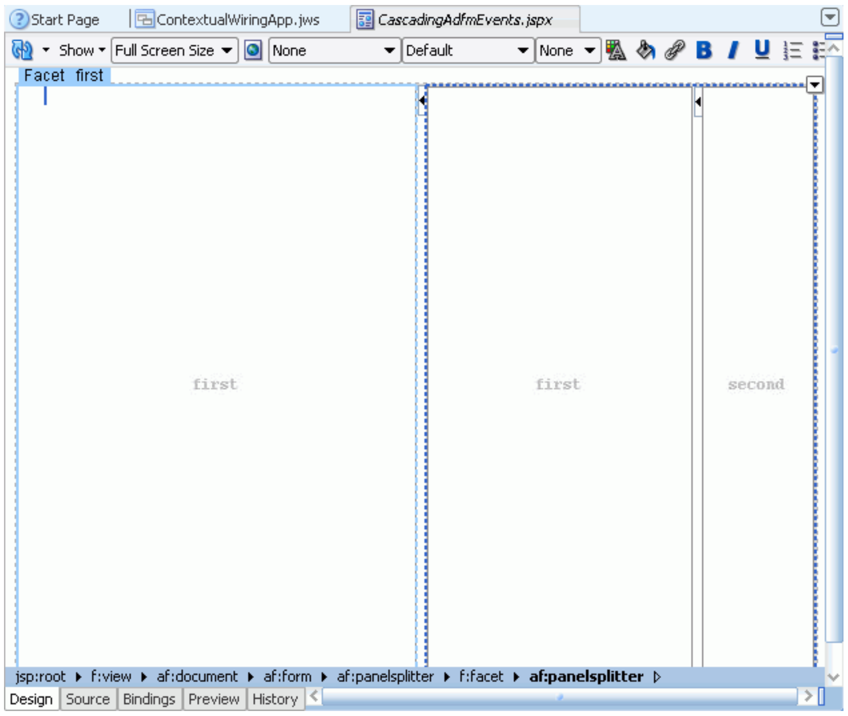
   You created this application in Section 9.7.1, "How to Link Portlets With Navigational Parameters."

2. Create a JSPX page using the instructions provided in steps 1 through 8 in Section 9.7.1, "How to Link Portlets With Navigational Parameters," setting the **File Name** of the page to `CascadingAdfmEvents.jspx` and the **SplitterPosition** to `200`.

3. In the Layout panel of the Component Palette, drag a second Panel Splitter component onto the **second** facet of the page.

   Your page should now look something like Figure 9–12:

Figure 9–12 Page Layout for Cascading ADFm Events Example



4. If the Integrated WLS is not already running, from the main menu, choose **Run > Start Server Instance.**
5. In the Application Resources panel of the Application Navigator, under the **Connections** node, expand **WSRP Producer** and then expand **MyWSRPSampleProducer.**

   You should have registered the sample WSRP producer. If you have registered the producer, see the "Before You Begin" section.
6. Drag the **Parameter Form Portlet** onto the leftmost **first** facet of the PortletsWithNavParam page.
7. Drag the **Parameter Display Portlet** onto the other **first** facet of the page.
8. Drag another **Parameter Display Portlet** onto the **second** facet of the page.
9. In the Projects panel of the Application Navigator, right-click **CascadingAdfmEvents.jspx** and choose **Go to Page Definition.**
10. In the Structure panel of the Application Navigator, right-click **CascadingAdfmEventsPageDef** and choose **Edit Event Map.**
11. In the Event Map Editor dialog, click the Add a New Event Entry icon.
12. In the Add New EventMap Entry dialog, from the **Producer** list, select **ParameterFormPortlet3_1.**

    This identifies the Parameter Form Portlet as the component that triggers the event.
13. The **Event Name** field is automatically populated with the value **ParameterFormPortlet3_1_Event.**
14. From the **Consumer** list, select **ParameterDisplayPortlet3_1.**

This identifies the first Parameter Display Portlet as a component that receives values when the event is triggered.

15. In the Consumer Params table, add the parameters listed in Table 9–14 by clicking the Add Consumer Parameter icon and specifying the name and value for each parameter.

    This maps the portlet parameters in the consumer (the Parameter Display Portlet) to the payload of the event triggered by the producer (the Parameter Form Portlet).

Table 9–14 Parameter Values

Param Name (Consumer)	Param Value (Producer)
ora_wsrp_navigparam_Parameter1	${payLoad.ora_wsrp_navigparam_Parameter1}
ora_wsrp_navigparam_Parameter2	${payLoad.ora_wsrp_navigparam_Parameter2}
ora_wsrp_navigparam_Parameter3	${payLoad.ora_wsrp_navigparam_Parameter3}

16. Click **OK** to close the Add New EventMap Entry dialog.
17. In the Event Map Editor dialog, click the Add a New Event Entry icon.
18. In the Add New EventMap Entry dialog, from the **Producer** list, select **ParameterDisplayPortlet3_1.**

    This identifies the first Parameter Display Portlet as the component that triggers the event.

19. The **Event Name** field is automatically populated with the value **ParameterDisplayPortlet3_1_Event.**
20. From the **Consumer** list, select **ParameterDisplayPortlet4_1.**

    This identifies the second Parameter Display Portlet as a component that receives values when the event is triggered.

21. In the Consumer Params table, add the parameters listed in Table 9–15 by clicking the Add Consumer Parameter icon and specifying the name and value for each parameter.

    This maps the portlet parameters in the consumer (the Parameter Display Portlet) to the payload of the event triggered by the producer (the Parameter Form Portlet).

Table 9–15 Parameter Values

Param Name (Consumer)	Param Value (Producer)
ora_wsrp_navigparam_Parameter1	${payLoad.ora_wsrp_navigparam_Parameter1}
ora_wsrp_navigparam_Parameter2	${payLoad.ora_wsrp_navigparam_Parameter2}
ora_wsrp_navigparam_Parameter3	${payLoad.ora_wsrp_navigparam_Parameter3}

22. Click **OK** to close the Add New EventMap Entry dialog, and then **OK** again to close the Event Map Editor dialog.
23. Next, you must ensure that the portlets refreshes themselves to display the correct values.

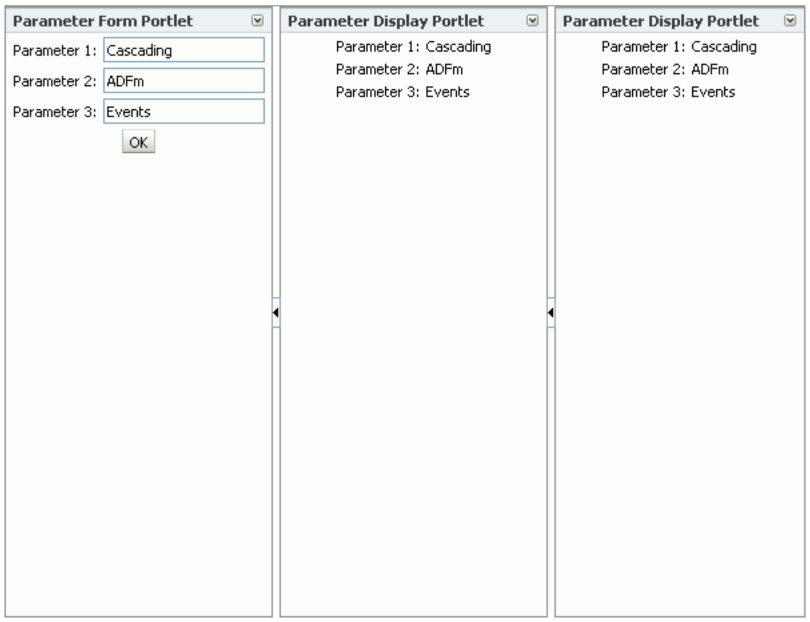
    In the CascadingAdfmEvents page, select the Parameter Form Portlet.

24. In the Property Inspector, click or expand the **Common** tab and make a note of the value in the **Id** field (this should be `portlet1`).
25. In the CascadingAdfmEvents page, select the first Parameter Display Portlet.
26. In the Property Inspector, click or expand the **Common** tab and in the **PartialTriggers** field, enter the ID of the Parameter Form Portlet that you noted earlier (`portlet1`).
27. Make a note of the value in the **Id** field (this should be `portlet2`).
28. In the CascadingAdfmEvents page, select the second Parameter Display Portlet.
29. In the Property Inspector, click or expand the **Common** tab and in the **PartialTriggers** field, enter the ID of the first Parameter Display Portlet that you noted earlier (`portlet2`).
30. From the main menu, choose **File > Save All.**
31. In the Application Navigator, right-click **CascadingAdfmEvents.jspx** and choose **Run.**

9.7.6 What Happens at Runtime

When you run the page, you should see one portlet that enables you to change the value of the parameters (the Parameter Form Portlet) and two that display the values of the parameters (the two Parameter Display Portlets). When you change the value of a parameter in the Parameter Form Portlet and click **OK,** an event is raised and the values from the parameters are sent as the payload to the first Parameter Display Portlet, which updates automatically to display the new value (see Figure 9–13). This then triggers an event in the Parameter Display Portlet, which in turn provides a payload for the second Parameter Display Portlet.

Figure 9–13 Cascading ADFm Events


Parameter Form Portlet

Parameter 1: Cascading

Parameter 2: ADFm

Parameter 3: Events

OK

Parameter Display Portlet

Parameter 1: Cascading

Parameter 2: ADFm

Parameter 3: Events

Parameter Display Portlet

Parameter 1: Cascading

Parameter 2: ADFm

Parameter 3: Events

10 Integrating with Oracle WebCenter Spaces

This chapter describes how to expose Oracle WebCenter Spaces functionality in a custom WebCenter application, using WebCenter Spaces APIs. Through these APIs, you can create group spaces, manage group space membership, retrieve group space information, and more, from your custom WebCenter applications.

The chapter also describes how to expose information from other applications, such as custom WebCenter applications in WebCenter Spaces.

This chapter includes the following sections:

Note: Custom-built task flows that implement group space APIs (described in this chapter) can be deployed on custom WebCenter applications, but you cannot deploy them to WebCenter Spaces. To find out how to build task flows for WebCenter Spaces that include calls to local group space APIs, refer to the whitepaper "Extending WebCenter Spaces" on the Oracle Technology Network.

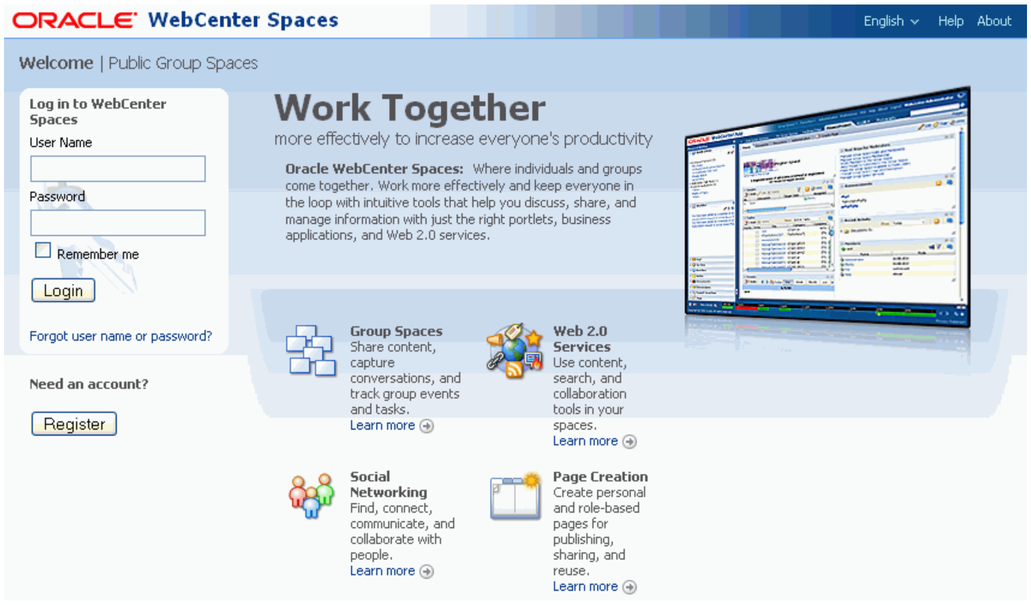
Audience

This section is intended for developers who want to expose WebCenter Spaces functionality in their custom WebCenter applications. or who are building custom applications that will be integrated with WebCenter Spaces.

10.1 Introduction to WebCenter Spaces

WebCenter Spaces is a Web-based application, built using the Oracle WebCenter Framework, WebCenter Web 2.0 services, and Oracle Composer, that offers the very latest technology for social networking, communication, collaboration, and personal productivity. Figure 10–1 shows the WebCenter Spaces home page.

Figure 10–1 Oracle WebCenter Spaces



Work Together

more effectively to increase everyone's productivity

Oracle WebCenter Spaces: Where individuals and groups come together. Work more effectively and keep everyone in the loop with intuitive tools that help you discuss, share, and manage information with just the right portlets, business applications, and Web 2.0 services.

Group Spaces Share content, capture conversations, and track group events and tasks. Learn more

Web 2.0 Services Use content, search, and collaboration tools in your spaces. Learn more

Social Networking Find, connect, communicate, and collaborate with people. Learn more

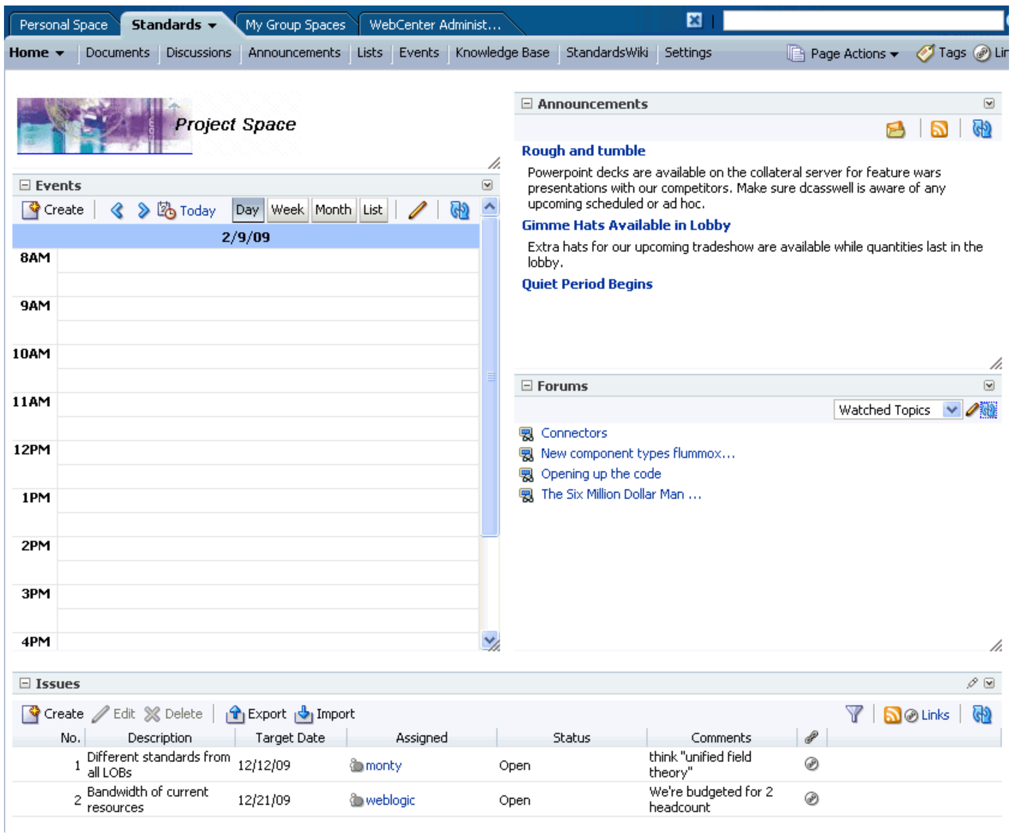
Page Creation Create personal and role-based pages for publishing, sharing, and reuse. Learn more

Out of the box, WebCenter Spaces offers a configurable work environment that enables individuals and groups to work and collaborate more effectively. The application is a self-service solution for managing individual and group interactions. It also provides intuitive tools that allow business users to come together and share information by adding pages and resources such as documents, charts, reports, portlets, business applications, Web 2.0 services, and other ADF resources or views.

WebCenter Spaces provides two work environments within a single application: *personal spaces* and *group spaces*. Personal spaces provide each user with a private work area for storing personal content, keeping notes, viewing and responding to business process assignments, maintaining a list of online buddies, and performing many other tasks relevant to his or her unique working day. Group spaces support discrete work areas organized around an area of interest or a common goal.

In WebCenter Spaces, group spaces support the formation and collaboration of project teams and communities of interest. Group spaces bring people together in a virtual environment for ongoing interaction and information sharing—in essence, forming a social network. Figure 10–2 shows an example of a group space.

Figure 10–2 An Oracle WebCenter Group Space



2/9/09

Rough and tumble

Powerpoint decks are available on the collateral server for feature wars presentations with our competitors. Make sure dcasswell is aware of any upcoming scheduled or ad hoc.

Gimme Hats Available in Lobby

Extra hats for our upcoming tradeshow are available while quantities last in the lobby.

Quiet Period Begins

- Connectors
- New component types flummox...
- Opening up the code
- The Six Million Dollar Man ...

No.	Description	Target Date	Assigned	Status	Comments
1	Different standards from all LOBs	12/12/09	monty	Open	think "unified field theory"
2	Bandwidth of current resources	12/21/09	weblogic	Open	We're budgeted for 2 headcount

Structurally, group spaces are comprised of pages, many of which are dedicated to a particular service. For example, a *Documents page* provides a central library for uploading, organizing, and managing group content. A *Lists page* provides the means to create and publish multicolumn lists. A *Search page* includes features for saving searches and managing search results. In addition to these and other default pages, a group space supports custom pages created by authorized users. Page creation is easy with a wide selection of predefined layouts. With little effort, users can provide pages neatly tailored to the unique needs of their team or community. For more information about group spaces, see the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

WebCenter provides various *services* that group spaces can use to offer a wide variety of functionality in support of group objectives. For example, a Documents service provides features for uploading and managing content; a Discussions service provides features for creating, managing, and participating in discussion forums. Some other examples of these services include Events, Lists, Announcements, and Search. These services are what helps to make group spaces a productive, collaborative working environment. Services expose subsets of their features and functionality through *task flows.* A task flow is a reusable view that may expose all or a subset of the features available from a particular service. For example, a Recent Documents task flow provides a subset of the functionality offered through the Documents service by listing documents that have recently been opened, added, or affected in some way.

To help users get group spaces up and running quickly, they can be based on out-of-the-box templates. These templates provide the optimal structure for supporting the two main types of group space. Use the Group Project template for group spaces where each member potentially comes from a different department but all members contribute toward reaching a common goal. Use the Community of Interest template to help create a group space for a community of people joining together to collaborate, create content, share ideas, and so on. Additionally, WebCenter Spaces allows users to turn any group space into a template, so they can design group spaces to suit specific audiences and offer them up as templates for others to use. For information about creating templates, see "Creating Your Own Group Space Templates" in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

WebCenter Spaces can consume portlets from any WSRP or Oracle PDK-Java portlet producer if the producer is registered first. Therefore, any portlets you create for custom WebCenter applications can be used in WebCenter Spaces too. For more information, see the "Managing Portlet Producers" chapter in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

10.2 Programmatically Exposing WebCenter Functionality in Custom WebCenter Applications

While using your custom WebCenter application, users may encounter situations where a group space would be useful to help them complete a particular task. In such cases, it would be much less disruptive to remain within the context of the current application, rather than having to switch to the WebCenter Spaces application. To this end, WebCenter Spaces provides access to a subset of its group space functionality through several APIs. Using these APIs, you can integrate powerful group space functionality into your custom WebCenter application.

You can use WebCenter Spaces APIs to:

- **Create and manage group spaces and templates.** You can create and delete group spaces, and add custom attributes. For more information, see Section 10.2.4.1, "Managing Group Spaces and Templates."
- **Manage group space membership.** You can add and remove group space members. For more information, see Section 10.2.4.2, "Managing Group Space Membership."
- **Retrieve group space and template information.** For example, you can retrieve the WebCenter Spaces URL or the URL of a specific group space. You can also retrieve group space and group space template metadata. For more information, see Section 10.2.4.3, "Retrieving Group Space and Template Information."

WebCenter Spaces APIs are contained within several classes. Table 10–1 lists the different classes and describes the purpose of the APIs within each class.

Table 10–1 WebCenter Spaces API Classes

Class	Contains APIs for	More Information
`GroupSpaceWSClient`	Creating and managing group spaces and templates Managing group space membership Retrieving group space information	Section 10.2.4, "How to Provide Group Space Functionality in Custom WebCenter Applications"
`GroupSpaceWSContext`	Establishing context before calling the APIs	Section 10.2.3.4, "Setting Up the Group Space Client Context"
`GroupSpaceWSException`	Managing exceptions raised by the APIs	Section 10.2.5, "How to Handle Exceptions Raised by WebCenter Spaces APIs"

Table 10–1 (Cont.) WebCenter Spaces API Classes

Class	Contains APIs for	More Information
`GroupSpaceWSMembers`	Retrieving information about group space members	Section 10.2.4.3, "Retrieving Group Space and Template Information"
`GroupSpaceWSMetadata`	Retrieving group space information	Section 10.2.4.3, "Retrieving Group Space and Template Information"

Group spaces have many different applications. From your perspective, as a custom WebCenter application developer, here are a couple of case studies describing scenarios where building and working with group spaces through the APIs might come in useful:

- In a purchasing application, a purchasing manager may create a group space to discuss the suitability of a new supplier. For more information, see Section 10.2.1, "Case Study 1: Purchasing Application Uses a Group Space to Evaluate Suppliers."
- In a Customer Support Center application, a support analyst may create a group space to collaborate with other users about a customer escalation. For more information, see Section 10.2.2, "Case Study 2: Customer Support Center Application Uses a Group Space to Discuss Customer Escalation."

10.2.1 Case Study 1: Purchasing Application Uses a Group Space to Evaluate Suppliers

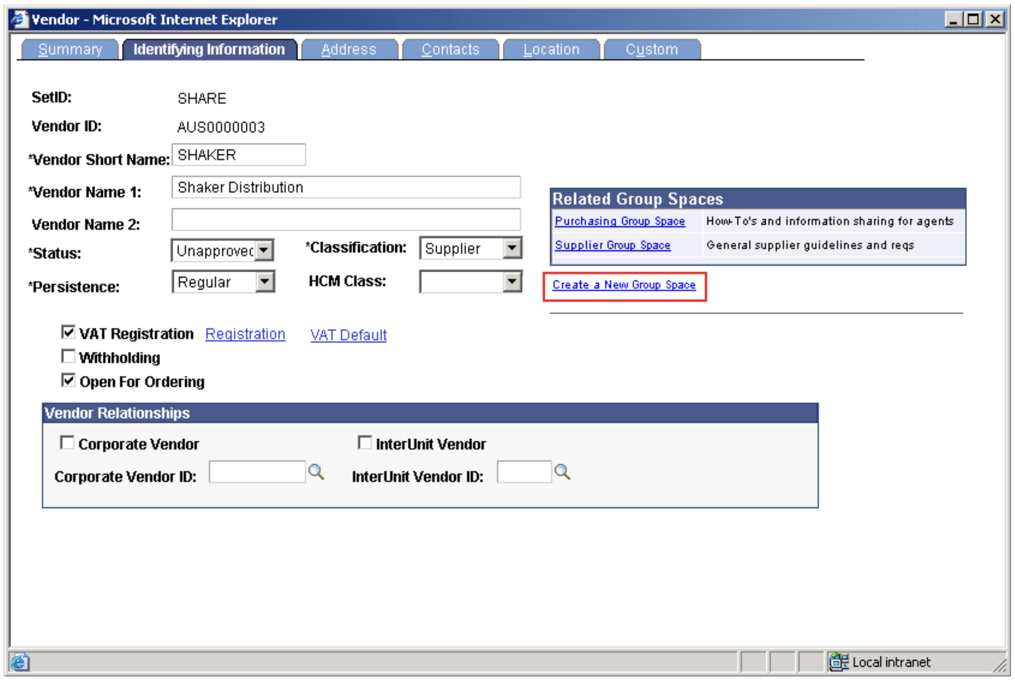
Consider a purchasing application built using the Oracle WebCenter Framework. This application tracks suppliers, pricing, lead-time requirements, delivery time estimates, and performance history.

Users of the purchasing application must also be able to select and evaluate supplier candidates. Supplier evaluation is a collaborative process; it requires people from various areas of a company. For example, a design engineer and manufacturing representative must verify that an item being purchased meets the required technical specifications; a purchasing agent can negotiate prices, logistics and contractual issues; and a manager or executive has to approve the deal. How could the purchasing application initiate supplier evaluations?

Typically, the purchasing manager receives a purchase requisition from the manufacturing department. Sometimes the purchase order cannot be completed because the requisition cannot be delivered by the usual supplier in the time frame required by manufacturing. Therefore, a new supplier that can meet delivery and pricing requirements must be determined. The purchasing manager can add a candidate supplier into the system but the purchasing manager needs a way to organize and share information, and collaborate with people in and outside his team so that he can assess the supplier.

A group space would provide this collaborative environment. So the purchasing application includes a call to the `createGroupSpace` API, and this enables the purchasing manager to click a link (**Create a New Group Space**) that displays a Create Group Space dialog, directly from the purchasing application, as shown in Figure 10–3.

Figure 10–3 Purchasing Application Includes a Group Space Creation Link



SetID:	SHARE
Vendor ID:	AUS0000003
*Vendor Short Name:	SHAKER
*Vendor Name 1:	Shaker Distribution
Vendor Name 2:	
*Status:	Unapproved
*Classification:	Supplier
*Persistence:	Regular
HCM Class:	

Related Group Spaces

Purchasing Group Space	How-To's and information sharing for agents
Supplier Group Space	General supplier guidelines and reqs

Create a New Group Space

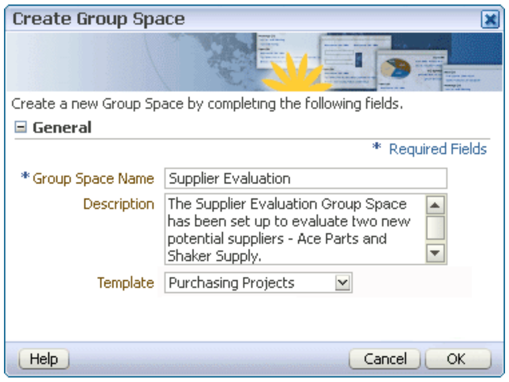
☑ VAT Registration Registration VAT Default
☐ Withholding
☑ Open For Ordering

Vendor Relationships

☐ Corporate Vendor ☐ InterUnit Vendor
Corporate Vendor ID: InterUnit Vendor ID:

When the purchasing manager clicks the link a custom dialog prompts him for some information. The purchasing manager enters a name and description for the group space, and also selects a template (Purchasing Projects) as shown in Figure 10–4. The Purchasing Projects template sets up the group space quickly so it is ready to use right away. For example, the template defines which services are required (events, a document library, and so on) and any custom pages that may be required. Because the APIs enable you to create your own Create Group Space dialog, you can apply your own look and feel and terminology.

Figure 10–4 Custom Create Group Space Dialog



Create a new Group Space by completing the following fields.

General

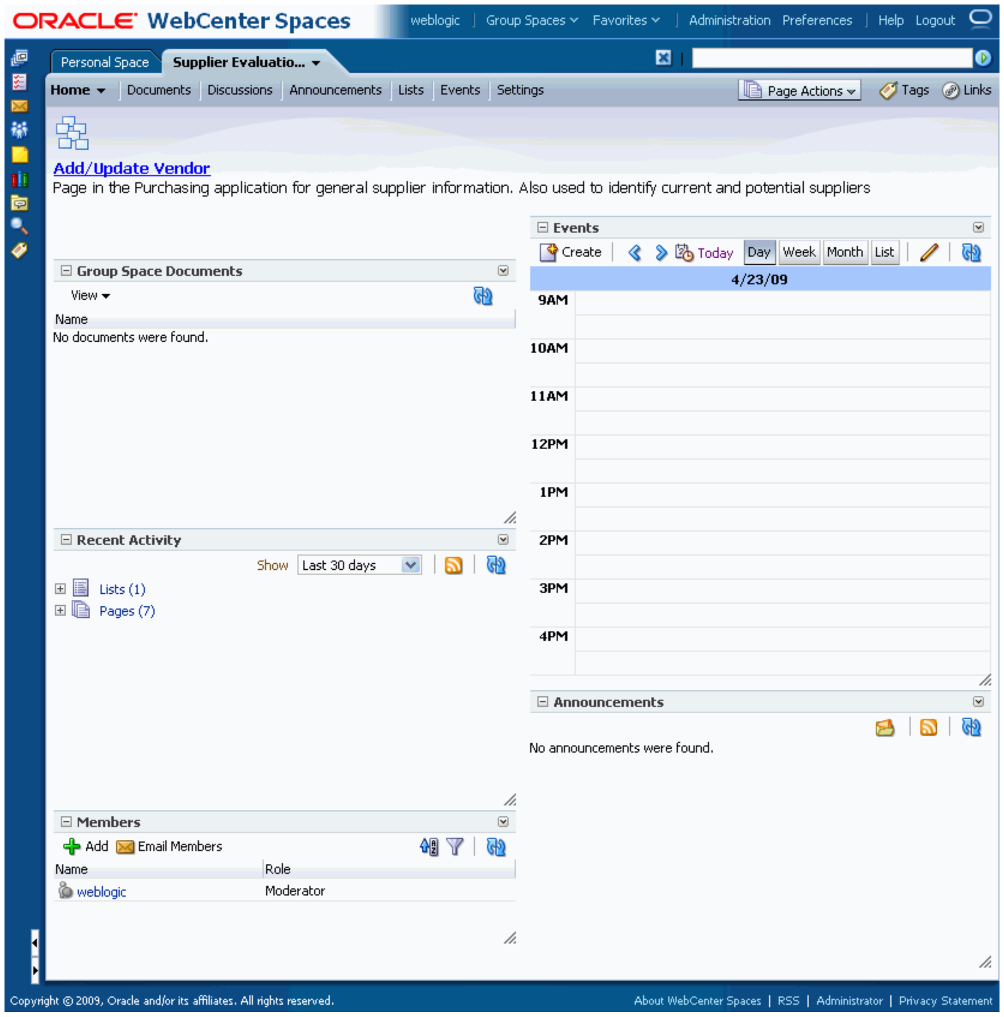
* Required Fields

* Group Space Name	Supplier Evaluation
Description	The Supplier Evaluation Group Space has been set up to evaluate two new potential suppliers - Ace Parts and Shaker Supply.
Template	Purchasing Projects

In this scenario, the purchasing manager chooses the Purchasing Projects template from the template list. Another approach would be to have the purchasing application pass in a default template value. With this additional default, there would be one less thing (determining which template to use) for the purchasing manager to think about. You could even generate the name of the group space from the Supplier ID so that the purchasing manager would not have to enter any details at all. The group space could then be created with just the click of a link.

When the purchasing manager clicks **OK**, WebCenter Spaces displays the new Supplier Evaluation group space in WebCenter Spaces, as shown in Figure 10–5.

Figure 10–5 Supplier Evaluation Group Space


ORACLE WebCenter Spaces
weblogic | Group Spaces | Favorites | Administration Preferences | Help Logout
Personal Space
Supplier Evaluatio...
Home | Documents | Discussions | Announcements | Lists | Events | Settings
Page Actions
Tags
Links
Events
Create
Today
Day
Week
Month
List
Group Space Documents
View
Name
Recent Activity
Show
Last 30 days
Lists (1)
Pages (7)
Announcements
Members
Add
Email Members
Name
Role
9AM
10AM
11AM
12PM
1PM
2PM
3PM
4PM
Copyright © 2009, Oracle and/or its affiliates. All rights reserved.
About WebCenter Spaces | RSS | Administrator | Privacy Statement

Add/Update Vendor

Page in the Purchasing application for general supplier information. Also used to identify current and potential suppliers

4/23/09

No documents were found.

No announcements were found.

Name	Role
weblogic	Moderator

Figure 10–5 shows the Home page for the Supplier Evaluation group space, which includes an events calendar, documents the group may find useful, an area for announcements, and so on. Each of these areas was determined by the Purchasing Projects template. In addition a link to the purchasing application transaction instance from which the group space was created is also provided. Clicking this link (**Add/Update Vendors**) displays the screen Add/Update Vendor transaction for the supplier Shaker Distribution.

In WebCenter Spaces, the purchasing manager becomes the moderator of the group space automatically. The moderator can add content to the group space, initiate discussions, invite members, and collaborate with interested parties. Once consensus is reached regarding the supplier, the purchasing manager is can approve or reject the supplier concerned.

10.2.2 Case Study 2: Customer Support Center Application Uses a Group Space to Discuss Customer Escalation

Consider a Customer Support Center application built using the Oracle WebCenter Framework that tracks customer calls and issues.

A support analyst is notified that a customer has escalated the service request that the analyst has been working on. The analyst knows that she can find a quicker resolution to the issue if she can involve other people from different areas of the company. For example:

- The project manager whose team implemented the project can provide more in depth knowledge of the project.
- An account manager who has been in constant touch with the customer can provide specific information about the implementation of the project at the customer site.
- Another support analyst, who has worked on a similar escalation before, can provide information about how that escalation was resolved.

This problem can be solved collaboratively using a group space. The support analyst creates a group space from within the Customer Support Center application and adds the required members. These members are then notified and use the group space to start discussing the customer situation. The support analyst views their updates to the group space inside the support application, navigating to the group space whenever necessary to obtain more specific details. Based on the information she gets from the other members of the group space, she can diagnose the problem and offer the customer a solution very quickly.

10.2.3 How to Set Up Your Custom WebCenter Application to Use the WebCenter Spaces APIs

Before you can use the WebCenter Spaces APIs you must ensure that the WebCenter Spaces application is up and running and that your project is set up correctly in JDeveloper.

This section includes the following subsections:

- Verifying That WebCenter Spaces Is Up and Running
- Setting Up the Custom WebCenter Application to Use WebCenter Spaces APIs
- Securing the Connection Between the Application and WebCenter Spaces
- Setting Up the Group Space Client Context

10.2.3.1 Verifying That WebCenter Spaces Is Up and Running

If you want to use the WebCenter SpacesAPIs, the WebCenter Spaces application must be up and running.

To verify that WebCenter Spaces is up and running:

1. Start WebCenter Spaces. The URL is:

   ```
   http://host:port/webcenter
   ```

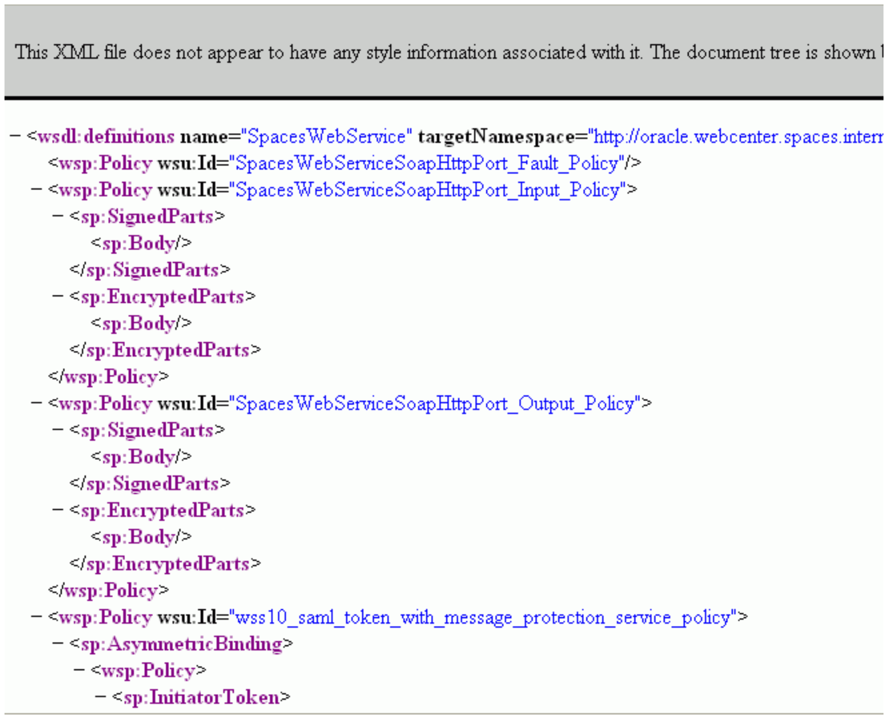
   You do not need to log in.

2. Verify that the WebCenter Spaces Web service is up and running. The URL is:

   ```
   http://host:port/webcenter/SpacesWebService?WSDL
   ```

You should see a page similar to the one shown in Figure 10–6. If you do not see this page, contact your Fusion Middleware administrator.

Figure 10–6 SpacesWebService WSDL



This XML file does not appear to have any style information associated with it. The document tree is shown

```
- <wsdl:definitions name="SpacesWebService" targetNamespace="http://oracle.webcenter.spaces.interr
    <wsp:Policy wsu:Id="SpacesWebServiceSoapHttpPort_Fault_Policy"/>
  - <wsp:Policy wsu:Id="SpacesWebServiceSoapHttpPort_Input_Policy">
    - <sp:SignedParts>
        <sp:Body/>
      </sp:SignedParts>
    - <sp:EncryptedParts>
        <sp:Body/>
      </sp:EncryptedParts>
    </wsp:Policy>
  - <wsp:Policy wsu:Id="SpacesWebServiceSoapHttpPort_Output_Policy">
    - <sp:SignedParts>
        <sp:Body/>
      </sp:SignedParts>
    - <sp:EncryptedParts>
        <sp:Body/>
      </sp:EncryptedParts>
    </wsp:Policy>
  - <wsp:Policy wsu:Id="wss10_saml_token_with_message_protection_service_policy">
    - <sp:AsymmetricBinding>
      - <wsp:Policy>
        - <sp:InitiatorToken>
```

For more information about setting up WebCenter Spaces, see the "Getting WebCenter Spaces Up and Running" chapter in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

10.2.3.2 Setting Up the Custom WebCenter Application to Use WebCenter Spaces APIs

Before you can call the WebCenter Spaces APIs in your custom WebCenter application, you must ensure that your application contains the correct libraries and that there is a connection to the WebCenter Spaces Web service.

To set up your application to use the WebCenter Spaces APIs:

1. Start JDeveloper.
2. In the Application Navigator, expand the application for which you want to provide group space functionality.

   The application should be based on the WebCenter Application template.
3. Right-click the view-controller project and choose **Project Properties.**
4. Select **Libraries and Classpath.**
5. Check that the following libraries are available in your project, adding any as necessary:

- ADF DVT Faces Runtime
- ADF Model Generic Runtime
- ADF Model Runtime
- JAX-RPC Client
- JAX-WS Client
- JAXB
- Oracle JEWT
- WebCenter Spaces Client
- `JDEV_HOME/jdeveloper/modules/oracle.adf.model_11.1.1/adfm.jar`
- `JDEV_HOME/jdeveloper/modules/oracle.xdk_11.1.1/xml.jar`

6. Click **OK.**
7. In the Application Resources panel of the Application Navigator, right-click **Connections** and choose **New Connection > URL.**
8. In the **Name** field, enter `SpacesWebServiceEndpoint`.
9. In the **URL Endpoint** field, enter the WebCenter Spaces Web service URL:

   ```
   http://host:port/webcenter/SpacesWebService
   ```

10. Click **OK.**

    The connection information is added to the `connections.xml` file in the application's `META-INF` directory, for example:

    ```
    C:\JDeveloper\mywork\mySpacesApplication\.adf\META-INF
    ```

    If the `connections.xml` file does not exist, it is created.

11. Open the `connections.xml` file and verify that the code shown in Example 10–1 appears:

Example 10–1 WebCenter Spaces Web Service Endpoint URL Connection

```
<Reference name="SpacesWebServiceEndpoint"
className="oracle.adf.model.connection.url.HttpURLConnection">
<Factory className="oracle.adf.model.connection.url.URLConnectionFactory"/>
<RefAddresses>
  <XmlRefAddr addrType="SpacesWebServiceEndpoint">
    <Contents>
      <urlconnection name="SpacesWebServiceEndpoint"
      url="http://host:port/webcenter/SpacesWebService"/>
    </Contents>
  </XmlRefAddr>
</RefAddresses>
</Reference>
```

If you need to set the WebCenter Spaces Web service endpoint at runtime, you can use the `setGroupSpaceWebServiceEndpoint` API (Example 10–2). You can write a wrapper API that takes the endpoint as a parameter and then calls `setGroupSpaceWebServiceEndpoint`, passing the parameter. An exception is raised if the endpoint is already set in `connections.xml`.

Example 10–2 Setting the WebCenter Spaces Web Service Endpoint at Runtime

```
client.setGroupSpacesWebServiceEndpoint("http://myserver.example.com/webcenter/Spa
cesWebService");
```

10.2.3.3 Securing the Connection Between the Application and WebCenter Spaces

Before using the WebCenter Spaces APIs in your custom WebCenter application, you must ensure that the communication between the application (the consumer) and WebCenter Spaces (the producer) is secure. This is so that the identity of the user invoking the APIs is propagated to WebCenter Spaces in a secure manner where the integrity and confidentiality of the communication is maintained.

To do this, the WebCenter Spaces APIs use a policy of SAML based token passing with message protection. Your administrator must create a Java keystore and update the credential store so that WebCenter Spaces can verify the authenticity of the security tokens received from your application. You must then register this keystore and update the credential store using JDeveloper.

For information about the steps that the administrator must perform, see "Securing WebCenter Spaces for Applications Consuming Spaces Client APIs with WS-Security" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

Before You Begin

Obtain the following from your administrator:

- Consumer keystore to use to secure the connection. This is a `.jks` file (for example, `consumer.jks`).
- Consumer public alias key stored in the keystore (for example, `consumer`).
- Password of the consumer public alias key (for example, `mypassword1`).
- Producer public alias key stored in the consumer keystore (for example, `producer`). This is the alias specified by the administrator when importing the trusted certificate of the producer.
- Consumer keystore password (for example, `mypassword2`).

To secure the connection:

1. Copy the keystore to the file system where your custom WebCenter application is running, for example:

   ```
   domain_home/config/fmwconfig
   ```

2. Open the `jps-config.xml` file and configure the keystore created by your administrator (Example 10–3).

Example 10–3 Keystore Instance in jps.config.xml

```
<!-- KeyStore Service Instance -->
    <serviceInstance name="keystore" provider="keystore.provider"
    location="./consumer.jks">
        <description>Default JPS Keystore Service</description>
        <property name="keystore.type" value="JKS"/>
            <property name="keystore.csf.map" value="oracle.wsm.security"/>
        <property name="keystore.pass.csf.key" value="keystore-csf-key"/>
        <property name="keystore.sig.csf.key" value="enc-csf-key"/>
        <property name="keystore.enc.csf.key" value="enc-csf-key"/>
    </serviceInstance>
```

You can use Fusion Middleware Control to specify the keystore location if you prefer. For more information see "Registering the Keystores" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

3. Take a backup of `cwallet.sso`.
4. Open a command line prompt.
5. Navigate to the directory where JDeveloper is installed, for example:

   ```
   C:\Jdev\jdeveloper\common\bin
   ```

6. Run the following command:
   - In Windows: `wlst.cmd`
   - In Linux: `./wlst.sh`
7. Run:

   ```
   connect('admin_user', 'admin_password', 'host:port')
   ```

   Where:
   - *`admin_user`* is the user name for the administrator of the server on which the application is running (for example, `weblogic`)
   - *`admin_password`* is the password for the administrator (for example, `weblogicpwd1`)
   - *`host`*`:`*`port`* is the server on which the application is running
8. Run the following commands:

   ```
   createCred(map="oracle.wsm.security", key="enc-csf-key", user="alias",
   password="key_password", desc="Enc Password")

   createCred(map="oracle.wsm.security", key="sign-csf-key", user="alias",
   password="key_password", desc="Enc Password")

   createCred(map="oracle.wsm.security", key="keystore-csf-key",
   user="keystore-csf-key", password="keystore_password", desc="Keystore
   password")
   ```

   Where:
   - *`alias`* is the consumer public alias key in the keystore
   - *`key_password`* is the password for the public key
   - *`keystore_password`* is the keystore password

   You can obtain this information from your administrator, as detailed in the "Before You Begin" section.

   If you need any additional help to run these commands, see the "createCred" section in the *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*.

   You can use Fusion Middleware Control to update the credential store if you prefer. For more information see the "Securing WebCenter Spaces for Applications Consuming Spaces Client APIs with WS-Security" section in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.
9. Restart the server on which your application is deployed, and then your default server.

10.2.3.4 Setting Up the Group Space Client Context

Before calling any of the WebCenter Spaces APIs in your application code, a few setup steps are required.

To set up the group space client context

1. Domain-wide configuration settings are used when you call WebCenter Spaces APIs from an application deployed on WebLogic Server (WLS), unless you specifically choose to override them. To override or set security configuration parameters, use the APIs provided by the `GroupSpaceWSContext` class.

   In particular, you must set the SAML issuer name and the public key alias for WebCenter Spaces, which are required for data encryption. If the WebCenter Spaces Web service endpoint is not set in the `connections.xml` file, you can set this too.

   For example:

   ```
   GroupSpaceWSContext context = new GroupSpaceWSContext();

   context.setEndPoint("endPointUrl");
   context.setSamlIssuerName("samlIssuer");
   context.setRecipientKeyAlias("producer");

   groupSpaceWSClient = new GroupSpaceWSClient(context);
   ```

   Where:

   - *`endPointUrl`* is the WebCenter Spaces Web service endpoint, for example, http://server.example.com:8912/webcenter/SpacesWebService).
   - *`samlIssuer`* is the issuer URI of the SAML Authority issuing assertions for this SAML Asserting Party (for example, `http://www.example.com/webcenter`)
   - *`producer`* is the public key alias for WebCenter Spaces, for example, `wcspaces`.

     You can obtain this information from your administrator, as detailed in the "Before You Begin" section.

   For a full list of the APIs provided by the `GroupSpaceWSContext` class, see the *Oracle Fusion Middleware WebCenter Group Spaces Java API Reference*.

2. Initialize the group space client by passing the context. For example:

   ```
   GroupSpaceWSClient client = new GroupSpaceWSClient(context);
   ```

3. Once you have initialized the client, check everything is set up correctly by calling:

   ```
   getWebCenterSpacesURL();
   ```

   If the correct URL is returned, everything is set up correctly and you can start using the group space APIs.

10.2.4 How to Provide Group Space Functionality in Custom WebCenter Applications

The WebCenter Spaces APIs enable you to perform commonly used group space operations within a custom WebCenter application. Most of these APIs are provided by the `GroupSpaceWSClient` class. The APIs can be grouped into three main categories:

- Managing Group Spaces and Templates

- Managing Group Space Membership
- Retrieving Group Space and Template Information

Table 10–2 lists the APIs in the `GroupSpaceWSClient` class.

Table 10–2 APIs for Performing Common Group Space Operations

Group Space API	Category	Description
`createGroupSpace`	Managing Group Spaces and Templates	Creates a new group space based on a template. See Section 10.2.4.1.1, "Creating a Group Space."
`setCustomAttribute`	Managing Group Spaces and Templates	Creates one or more custom attributes for a group space. See Section 10.2.4.1.2, "Creating a Custom Attribute."
`deleteGroupSpace`	Managing Group Spaces and Templates	Permanently removes a group space from WebCenter Spaces. See Section 10.2.4.1.4, "Deleting a Group Space."
`createGroupSpaceTemplate`	Managing Group Spaces and Templates	Creates a new group space template based on an existing group space. See Section 10.2.4.1.3, "Creating a Group Space Template."
`addMember`	Managing Group Space Membership	Adds a group space member with a specific role. See Section 10.2.4.2.1, "Adding Members to a Group Space."
`inviteMember`	Managing Group Space Membership	Invites a list of users to become members of a group space. See Section 10.2.4.2.2, "Inviting Users to Join a Group Space."
`removeMember`	Managing Group Space Membership	Revokes group space membership. See Section 10.2.4.2.3, "Removing Members from a Group Space."
`getRoles`	Managing Group Space Membership	Retrieves all the roles for a given group space. See Section 10.2.4.2.4, "Retrieving Role Information."
`getGroupSpaces`	Retrieving Group Space and Template Information	Retrieves a list of group spaces for a given query string. See Section 10.2.4.3.1, "Retrieving a List of Group Spaces."
`getPublicGroupSpaces`	Retrieving Group Space and Template Information	Retrieves a list of public group spaces for a given query string. See Section 10.2.4.3.2, "Retrieving a List of Public Group Spaces."
`getGroupSpaceMetadata`	Retrieving Group Space and Template Information	Retrieves information (metadata) about the group space. See Section 10.2.4.3.3, "Retrieving Group Space Metadata."
`getGroupSpaceMetadataByGuid`	Retrieving Group Space and Template Information	Retrieves information (metadata) about a group space given the group space's GUID. See Section 10.2.4.3.3, "Retrieving Group Space Metadata."

Table 10–2 (Cont.) APIs for Performing Common Group Space Operations

Group Space API	Category	Description
`getGroupSpaceTemplates`	Retrieving Group Space and Template Information	Retrieves a list of group space templates for a given query string. See Section 10.2.4.3.4, "Retrieving a List of Group Space Templates."
`getGroupSpaceTemplateMetadata`	Retrieving Group Space and Template Information	Retrieves information (metadata) about the group space template. See Section 10.2.4.3.5, "Retrieving Group Space Template Metadata."
`getGroupSpaceTemplateMetadataByGuid`	Retrieving Group Space and Template Information	Retrieves information (metadata) about the group space template given the template's GUID. See Section 10.2.4.3.5, "Retrieving Group Space Template Metadata."
`getWebCenterSpacesURL`	Retrieving Group Space and Template Information	Retrieves the WebCenter Spaces URL. See Section 10.2.4.3.6, "Retrieving the WebCenter Spaces URL."
`getGroupSpaceURL`	Retrieving Group Space and Template Information	Retrieves a group space URL. See Section 10.2.4.3.7, "Retrieving a Group Space URL."

10.2.4.1 Managing Group Spaces and Templates

Use the following WebCenter Spaces APIs to manage your group spaces:

- `createGroupSpace`
- `setCustomAttribute`
- `createGroupSpaceTemplate`
- `deleteGroupSpace`

Before you begin

Before using any of these APIs, you must complete all the steps listed in Section 10.2.3, "How to Set Up Your Custom WebCenter Application to Use the WebCenter Spaces APIs."

This section includes the following subsections

- Creating a Group Space
- Creating a Custom Attribute
- Creating a Group Space Template
- Deleting a Group Space

10.2.4.1.1 Creating a Group Space Use the `createGroupSpace` API to create a group space that is based on an existing group space template.

To use this API, specify:

- The internal name of the new group space. This name can contain spaces.
- A display name for the new group space.
- A description of the group space. Although this is not mandatory, we recommend that you provide a description to help users identify the group space's purpose.

- The name of the group space template you want to use. This should be the internal name of the template, not the display name. For example, the internal names for the built-in templates are Basic, CommunityOfInterest, and ProjectSpace.

Optionally, you can provide a comma separated list of keywords to help users locate the group space in searches.

Example 10–4 creates a group space with the name *Databases* that is based on the *CommunityofInterest* template. The example also specifies two search keywords (*databases* and *Oracle*).

Example 10–4 Creating a Group Space

```
//create the group space
GroupSpaceWSMetadata gsMetadata =
client.createGroupSpace("Databases", "Databases" "A community for people
interested in databases", "databases, oracle", "CommunityofInterest");
//print the group space GUID to provide confirmation of creation
System.out.println("GUID: " + gsMetadata.getGuid());
```

For an example of how to provide custom attributes for a new group space, see Section 10.2.4.1.2, "Creating a Custom Attribute."

10.2.4.1.2 Creating a Custom Attribute Every group space comes with built-in attributes such as name, description, date created, icon, and so on. In addition, you can define custom attributes that store additional information (metadata) that is unique to the group space and its characteristics.

Use the `setCustomAttribute` API to specify a custom attribute for a group space. To use this API, specify the name of the group space and a name, description, type, and value for the custom attribute.

Example 10–5 creates the Databases group space and then creates a custom attribute for that group space. The attribute is named *Vendors,* with the description *List of vendors.* It takes string values of *Oracle* and *IBM.*

Example 10–5 Creating a Custom Attribute

```
//create the group space
client.createGroupSpace("Databases", "Databases", "A community for people
interested in databases", null, "CommunityofInterest");
//create the custom attribute
client.setCustomAttribute("Databases", "Vendors", "List of vendors",
"java.lang.String", "Oracle, IBM");
```

10.2.4.1.3 Creating a Group Space Template Use the `createGroupSpaceTemplate` API to create a group space template based on an existing group space. To use this API specify a name, display name, and description for the template, and the name of the group space to use to create the template.

Example 10–6 creates a group space template based on the *Databases* group space.

Example 10–6 Creating a Group Space Template

```
GroupSpaceWSMetadata templMetadata =
client.createGroupSpaceTemplate("DatabasesTemplate", "Databases Template",
"A template based on the Databases group space", "Databases");
//print the template GUID to provide confirmation of creation
System.out.println("GUID: " + templMetadata.getGuid());
```

10.2.4.1.4 Deleting a Group Space Use the `deleteGroupSpace` API to permanently delete a group space from WebCenter Spaces. To use this API, specify the name of the group space to delete. The API returns a boolean value to indicate whether the deletion was successful.

Example 10–7 deletes the group space with the name *Databases.* The example uses the boolean value returned by the API to print a message about the success or failure of the operation.

Example 10–7 Deleting a Group Space

```
//delete the group space
Boolean success = client.deleteGroupSpace("Databases");
//print a message depending on the result of the deletion
if(success)
{
  System.out.println("Operation Succeeded");
}
else
{
  System.out.println(Operation Failed");
}
```

10.2.4.2 Managing Group Space Membership

Use the following group space APIs to manage group space membership:

- `addMember`
- `inviteMember`
- `removeMember`
- `getRoles`

Before you begin

Before using any of these APIs, you must complete all the steps listed in Section 10.2.3, "How to Set Up Your Custom WebCenter Application to Use the WebCenter Spaces APIs."

This section includes the following subsections:

- Adding Members to a Group Space
- Inviting Users to Join a Group Space
- Removing Members from a Group Space
- Retrieving Role Information

10.2.4.2.1 Adding Members to a Group Space Use the `addMember` API to give users group space membership and assign the new members to a group space role. To use this API, specify the name of the group space, and a list of users. The list must contain objects of type `GroupSpaceWSMembers` that specify the user name and the role to give that user in the group space.

You must specify a valid user name that exists in the WebCenter Spaces identity store. For the role, choose one of the default roles (moderator, participant, viewer) or one of the custom group space roles (if any). To retrieve a list of the available roles for a group space, use the `getRoles` API. For more information, see Section 10.2.4.2.4, "Retrieving Role Information." For more information about roles, see the "What You

Should Know About Group Space Roles and Permissions" section in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

Example 10–8 adds users (of type `GroupSpaceWSMembers`) named *Pat* and *Vicki* as members of the *Databases* group space with *viewer* and *participant* roles respectively.

Example 10–8 Adding Members to a Group Space

```
//create the list of users
List addMem = new ArrayList();
//create the GroupSpaceWSMembers objects
GroupSpaceWSMembers mem1 = new GroupSpaceWSMembers("pat", "viewer");
GroupSpaceWSMembers mem2 = new GroupSpaceWSMembers("vicki", "participant");
//add the GroupSpaceWSMembers objects to the list of users
addMem.add(mem1);
addMem.add(mem2);
//add the users to the group space
client.addMember("Databases", addMem);
//print a list of members to confirm the new members were added
GroupSpaceWSMetadata memMetData = client.getGroupSpaceMetadata("Databases");
for (GroupSpaceWSMembers mems: memMetData.getMembers())
{
  System.out.println(mems.getMember());
  System.out.println(mems.getRole());
}
```

10.2.4.2.2 Inviting Users to Join a Group Space Use the `inviteMember` API to invite users to become members of a group space. To use this API, specify the name of the group space and a list of users to invite. The list must contain objects of type `GroupSpaceWSMembers` that specify the user name and the role to give that user in the group space. An invitation to join the group space is sent to each user, and each user can then accept or reject that invitation.

You must specify a valid user name that exists in the WebCenter Spaces identity store. For the role, choose one of the default roles (moderator, participant, viewer) or one of the custom group space roles (if any). To retrieve a list of the available roles for a group space, use the `getRoles` API. For more information, see Section 10.2.4.2.4, "Retrieving Role Information." For more information about roles, see the "What You Should Know About Group Space Roles and Permissions" section in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

Example 10–9 invites users (of type `GroupSpaceWSMembers`) named *Pat* and *Vicki* to become members of the *Databases* group space with *viewer* and *participant* roles respectively.

Example 10–9 Inviting Users to Join a Group Space

```
//create the list of users
List inviteMem = new ArrayList();
//create the GroupSpaceWSMembers objects
GroupSpaceWSMembers mem1 = new GroupSpaceWSMembers("pat", "viewer");
GroupSpaceWSMembers mem2 = new GroupSpaceWSMembers("vicki", "participant");
//add the GroupSpaceWSMembers objects to the list of users
inviteMem.add(mem1);
inviteMem.add(mem2);
//invite the list of users to join the group space
client.inviteMember("Databases", inviteMem);
```

10.2.4.2.3 Removing Members from a Group Space Use the `removeMember` API to revoke group space membership. To use this API, specify the name of the group space, and a list of users. The list must contain objects of type `GroupSpaceWSMembers` that specify the user name. To obtain a list of current group space members, use the `getGroupSpaceMetadata` API. For more information, see Section 10.2.4.3.3, "Retrieving Group Space Metadata."

Example 10–10 removes users (of type `GroupSpaceWSMembers`) *Pat* and *Vicki* from the membership of the *Databases* group space.

Example 10–10 Removing Members from a Group Space

```
//create the list of users
List remMem = new ArrayList();
//create the GroupSpaceWSMembers objects
GroupSpaceWSMembers mem1 = new GroupSpaceWSMembers("pat");
GroupSpaceWSMembers mem2 = new GroupSpaceWSMembers("vicki");
//add the GroupSpaceWSMembers objects to the list of users
remMem.add(mem1);
remMem.add(mem2);
//remove the users from the group space
client.removeMember("Databases", remMem);
//print a list of members to confirm the members were removed
GroupSpaceWSMetadata memMetData = client.getGroupSpaceMetadata("Databases");
for (GroupSpaceWSMembers mems: memMetData.getMembers())
{
  System.out.println(mems.getMember());
  System.out.println(mems.getRole());
}
```

10.2.4.2.4 Retrieving Role Information Use the `getRoles` API to retrieve all the roles for a given group space. This is useful for determining which roles are available when adding or inviting members to a group space. Roles include the default roles (moderator, participant, viewer) as well as custom group space roles (if any). To use this API, specify the name of the group space.

Example 10–11 retrieves and displays role information for the *Databases* group space.

Example 10–11 Retrieving Role Information

```
//retrieve the list of roles
List<String> roles = client.getRoles("Databases");
//print the list of roles
for (String role: roles)
{
  System.out.println(role);
}
```

10.2.4.3 Retrieving Group Space and Template Information

Use the following group space APIs to retrieve group space information:

- `getGroupSpaces`
- `getPublicGroupSpaces`
- `getGroupSpaceMetadata`
- `getGroupSpaceMetadataByGuid`

- `getGroupSpaceTemplates`
- `getGroupSpaceTemplateMetadata`
- `getGroupSpaceTemplateMetadataByGuid`
- `getWebCenterSpacesURL`
- `getGroupSpaceURL`

Before you begin

Before using any of these APIs, you must complete all the steps listed in Section 10.2.3, "How to Set Up Your Custom WebCenter Application to Use the WebCenter Spaces APIs."

This section includes the following subsections

- Retrieving a List of Group Spaces
- Retrieving a List of Public Group Spaces
- Retrieving Group Space Metadata
- Retrieving a List of Group Space Templates
- Retrieving Group Space Template Metadata
- Retrieving the WebCenter Spaces URL
- Retrieving a Group Space URL

10.2.4.3.1 Retrieving a List of Group Spaces Use the `getGroupSpaces` API to obtain a list of group spaces that match a given query string. To use this API, specify a query string. A null value query string returns a list of *all* group spaces that are accessible to the current user.

The API returns group spaces that are accessible to the current user, up to a maximum of 500.

Example 10–12 returns a list of group spaces containing the string *Database.*

Example 10–12 Retrieving a List of Specific Group Spaces

```
List<String> allGroupSpaces = client.getGroupSpaces("Database");
```

Example 10–13 returns a list of *all* group spaces to which the current user has access. This is achieved by specifying a *null* query string.

Example 10–13 Retrieving a List of All Group Spaces

```
List<String> allGroupSpaces = client.getGroupSpaces(null)
```

10.2.4.3.2 Retrieving a List of Public Group Spaces Use the `getPublicGroupSpaces` API to obtain a list of public group spaces that match a given query string. To use this API, specify a query string. A null value query string returns a list of *all* public group spaces.

The API returns group spaces that are accessible to all users, even those who are not logged in to WebCenter Spaces, up to a maximum of 500.

Example 10–14 returns a list of public group spaces containing the string *Database*.

Example 10–14 Retrieving a List of Specific Public Group Spaces

```
List<String> allPublicGroupSpaces = client.getPublicGroupSpaces("Database");
```

Example 10–15 returns a list of *all* public group spaces. This is achieved by specifying a *null* query string.

Example 10–15 Retrieving a List of All Public Group Spaces

```
List<String> allPublicGroupSpaces = client.getPublicGroupSpaces(null)
```

10.2.4.3.3 Retrieving Group Space Metadata

Use the `getGroupSpaceMetadata` or `getGroupSpaceMetadataByGuid` APIs to obtain information (metadata) about a particular group space. This includes information such as the description of the group space, the name of the user who created it, the date on which it was last updated, and so on.

To use the `getGroupSpaceMetadata` API, specify the name of the group space. To use the `getGroupSpaceMetadataByGuid` API, specify the GUID of the group space. Note that while the group space name may be changed during the existence of the group space, the GUID always remains the same. You can obtain the GUID of a group space as follows:

```
getGroupSpaceMetadata("spaceName").getGuid();
```

Both APIs return a bean object that contains more APIs that you can then use for retrieving the group space metadata. These APIs are provided by the `GroupSpaceWSMetadata` class. Table 10–3 lists the APIs returned by the bean object:

Table 10–3 APIs for Retrieving Group Space Metadata

Group Space API	Description
`getCreatedBy`	Returns the name of the person who created the group space.
`getCustomAttributes`	Returns the custom attributes for a group space.
`getDescription`	Returns the description of the group space.
`getDisplayName`	Returns the display name of the group space.
`getGroupSpaceState`	Returns the state of the group space: active, offline, deleted.
`getGuid`	Returns the GUID of the group space.
`getIconURL`	Returns the location of the group space icon.
`getKeywords`	Returns a comma separated list of keywords used to describe the group space.
`getLastUpdated`	Returns the date the group space was last updated.
`getLogoURL`	Returns the location of the group space logo.
`getMailingList`	Returns the mailing list for the group space.
`getMembers`	Returns the current list of group space members.
`getName`	Returns the internal name of the group space.
`isDiscoverable`	Returns if the group space is returned in searches made by users who are not members of the group space.
`isPublic`	Returns if the group space is available to users who are not logged in.

Example 10–16 retrieves the *Description, Keywords,* and *Last Updated Date* information for the *Databases* group space given the group space name.

Example 10–16 Retrieving Group Space Metadata Using the Group Space Name

```
//get the exact name of the group space
List<String> myGroupSpace = client.getGroupSpaces("Databases");
//check that the API returns a single result
if(myGroupSpace.size() == 1)
{
  //retrieve the metadata
  GroupSpaceWSMetadata metadata = client.getGroupSpaceMetadata(myGroupSpace);
  //get group space description
  System.out.println("Description: " + metadata.getDescription());
  //get group space keywords
  System.out.println("Keywords: " + metadata.getKeywords());
  //get the date the group space was last updated
  System.out.println("Last Updated Date: "+ metadata.getLastUpdated().toString());
}
```

Example 10–17 retrieves the *Display Name, Creator,* and *Last Updated Date* information for the Databases group space given the group space GUID.

Example 10–17 Retrieving Group Space Metadata Using Group Space GUID

```
GroupSpaceWSMetadata metadata = client.getGroupSpaceMetadataByGuid("Guid");
//get group space display name
System.out.println("Display Name: " + metadata.getDisplayName());
//get the name of the user who created the group space
System.out.println("Created By: " + metadata.getCreatedBy());
//get the date the group space was last updated
System.out.println("Last Updated Date: " + metadata.getLastUpdated().toString());
```

Example 10–18 retrieves the *Name, Description, Type,* and *Value* for every custom attribute associated with the *Databases* group space.

Example 10–18 Retrieving Custom Attribute Metadata

```
//get the exact name of the group space
List<String> myGroupSpace = client.getGroupSpaces("Databases");
//check that the API returns a single result
if(myGroupSpace.size() == 1)
{
  //retrieve the metadata
  GroupSpaceWSMetadata metadata = client.getGroupSpaceMetadata(myGroupSpace);
  //get list of custom attributes
  List<CustomMetadata.Attribute> attributes = metadata.getCustomAttributes();
  //get name, description, type, and value for each custom attribute
  for(CustomMetadata.Attribute attribute: attributes)
  {
    System.out.println("Name :" + attribute.getName());
    System.out.println("Description :" + attribute.getDescription());
    System.out.println("Type :" + attribute.getType());
    System.out.println("value.toString() :" + attribute.getValue().toString());
  }
}
```

Example 10–19 retrieves a list of members of the *Databases* group space.

Example 10–19 Retrieving Membership Information

```
//get the exact name of the group space
List<String> myGroupSpace = client.getGroupSpaces("Databases");
//check that the API returns a single result
if(myGroupSpace.size() == 1)
{
  //retrieve the metadata
  GroupSpaceWSMetadata metadata = client.getGroupSpaceMetadata(myGroupSpace);
  //get the list of members
  for(String member: metadata.getMembers())
  {
    System.out.println("Member UID: " + member);
  }
}
```

10.2.4.3.4 Retrieving a List of Group Space Templates

Use the `getGroupSpaceTemplates` API to obtain a list of group space templates that match a given query string. To use this API, specify a query string. A null value query string returns a list of *all* templates that are accessible to the current user.

The API returns templates that are accessible to the current user, up to a maximum of 500.

Example 10–20 returns a list of group space templates containing the string *Interest.*

Example 10–20 Retrieving a List of Specific Group Space Templates

```
List<String> allGroupSpaceTemplates = client.getGroupSpaceTemplates("Interest");
```

Example 10–21 returns a list of *all* group space templates to which the current user has access. This is achieved by specifying a *null* query string.

Example 10–21 Retrieving a List of All Group Space Templates

```
List<String> allGroupSpaceTemplates = client.getGroupSpaceTemplates(null);
```

10.2.4.3.5 Retrieving Group Space Template Metadata

Use the `getGroupSpaceTemplateMetadata` and `getGroupSpaceTemplateMetadataByGuid` APIs to obtain information (metadata) about a particular group space template. This includes information such as the description of the template, the name of the user who created it, and so on.

To use the `getGroupSpaceTemplateMetadata` API, specify the name of the template. To use the `getGroupSpaceTemplateMetadataByGuid` API, specify the GUID of the template. Note that while the group space template name may be changed during the existence of the template, the GUID always remains the same.

Both APIs return a bean object that contains more APIs that you can then use for retrieving the group space template metadata. These APIs are provided by the `GroupSpaceWSMetadata` class. Table 10–4 lists the APIs returned by the bean object:

Table 10–4 APIs for Retrieving Group Space Template Metadata

Group Space API	Description
`getCreatedBy`	Returns the name of the person who created the template.
`getDescription`	Returns the description of the template.

Table 10–4 (Cont.) APIs for Retrieving Group Space Template Metadata

Group Space API	Description
`getDisplayName`	Returns the display name of the template.
`getGuid`	Returns the GUID of the template.
`getIconURL`	Returns the location of the template icon.
`getKeywords`	Returns a comma separated list of keywords used to describe the template.
`getLogoURL`	Returns the location of the template logo.
`getName`	Returns the internal name of the template.

Example 10–22 retrieves the *GUID, Description,* and *Created By* information for the *CommunityofInterest* group space template given the template name.

Example 10–22 Retrieving Template Metadata Using the Template Name

```
GroupSpaceWSMetadata metadata =
client.getGroupSpaceTemplateMetadata(myGroupSpaceTemplate);
//get the exact name of the group space template
List<String> myGroupSpaceTemplate =
client.getGroupSpaceTemplates("CommunityofInterest");
//check that the API returns a single result
if(myGroupSpaceTemplate.size() == 1)
{
  //retrieve the metadata -- get GUID
  System.out.println("GUID: " + metadata.getGuid());
  //get template description
  System.out.println("Description: " + metadata.getDescription());
  //get name of user who created the template
  System.out.println("Keywords: " + metadata.getCreatedBy());
}
```

Example 10–23 retrieves the *name* of a group space template given the template GUID.

Example 10–23 Retrieving Template Metadata Given the Template GUID

```
GroupSpaceWSMetadata templGuidMetadata =
client.getGroupSpaceTemplateMetadataByGuid("Guid");
System.out.println("Template Name: " + templGuidMetadata.getName());
```

10.2.4.3.6 Retrieving the WebCenter Spaces URL Use the `getWebCenterSpacesURL` API to obtain the URL of WebCenter Spaces.

Example 10–24 retrieves the URL of the currently running instance of WebCenter Spaces.

Example 10–24 Retrieving the WebCenter Spaces URL

```
String myWebCenterSpacesURL = client.getWebCenterSpacesURL();
```

10.2.4.3.7 Retrieving a Group Space URL Use the `getGroupSpaceURL` API to obtain the URL of a specific group space. This is useful for when you want to construct a hyperlink for a group space or you have a relative URL that you need to make into an absolute URL. To use this API, specify the name of the group space.

Example 10–25 retrieves the URL of the *Databases* group space.

Example 10–25 Retrieving a Group Space URL

```
String myGroupSpaceURL = client.getGroupSpaceURL("Databases");
```

Example 10–26 prints a list of group spaces as hyperlinks.

Example 10–26 Printing a List of Group Spaces as Hyperlinks

```
//get the list of group spaces
List<String> spaces = client.getGroupSpaces("");
//print the list of group spaces as hyperlinks
for (String spaceName : spaces)
{
  print("<a href =""" + client.getGroupSpaceURL(spaceName) + "">" + spaceName +
"</a><br>");
}
```

To construct the URL of a particular group space page, retrieve the group space URL and then add the page information. For information about how to create pretty URLs for group space pages, see "Using Pretty URLs for WebCenter Spaces Pages" in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

10.2.5 How to Handle Exceptions Raised by WebCenter Spaces APIs

The `GroupSpaceWSException` class provides APIs for handling exceptions raised by the group space APIs.

Table 10–5 Group Space APIs in the GroupSpaceWSException Class

API	Class	Description
`getLocalizedMessage`	GroupSpaceWSException	Composes the localized error message.
`printStackTrace`	GroupSpaceWSException	Prints the exception and its back trace to the standard error stream.

This section includes the following subsections:

- Providing Localized Error Messages
- Listing the Error Stack

10.2.5.1 Providing Localized Error Messages

Sometimes you may find that the default error messages provided by the APIs are not specific enough for your particular application. In these cases you can provide your own error messages.

Use the `getLocalizedMessage` API to compose application-specific error messages.

Example 10–27 shows a servlet that includes code to create a group space. If any exceptions are raised during the creation process, a localized message is printed.

Example 10–27 Printing a Localized Error Message

```
servlet1.java

doGet()
{
```

```
...
  print("<b>Output</b>");
  try
    {
      GroupSpaceWSClient client = new GroupSpaceWSClient(context);
      client.createGroupSpace("Databases", "Databases, "A community for people
interested in databases", "databases, oracle", "CommunityofInterest");
      print("Successfully created group space");
    }
  catch(GroupSpaceWSException ex)
    {
      if(ex instanceof GroupSpaceNameNullException)
        print(ex.getLocalizedMessage());
      else if (ex instanceof GroupSpaceDescNullException)
        print(ex.getLocalizedMessage());
    }
...
}
```

10.2.5.2 Listing the Error Stack

For debugging purposes, it is often useful to see which errors ultimately led to the failure of a particular operation to discover the underlying cause of the problem. Use the `printStackTrace` API to list all the errors that caused a particular exception.

Example 10–28 prints the exception and all the errors leading up to it.

Example 10–28 Listing the Error Stack

```
servlet1.java

doGet()
{
...
  print("<b>Output</b>");
  try
    {
      GroupSpaceWSClient client = new GroupSpaceWSClient(context);
      client.createGroupSpace("Databases", "Databases", "A community for people
interested in databases", "databases, oracle", "CommunityofInterest");
      print("Successfully created group space");
    }
  catch(GroupSpaceWSException ex)
    {
      ex.printStackTrace();
    }
...
}
```

10.2.6 Finding Further Information

API Reference Documentation

For detailed syntax information for each API, see the *Oracle Fusion Middleware WebCenter Group Spaces Java API Reference*.

10.3 Exposing Enterprise Applications in WebCenter Spaces

You can expose various enterprise applications inside WebCenter Spaces, including:

- Custom WebCenter applications—applications developed using the WebCenter Framework.
- Other portals in the WebCenter Suite—for example, Oracle Portal.
- Oracle Applications Unlimited products—for example, E-Business Suite, Siebel, PeopleSoft, and JDEdwards.
- Other custom applications—applications developed using non-Oracle platforms.

Technologies such as WSRP, Oracle JSF Portlet Bridge, ADF task flows, and WebCenter support for external applications, enable WebCenter Spaces to consume and present application data in a unified way. The following sections tell you how.

This section includes the following subsections:

- Exposing Custom WebCenter Applications in WebCenter Spaces
- Exposing Oracle Applications in WebCenter Spaces
- Exposing Non-Oracle Applications in WebCenter Spaces

10.3.1 Exposing Custom WebCenter Applications in WebCenter Spaces

You can expose custom WebCenter applications, built using the WebCenter Framework, as portlets and task flows in WebCenter Spaces:

- **Portlets:** Web components that are deployed inside a container and generate dynamic content. For more information, see Chapter 27, "Overview of Portlets."
- **Oracle JSF Portlet Bridge:** A technology that enables you to easily expose existing ADF applications and task flows as JSR 168 portlets. For more information, see Chapter 28, "Creating Portlets with the Oracle JSF Portlet Bridge."
- **Task flows:** ADF navigational components that define the transition between states and activities using call methods on managed beans, evaluating an EL expression, and so on. For more information see, "Getting Started with ADF Task Flows" in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

10.3.2 Exposing Oracle Applications in WebCenter Spaces

You can expose Oracle Applications Unlimited products, for example, E-Business Suite, Siebel, PeopleSoft, and JDEdwards, in WebCenter Spaces, using:

- **Web services and ADF:** Enterprise applications like Siebel expose various web service interfaces. You can leverage these web service interfaces to build data controls, ADF task flows, and portlets that can be consumed in WebCenter Spaces.
- **Prebuilt portlets:** Enterprise applications such as Oracle E-Business Suite ship portlets out of the box that you can register with WebCenter Spaces and consume in the same way as any other portlet.

10.3.3 Exposing Non-Oracle Applications in WebCenter Spaces

You can expose web applications developed using non-Oracle platforms in WebCenter Spaces, as follows:

- **WebCenter support for external applications:** External applications enable credential stores to pass mapped user identities to web applications that require their own authentication. For more information, see "Registering External Applications" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter* and "Working with External Applications" in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.
- **Oracle Composer iframe-level integration:** The Oracle Composer *Web Page* page style and component display any web page content, including pages from applications built using a different web technology. For more information, see "Working with Page Layouts, Styles, and Schemes" in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.
- **Raw HTML markup injection in Oracle Composer:** The Oracle Composer *HTML Markup* layout component injects HTML or JavaScript snippets into WebCenter Spaces pages. You can use this component to display content from an external source. For more information see "Working with HTML Markup Layout Component Properties" in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.
- **OmniPortlet:** A portlet that publishes data from various data sources using a variety of layouts. For more information, see Chapter 31, "Creating Portlets with OmniPortlet."

Part III

Integrating Services

Part III contains the following chapters:

11

Preparing Your Application for Oracle WebCenter Web 2.0 Services

This chapter describes how to prepare your application to consume Oracle WebCenter Web 2.0 services. It contains the following sections:

- Section 11.1, "Preparing Your WebCenter Application to Consume Services"
- Section 11.2, "Extending Your Application with Custom Components"
- Section 11.3, "Configuring General Settings for Your Services"

Note: For important basic information about WebCenter applications, such as how to create an application, how to create customizable pages, and how to implement security, see Chapter 3, "Preparing Your Development Environment."

For conceptual information about WebCenter Web 2.0 services, see Section 1.1.4, "Introduction to Oracle WebCenter Web 2.0 Services."

11.1 Preparing Your WebCenter Application to Consume Services

This section describes the steps you must take to prepare your application to use Oracle WebCenter Web 2.0 services. It contains the following subsections:

- Section 11.1.1, "How to Prepare Your Application to Consume Services"
- Section 11.1.2, "Setting Up an External Application Connection"
- Section 11.1.3, "Automated Task Flow Grants"

11.1.1 How to Prepare Your Application to Consume Services

You can configure any application to include a WebCenter Web 2.0 Service. When you create an application in JDeveloper, you can choose to base the application on a template. Although not a requirement, the WebCenter Application template makes all the appropriate WebCenter connection wizards and tag libraries readily visible and available in the New Gallery and Component palette. When you consume a WebCenter Services task flow or component, the necessary libraries are automatically added to the project.

Depending upon the service you plan to consume, your application must meet certain prerequisites. For example, if the service needs to know the identity of users, then your application must provide some level of security with user authentication.

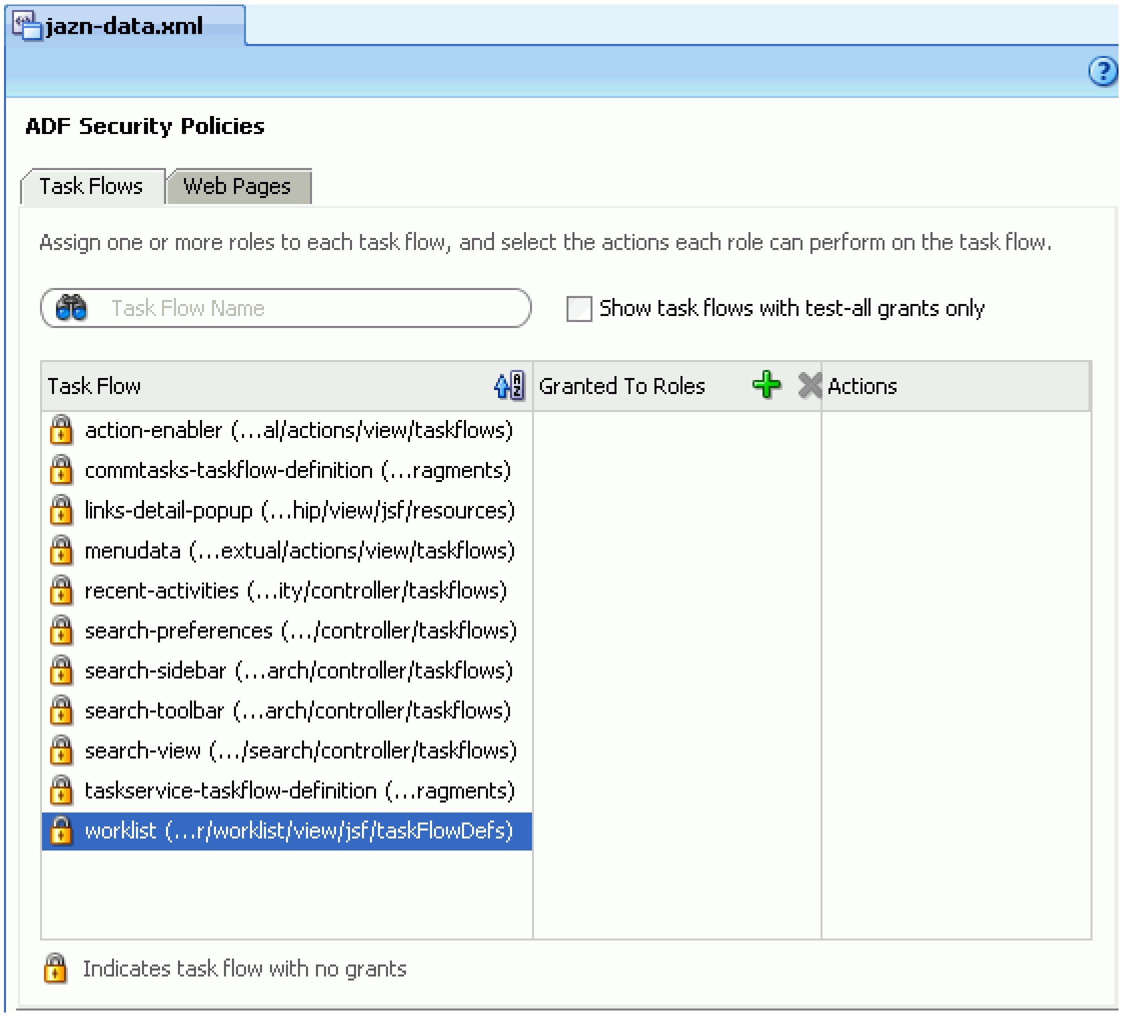
11.1.1.1 Implementing Security for Services

Some services must know the identity of the user (for example, the Search service needs the user's identity for saving searches). For these services, you must at least configure your application to authenticate users such that they have distinct identities for the purposes of personalization and user preferences.

For details on how to implement a basic security solution for your WebCenter custom application, see Section 3.5.1, "How to Configure ADF Security." For details on how to implement a complete security solution for your custom application, see Chapter 24, "Securing Your WebCenter Application."

Once you configure ADF security for your application, you can open the `jazn-data.xml` file and modify your sample user's privileges for each task flow. To open the ADF Security Policies Editor, locate the file in the Application Resources pane and double-click its name. You will see the default task flow grants, as shown in Figure 11–1. You can further configure these grants in this view.

Figure 11–1 ADF Security Policies in the jazn-data.xml file



Assign one or more roles to each task flow, and select the actions each role can perform on the task flow.

Show task flows with test-all grants only

Task Flow	Granted To Roles	Actions
action-enabler (...al/actions/view/taskflows)		
commtasks-taskflow-definition (...ragments)		
links-detail-popup (...hip/view/jsf/resources)		
menudata (...extual/actions/view/taskflows)		
recent-activities (...ity/controller/taskflows)		
search-preferences (.../controller/taskflows)		
search-sidebar (...arch/controller/taskflows)		
search-toolbar (...arch/controller/taskflows)		
search-view (.../search/controller/taskflows)		
taskservice-taskflow-definition (...ragments)		
worklist (...r/worklist/view/jsf/taskFlowDefs)		

Indicates task flow with no grants

11.1.2 Setting Up an External Application Connection

When a WebCenter Web 2.0 service interacts with an application that handles its own authentication, you can associate that application with an external application definition to allow for credential provisioning.

The following WebCenter Web 2.0 services permit the use of an external application to connect with the service and define authentication for it:

- Documents
- Instant Messaging and Presence
- Mail
- RSS Viewer (when using a secured RSS feed)

For more information about setting up the external application for a WebCenter Web 2.0 service, see Chapter 14, "Integrating the Documents Service," Chapter 15, "Integrating the Instant Messaging and Presence Service," Chapter 17, "Integrating the Mail Service," and Chapter 19, "Integrating the RSS Service." For more information about external applications in general, see Section 24.2, "Working with External Applications."

11.1.3 Automated Task Flow Grants

Table 11–1 lists the grants that are given when a task flow from the corresponding service is consumed. Common grants are granted whenever any Oracle WebCenter Web 2.0 service is consumed.

Table 11–1 Automated Task Flow Grants

Service	Grant	Actions	Name (Target)
Common	TaskFlowPermission	View	/oracle/webcenter/framework/service/controller/taskflows/resourceViewer.xml#resourceViewer
Announcement	TaskFlowPermission	View	/oracle/webcenter/collab/announcement/view/taskflows/main-view-definition.xml#announcement-main-view /oracle/webcenter/collab/announcement/view/taskflows/mini-view-definition.xml#announcement-mini-view /oracle/webcenter/collab/announcement/view/taskflows/link-existing-view-definition.xml#link-existing-view-definition /oracle/webcenter/collab/announcement/view/taskflows/config-view-definition.xml#announcement-config-view /oracle/webcenter/collab/announcement/view/taskflows/announcement-view-definition.xml#announcement-resource-view
	RegionPermission	View	oracle.webcenter.collab.announcement.view.pageDefs.oracle_webcenter_collab_announcement_view_jsf_pages_AnnouncementLinkExistingViewPageDef oracle.webcenter.collab.announcement.view.pageDefs.oracle_webcenter_collab_announcement_view_jsf_pages_AnnouncementLinkResourceViewPageDef

Table 11–1 (Cont.) Automated Task Flow Grants

Service	Grant	Actions	Name (Target)
Discussions	TaskflowPermission	View	/oracle/webcenter/collab/forum/view/taskflows/main-task-flow.xml#forum-main /oracle/webcenter/collab/forum/view/taskflows/miniview-task-flow.xml#forum-miniview /oracle/webcenter/collab/forum/view/taskflows/popularTopic-task-flow.xml#forum-popularTopic /oracle/webcenter/collab/forum/view/taskflows/recentTopic-task-flow.xml#forum-recentTopic /oracle/webcenter/collab/forum/view/taskflows/watchedTopic-task-flow.xml#forum-watchedTopic /oracle/webcenter/collab/forum/view/taskflows/watchedForum-task-flow.xml#forum-watchedForum /oracle/webcenter/collab/forum/view/taskflows/config-task-flow.xml#forum-config-view /oracle/webcenter/collab/forum/view/taskflows/link-existing-task-flow.xml#forum-link-existing /oracle/webcenter/collab/forum/view/taskflows/link-new-task-flow.xml#forum-link-new /oracle/webcenter/collab/forum/view/taskflows/message-task-flow.xml#forum-message /oracle/webcenter/collab/forum/view/taskflows/resource-view-task-flow.xml#forum-resource-view /oracle/webcenter/collab/forum/view/taskflows/scope-config-task-flow.xml#forum-scope-config-view
	RegionPermission	View	oracle.webcenter.collab.forum.view.pageDefs.oracle_webcenter_collab_forum_view_jsf_pages_ForumLinkExistingViewPageDef oracle.webcenter.collab.forum.view.pageDefs.oracle_webcenter_collab_forum_view_jsf_pages_ForumLinkNewViewPageDef oracle.webcenter.collab.forum.view.pageDefs.oracle_webcenter_collab_forum_view_jsf_pages_ForumLinkResourceViewPageDef
Documents	TaskflowPermission	View	/oracle/webcenter/doclib/view/jsf/taskflows/mainView.xml#doclib-document-library /oracle/webcenter/doclib/view/jsf/taskflows/docListViewer.xml#doclib-document-list-viewer /oracle/webcenter/doclib/view/jsf/taskflows/recentDocuments.xml#doclib-recent-documents
	RegionPermission	View	oracle.webcenter.doclib.view.pageDefs.oracle_webcenter_doclib_view_jsf_fragments_mainViewPageDef oracle.webcenter.doclib.view.pageDefs.oracle_webcenter_doclib_view_jsf_pages_linkToExistingDocumentPageDef oracle.webcenter.doclib.view.pageDefs.oracle_webcenter_doclib_view_jsf_pages_linkToNewDocumentPageDef
External Application	TaskflowPermission	View	/oracle/adfinternal/extapp/view/fragments/extapp-credential-provisioning-taskflow.xml#extapp-credential-provisioning-taskflow /oracle/adfinternal/extapp/view/fragments/extapp-change-password-taskflow.xml#extapp-change-password-taskflow

Table 11–1 (Cont.) Automated Task Flow Grants

Service	Grant	Actions	Name (Target)
Instant Messaging and Presence			
Links	TaskflowPermission	View	/oracle/webcenter/relationship/view/jsf/resources/links-detail.xml#links-detail /oracle/webcenter/relationship/view/jsf/resources/links-detail-popup.xml#links-detail-popup
	RelationshipPermission	Manage	*
Mail	TaskflowPermission	View	/oracle/webcenter/collab/mail/view/jsf/regions/compose-task-flow.xml#mail-compose-view /oracle/webcenter/collab/mail/view/jsf/regions/content-view-definition.xml#mail-content-view /oracle/webcenter/collab/mail/view/jsf/regions/dl-config-definition.xml#dl-config-view /oracle/webcenter/collab/mail/view/jsf/regions/mini-view-definition.xml#mail-mini-view
	RegionPermission	View	oracle.webcenter.collab.mail.view.pageDefs.oracle_webcenter_collab_mail_view_jsf_pages_ComposeviewPageDef oracle.webcenter.collab.mail.view.pageDefs.oracle_webcenter_collab_mail_view_jsf_pages_ContentViewPageDef
Page Editor	TaskflowPermission	View	/oracle/webcenter/.* /oracle/adfinternal/pageeditor/.*
	RegionPermission	View	oracle.webcenter.*
Page	TaskFlowPermission	View	/oracle/webcenter/page/view/jsf/fragments/page-create-page.xml#page-create-page /oracle/webcenter/page/view/jsf/fragments/page-doc-prop-panel-definition.xml#page-doc-prop-panel-definition /oracle/webcenter/page/view/jsf/fragments/page-doc-sec-panel-definition.xml#page-doc-sec-panel-definition /oracle/webcenter/page/view/jsf/fragments/golink-prop-panel-definition.xml#golink-prop-panel-definition
People	TaskFlowPermission	View	/oracle/webcenter/people/view/jsf/regions/people-contacts.xml#people-contacts
Recent Activities	TaskFlowPermission	View	/oracle/webcenter/recentactivity/controller/taskflows/recent-activities.xml#recent-activities
RSS	TaskFlowPermission	View	/oracle/webcenter/rssviewer/view/jsf/fragments/RSSViewerTaskFlow.xml#RSSViewerTaskFlow

Table 11–1 (Cont.) Automated Task Flow Grants

Service	Grant	Actions	Name (Target)
Search	TaskFlowPermission	View	/oracle/webcenter/search/controller/taskflows/searchResults.xml#search-view
			/oracle/webcenter/search/controller/taskflows/localToolbarSearch.xml#search-toolbar
			/oracle/webcenter/search/controller/taskflows/preferences.xml#search-preferences
			/oracle/webcenter/search/controller/taskflows/allSavedSearches.xml#all-saved-searches
			/oracle/webcenter/search/controller/taskflows/simpleSearchResults.xml#search-simple-view
			/oracle/webcenter/search/controller/taskflows/customize.xml#search-customize
Tags	TaskFlowPermission	View	/oracle/webcenter/tagging/controller/taskflows/related-links.xml#tagging-related-links
			/oracle/webcenter/tagging/controller/taskflows/launch-dialog.xml#tagging-launch-dialog
			/oracle/webcenter/tagging/controller/taskflows/tagging-personal-view.xml#tagging-personal-view
			/oracle/webcenter/tagging/controller/taskflows/tag-selection.xml#tag-selection
			/oracle/webcenter/tagging/controller/taskflows/related-resources.xml#related-resources
			/oracle/webcenter/tagging/controller/taskflows/tag-center-task-flow.xml#tag-center
			/oracle/webcenter/tagging/controller/taskflows/tag-center-related-tags.xml#tag-center-related-tags
			/oracle/webcenter/tagging/controller/taskflows/tag-center-related-users.xml#tag-center-related-users
	RegionPermission	View	oracle.webcenter.tagging.view.pageDefs.oracle_webcenter_tagging_view_jsf_fragments_launch_dialogPageDef
			oracle.webcenter.tagging.view.pageDefs.oracle_webcenter_tagging_view_jsf_fragments_tag_centerPageDef
			oracle.webcenter.tagging.view.pageDefs.oracle_webcenter_tagging_view_jsf_fragments_tag_center_related_resourcesPageDef
			oracle.webcenter.tagging.view.pageDefs.oracle_webcenter_tagging_view_jsf_fragments_tag_center_related_tagsPageDef
			oracle.webcenter.tagging.view.pageDefs.oracle_webcenter_tagging_view_jsf_fragments_tag_center_related_usersPageDef
			oracle.webcenter.tagging.view.pageDefs.oracle_webcenter_tagging_view_jsf_fragments_tag_center_tag_selectionPageDef
Worklist	TaskFlowPermission	View	/oracle/webcenter/worklist/view/jsf/taskFlowDefs/worklist.xml#worklist

In addition to the services in Table 11–1, grants are also given when the `External Application - Change Password` task flow is applied when creating an external application in Oracle JDeveloper. Specifically, the `TaskflowPermission` is given and placed in `/oracle/adfinternal/extapp/view/fragments/.*`.

11.2 Extending Your Application with Custom Components

This section introduces the Resource Action Handling Framework and describes how to register a resource viewer. It contains the following subsections:

- Section 11.2.1, "Introducing the Resource Action Handling Framework"
- Section 11.2.2, "Registering a Resource Viewer"

11.2.1 Introducing the Resource Action Handling Framework

Custom components are just like out-of-the-box WebCenter Web 2.0 services in that they manage and own resources. As such, they must make declarations that enable their resources to be accessible to the WebCenter Web 2.0 services that invoke other services (that is, the Search, Tags, Links, and Recent Activities services). WebCenter provides a *Resource Action Handling framework* for services that expose resources to be viewed, searched, and tagged.

For example, the Resource Action Handling framework enables the Search service to look up and follow the mechanism to render a resource for any given service. Other facilities of the Resource Action Handling framework allow for authorization of resources to be declared and enforced when access is made.

The following services invoke resource viewers using the Resource Action Handling framework:

- Search
- Tags
- Links
- Recent Activities

The following services own task flows that can be invoked by the Resource Action Handling framework.

- Announcements
- Discussions
- Documents (Resource URL rewriter to point to GET handler)
- Events
- Mail
- Notes
- Page (Resource URL rewriter to point to page URL)
- Tags - Tag Center

11.2.2 Registering a Resource Viewer

You must register a resource viewer to enable custom resources to be rendered using Search or Tags, or to make the resources linkable to and from each other.

In the `service-definition.xml` file, you can define a resource viewer for any service to render that service's resources. The viewer can be a task flow or a URL rewriter acting on the resource ID.

For example, suppose a discussion forum message is found through the Search service. Clicking the result link launches the resource viewer that was declared for the

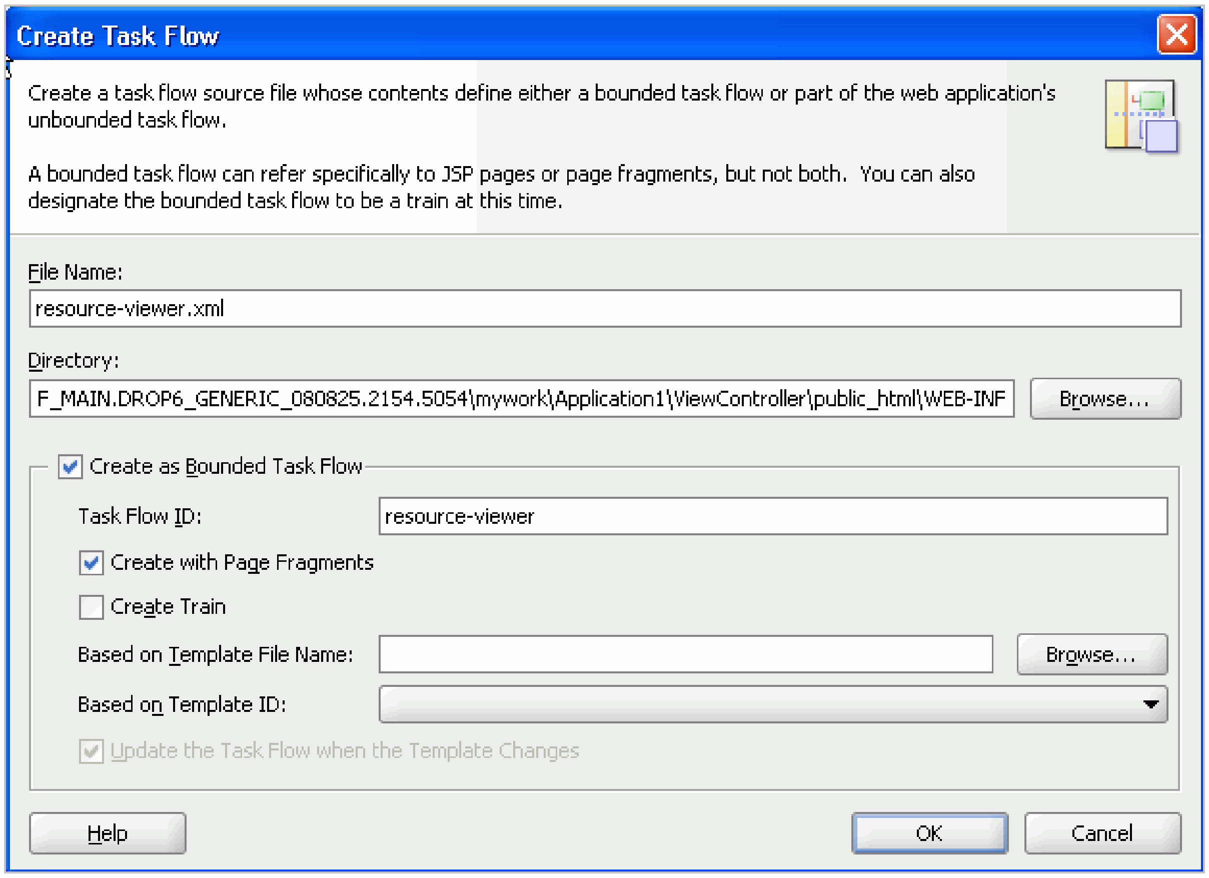
Discussions service (by looking up the `service-definition.xml` of the Discussions service). By default, the target is rendered in a dialog.

Note: When you add a Search, Tags, or Recent Activities task flow to your application, that service creates a sample `service-definition.xml` file. This file enables developers to declare their resource viewers as well as their implementations of interfaces for interacting with the Links and Search services; for example, the QueryManager interface implementation.

To register a custom resource viewer:

1. In the Application Navigator, right-click the **Page Flows** node for the **ViewController** project and select **New** from the context menu.
2. Under **Categories**, select **JSF**, then **ADF Task Flow**.
3. Click **OK**.
4. In the Create ADF Task Flow dialog, change **File Name** to `resource-viewer.xml`. The Task Flow ID should automatically update to `resource-viewer` (Figure 11–2).

Figure 11–2 Create ADF Task Flow Dialog for Resource Viewer



Create Task Flow

Create a task flow source file whose contents define either a bounded task flow or part of the web application's unbounded task flow.

A bounded task flow can refer specifically to JSP pages or page fragments, but not both. You can also designate the bounded task flow to be a train at this time.

File Name: resource-viewer.xml

Directory: F_MAIN.DROP6_GENERIC_080825.2154.5054\mywork\Application1\ViewController\public_html\WEB-INF

[✓] Create as Bounded Task Flow

Task Flow ID: resource-viewer

[✓] Create with Page Fragments

[ ] Create Train

Based on Template File Name:

Based on Template ID:

[✓] Update the Task Flow when the Template Changes

5. Click **OK**.

Ensure that the `resource-viewer.xml` file is created under `..\public_html\WEB-INF`.

6. From the Component Palette, make sure that ADF Task Flow is selected and drag **View** onto the task flow to create a page fragment.
7. Rename the page fragment to `resource-viewer`, and click in an open area of the canvas to accept the change.
8. Define a JSFF file for your page fragment.
9. Double-click the page fragment.
10. Click **OK** in the Create New JSF Page Fragment dialog.
11. Go to the Source view in the editor and paste an `af:outputText` tag inside of the `jsp:root` element.

    Example 11–1 provides a very simple resource viewer. You could create a much more complex viewer.

Example 11–1 Simple Resource Viewer

```
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich">
  <af:outputText value="Rendering #{pageFlowScope.resourceId}"
                 inlineStyle="font-size:5"/>
</jsp:root>
```

12. Add a `resourceId` input parameter to the task flow, and set the value to `#{pageFlowScope.resourceId}`.
13. After the task flow is defined, you must register it in the `service-definition.xml` file. In the Application Navigator, expand **Application Resources**, **Descriptors** and **ADF META-INF**.
14. Double-click `service-definition.xml` to open it in the editor.
15. In the file, you will find a service definition that has been commented out. Uncomment it by removing the `<!--` and `-->` tags around it.
16. Set the `id` attribute to the service identifier.

    For example, the service identifier used when defining the Tagging Button could be `mycompany.myproduct.myapp.mycomponent`.
17. Set the `taskFlowId` to `/WEB-INF/resource-viewer.xml#resource-viewer`.

    This identifier is the task flow path and file name concatenated with the task flow identifier declared in the Create ADF Task Flow dialog. You can omit the remaining tags.
18. The `<resource-view>` declaration has a class specification for the resource authorizer. This provides various ways to specify the resource view and determine which tagged objects are visible (and searchable) to users.

Example 11–2 Specifying a Task Flow ID in a resource-view Declaration

```
<resource-view taskFlowId="/somepath/somefile.xml#someId"
               authorizerClass="some.service.id.Authorizer1"/>
```

The Search, Tags, and Discussions services use this type of declaration. Instead of having a view ID (JSPX page), the service supplies a task flow to render the resource.

- The attribute `taskFlowId` signifies that the renderer is a task flow.
- A task flow is used to render the resource specified by the resource ID.
- The task flow honors a `resourceId` input parameter and uses that to render the resource.

Example 11–3 Specifying a URL Rewriter Class Name (1)

```
<resource-view
urlRewriterClass="oracle.webcenter.framework.internal.resource.IdentityResourceUrl
Rewriter"
               authorizerClass="some.service.id.Authorizer5"/>
```

or

Example 11–4 Specifying a URL Rewriter Class Name (2)

```
<resource-view
urlRewriterClass="oracle.webcenter.framework.internal.resource.MessageFormatResour
ceUrlRewriter"
               authorizerClass="some.service.id.Authorizer4">
  <parameters>
    <!-- This starts from the context root which would be /webcenter -->
    <parameter name="message" value="/content/conn/UCM/path/{0}"/>
  </parameters>
</resource-view>
```

or

Example 11–5 Specifying a URL Rewriter Class Name (3)

```
<resource-view
urlRewriterClass="oracle.webcenter.framework.internal.resource.MyResourceUrlRewrit
er"
               authorizerClass="some.service.id.Authorizer6">
  <parameters>
    <parameter name="myParam" value="myValue"/>
  </parameters>
</resource-view>
```

Services, such as Oracle SES and Documents, use a URL rewriter to treat the resource ID as a URL and rewrite the URL in an appropriate fashion. A URL rewriter changes the resource ID into a URL that can be launched with a `goLink`. Two `ResourceUrlRewriters` are provided:

- `oracle.webcenter.framework.internal.resource.IdentityResourceUrlRewriter` is for Oracle SES. It takes the `resourceId` value as a URL as is.
- `oracle.webcenter.framework.internal.resource.MessageFormatResourceUrlRewriter` is for Documents. It takes in a `message` parameter (in the `service-definition`). The `message` parameter contains a {0}, which is replaced with the value of the `resourceId`. The `urlExternal` parameter applies to all URL rewriters. This parameter's value determines whether the JSPX that consumes a resource action handler link should open a new window.

The default behavior is to launch a new window dialog with that URL.

Some services, like the Page service, can have no resource viewer but may allow dubbing their resource IDs as page view IDs (Example 11–6).

Example 11–6 No Resource Viewer (Resource ID as Page View ID)

```
<resource-view authorizerClass="some.service.id.Authorizer3"/>
```

The resource authorizer interface makes use of Java interfaces and classes to return useful information about service resources. For example, the `oracle.webcenter.framework.resource.ResourceAuthorizer` interface contains a method that returns the status (available or not available) for a unique resource ID (Example 11–7).

Example 11–7 Method for Returning Status Against a Unique Resource ID

```
/*
 * Check if the passed in resourceIds are viewable by the current user.
 *
 * @param resourceId - the resourceId that we want to check
 * @return - a <code>ResourceInfo</code> with a ResourceStatus
 * of RESOURCE_IS_AVAILABLE if the passed in resourceId
 * is viewable by current user, otherwise returns
 * <code>RESOURCE_IS_NOT_AVAILABLE</code>.
 */
ResourceInfo getResourceInfo(String resourceId, Scope scope);

/*
 * Check if the passed in resourceIds are viewable by the current user.
 *
 * @param resourceIds - the resourceIds that we want to check
 * @return - a List of <code>ResourceInfo</code> with a ResourceStatus
 * of RESOURCE_IS_AVAILABLE if the passed in resourceId
 * is viewable by current user, otherwise returns
 * <code>RESOURCE_IS_NOT_AVAILABLE</code>.
 */
List<ResourceInfo> getResourceInfo(List<String> resourceIds, Scope scope);
```

The `oracle.webcenter.framework.resource.ResourceInfo` interface exposes the following method:

```
public ResourceStatus getResourceStatus();
```

The `oracle.webcenter.framework.resource.ResourceInfo.ResourceStatus` enumeration includes the following possible values:

Example 11–8 Possible ResourceStatus Values

```
/**
 * Used to describe a related object's status.
 *
 * Certain operations may be performed by relationship service depending
 * on what object status it receives from the service.  For example,
 * if it receives {@link #RESOURCE_DOES_NOT_EXIST} it may remove all
 * relationships associated with this object from the relationship schema
 * so future calls are not wasted on this object.
 */
 public enum ResourceStatus
 {
    /**
     * Used to describe a resource that is available.
     */
    RESOURCE_IS_AVAILABLE,

    /**
     * Used to describe a resource that does not exist.
     */
    RESOURCE_DOES_NOT_EXIST,

    /**
     * Should be used when the user does not have view access on the resource.
     */
    USER_DOES_NOT_HAVE_ACCESS;
  }
```

The only method in the `oracle.webcenter.framework.resource.ResourceUrlRewriter` interface writes the URL, after a unique identifier is passed in. This URL can be an absolute URL or it can be one relative to the faces context root of the owning application.

Example 11–9   Resource URL Rewriter Method

```
public String rewriteUrl(String id);
```

19. Your service should look similar to the one shown in Example 11–10.

Example 11–10   Sample Service Definition for Resource View

```
<service-definitions>
  <service-definition id="mycompany.myproduct.myapp.mycomponent"
                      version="11.1.1.0.0">
  <resource-view taskFlowId="/somepath/somefile.xml#someId"
authorizerClass="mycompany.myproduct.myapp.mycomponent.ResourceAuthorizer"/>
  <name>My Custom Component</name>
  <description>My Custom Component's Description</description>
  <icon>/mycompany/myproduct/myapp/mycomponent/component.png</icon>
  </service-definition>
 ...
</service-definitions>
```

20. Save the `service-definition.xml` file.

11.3 Configuring General Settings for Your Services

Optionally, you can enable users to set the time zone, the date and time format, the language (locale), and the accessibility mode for the services you add to your application. You can provide these capabilities through the WebCenter General Settings Service. This service contains values that you can modify either directly from the JSF page(s) in your WebCenter application or indirectly, through an API. The WebCenter General Settings Service covers five areas:

- Time Zone: By default, displays the time defined by the server running the WebCenter application.
- Date-Time Format: By default, configured to **java.text.DateFormat.SHORT**.
- Language: By default, defined by the locale of the user's browser.

- Accessibility Mode: A value to use with the `accessibility-mode` setting in the `trinidad-config.xml` file.
- Skin Family: A value to use with the `skin-family` setting in the `trinidad-config.xml` file.

To use the service directly from your application, you can use Expression Language in your JSF pages to access the `generalSettings` managed Bean. When you add `generalSettings` as the value attribute of an ADF component, for example `af:activeOutputText`, you can open the Expression Builder, then navigate to **JSF Managed Beans > generalSettings** to view the preference values for the bean.

Figure 11–3 Expression Builder for the generalSettings JSF Managed Bean

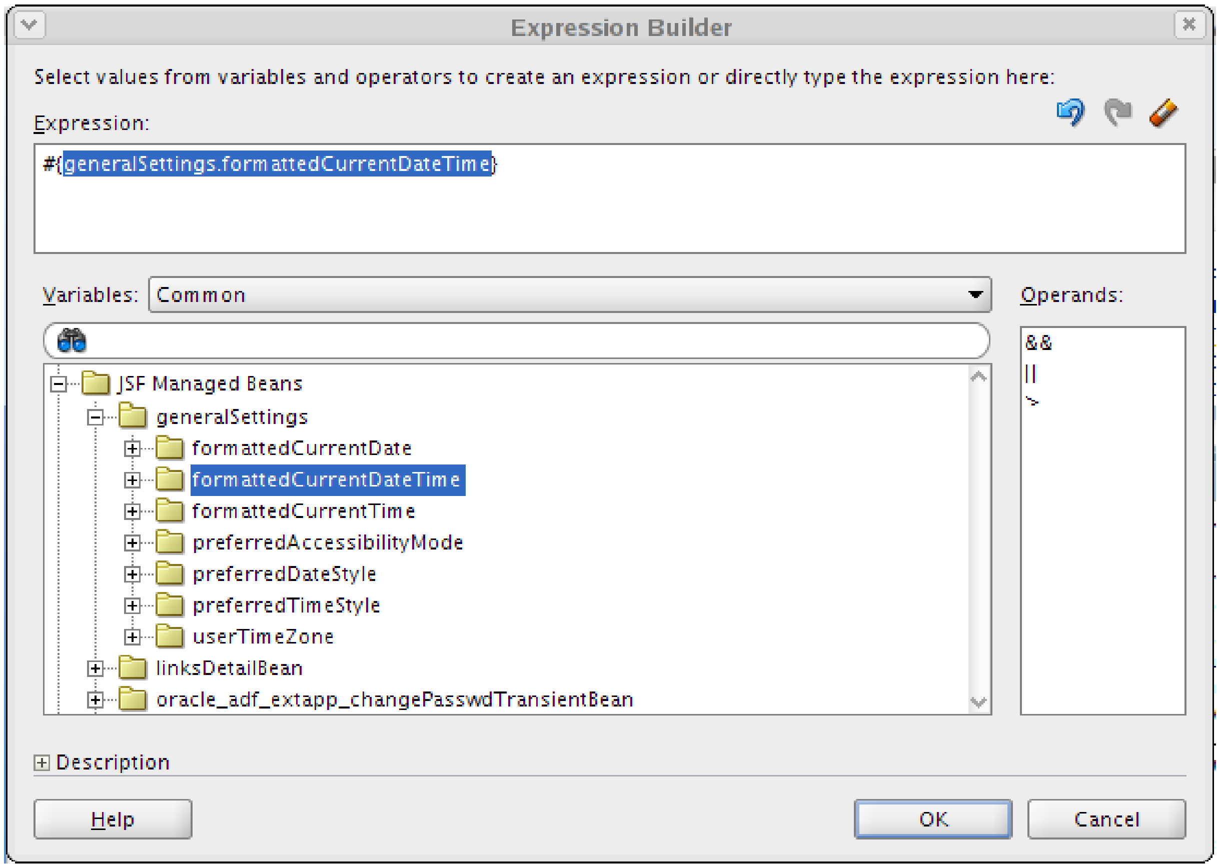


Table 11–2 describes the seven different preference values for the General Settings service.

Table 11–2 General Settings Managed Bean Preference Values and Descriptions

General Settings Preference Value	Description
`formattedCurrentDate`	Displays the current date in the user's selected locale.
`formattedCurrentDateTime`	Displays the current date and time in the user's selected locale.
`formattedCurrentTime`	Displays the current time in the user's selected locale.
`preferredAccessibilityMode`	Displays the preferred accessibility mode (`default`, `inaccessible`, or `screenReader`).

Table 11–2 (Cont.) General Settings Managed Bean Preference Values and Descriptions

General Settings Preference Value	Description
`preferredDateStyle`	Java date style to be used in `dateStyle` attributes when displaying dates and times.
`preferredTimeStyle`	Java time style to be used in `timeStyle` attributes when displaying dates and times.
`userTimeZone`	Java timezone to be used in `timeZone` attributes when displaying dates and times.

For example, to display the current date and time in the user's selected locale, time zone, and format, add the following to your page:

Example 11–11 Code for Displaying Current Date and Time in User's Locale

```
<af:outputText value="#{generalSettings.formattedCurrentDateTime}"/>
```

Or, to display a specific date and time:

Example 11–12 Code for Displaying Specified Date and Time

```
<af:outputText value="#{row.dateTimeValue}">
    <af:convertDateTime type="both"
                        dateStyle="#{generalSettings.preferredDateStyle}"
                        timeStyle="#{generalSettings.preferredTimeStyle}"
                        timeZone="#{generalSettings.userTimeZone}"
                        locale="#{facesContext.externalContext.requestLocale}" />
    </af:outputText>>
```

To take advantage of the preferred accessibility mode, you must add an EL (Expression Language) expression to the application's `trinidad-config.xml` file to set the accessibility mode, for example:

Example 11–13 Setting the Accessibility Mode in trinidad-config.xml

```
<accessibility-mode>#{generalSettings.preferredAccessibilityMode}</accessibility-m
ode>
```

You can also build a user interface using the General Settings API to enable users to either accept the default settings or apply their desired settings. The API contains the same seven preference values as described in Table 11–2.

Any WebCenter Web 2.0 service that displays a date and time uses the settings configured in the General Settings service. For more information on using the General Settings Service API to build a General Settings User Interface, see the Javadoc for `oracle.webcenter.generalsettings`.

12 Integrating the Announcements Service

This chapter explains how to integrate the Announcements service into a custom WebCenter application.

This chapter includes the following sections:

12.1 Introduction to Announcements

The Announcements service lets you create and expose announcements on your application pages. Access to announcements boosts community participation, problem resolution, and knowledge sharing.

Announcements can show either forum-level announcements or global announcements (that is, announcements not specific to a forum).

12.1.1 Understanding the Announcements Service

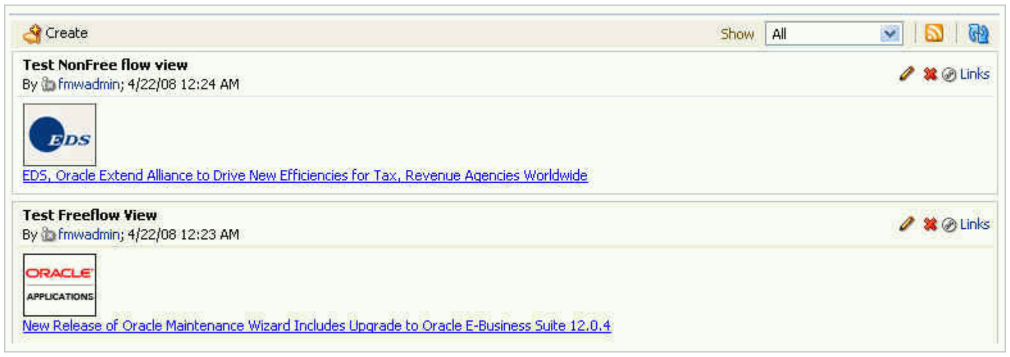
With the Announcements service, you can do the following:

- Create an announcement by clicking the **Create** icon, and then send it to a distribution list.
- Edit an existing announcement by clicking the **Edit** icon next to the announcement.
- Delete an existing announcement by clicking the **Delete** icon next to the announcement.
- See the latest announcement data by clicking the **Refresh** icon on the Anouncement view.
- Personalize the Announcements view. You can select to display announcements sent today, this week, this month, or all announcements. A **Show** list provides options for displaying announcements created within the selected time range.
- Manage announcements, such as publishing announcements on a later date and specifying auto-expire on certain dates.

The Announcements service is integrated with many WebCenter Web 2.0 services, such as the RSS, Search (to search announcement text), Instant Messaging and Presence (IMP), and Recent Activities services. Use the Links service to link announcements to other services, such as Discussions.

Figure 12–1 shows a sample announcement.

Figure 12–1 Sample Announcement From the Task Flow



Test NonFree flow view
By fmwadmin; 4/22/08 12:24 AM
EDS, Oracle Extend Alliance to Drive New Efficiencies for Tax, Revenue Agencies Worldwide

Test Freeflow View
By fmwadmin; 4/22/08 12:23 AM
New Release of Oracle Maintenance Wizard Includes Upgrade to Oracle E-Business Suite 12.0.4

For more information about the service at runtime, see *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

Note: You can see the Announcements service in action in the sample application, as described in Chapter 2, "Introduction to the WebCenter Sample Application."

12.1.2 Requirements for Announcements

The Announcements service requires a discussions server. You must install and configure the discussions server that comes with Oracle Fusion Middleware.

For information on installing the discussions server, see *Oracle Fusion Middleware Installation Guide for Oracle WebCenter*.

12.2 Basic Configuration for the Announcements Service

This section describes required steps for adding this service to your application. It contains the following subsections:

- Section 12.2.1, "Setting Up a Connection for Announcements"
- Section 12.2.2, "Adding the Announcements Service at Design Time"
- Section 12.2.3, "Setting Security for Announcements"

12.2.1 Setting Up a Connection for Announcements

To take advantage of the Announcements service, you must first create a connection to the discussions server from your custom WebCenter application. To do so, ensure that you have the connection information for the discussions server.

12.2.1.1 Announcements Connections

The Announcements service requires a WebCenter Discussion connection to the discussions server. When you create a WebCenter Discussion connection or set a connection as active, both the Announcements and Discussions services will use this same connection. If you already have an existing connection, then you can skip this section and see Section 12.2.2, "Adding the Announcements Service at Design Time."

If you do not have an existing connection, then you must create a new WebCenter Discussion connection.

Note: While you can set up the connections to back-end servers at design time in Oracle JDeveloper, you can later add, delete, or modify connections in your deployed environment using Enterprise Manager Fusion Middleware Control. For more information, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

12.2.1.2 How to Set Up Connections for Announcements

To set up the Announcements connection:

1. In Oracle JDeveloper, open the application in which you plan to consume Announcements.

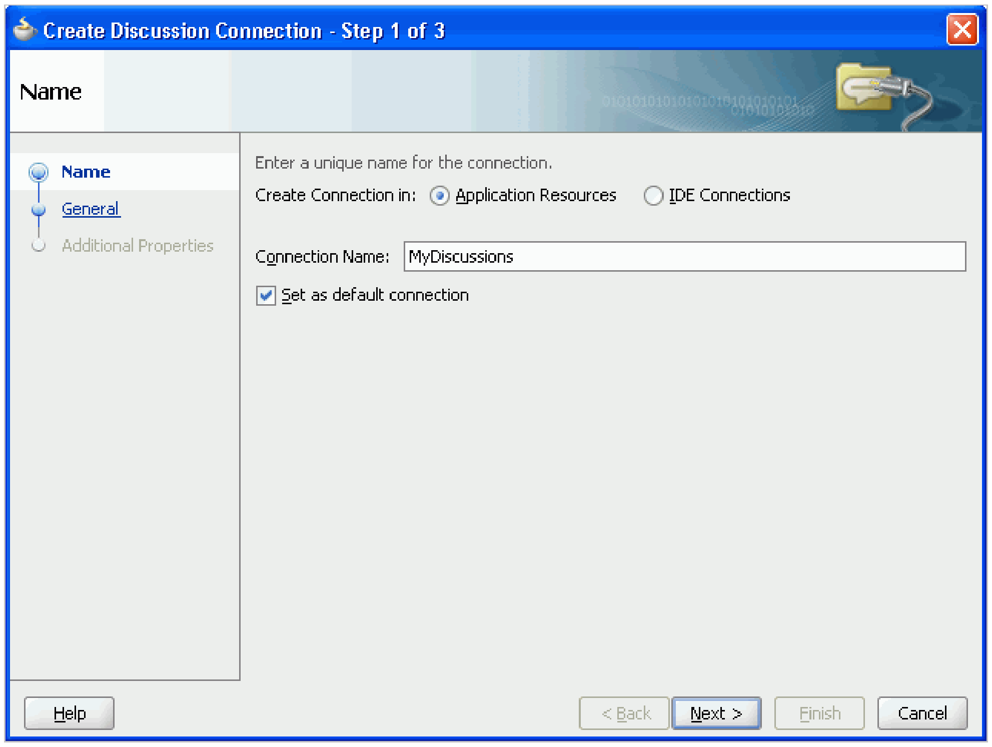
   Note: If you already created a WebCenter Discussion connection for the Discussions service, then that will be used by default for the Announcements service. No extra configuration is required.

   ---

2. In the Resource Palette, click the **New** icon and select **New Connection**, **WebCenter Discussion Connection**.
3. On the **Name** page, select to create the connection in **Application Resources**. (A connection in Application Resources will be available only for that application, while a connection in IDE Connections will be available for all applications you create. If you plan to use the connection in other applications, then select IDE Connections so you do not need to re-create it.)
4. For **Connection Name**, enter a meaningful name; for example, `MyDiscussions`.
5. Select the **Set as default connection** checkbox. You can have more than one connection, but only one can be active (default). If you have different discussions servers (for example, for each page in a space), then do not select the checkbox, but the service requires that one connection be marked as the active connection (Figure 12–2).

   Note: After you create a connection as the active connection, you cannot edit it so that it is *not* the default. To use a different default connection, you must create a new connection and mark that as the default connection.

Figure 12–2 Create Discussion Connections, Step 1



Enter a unique name for the connection.

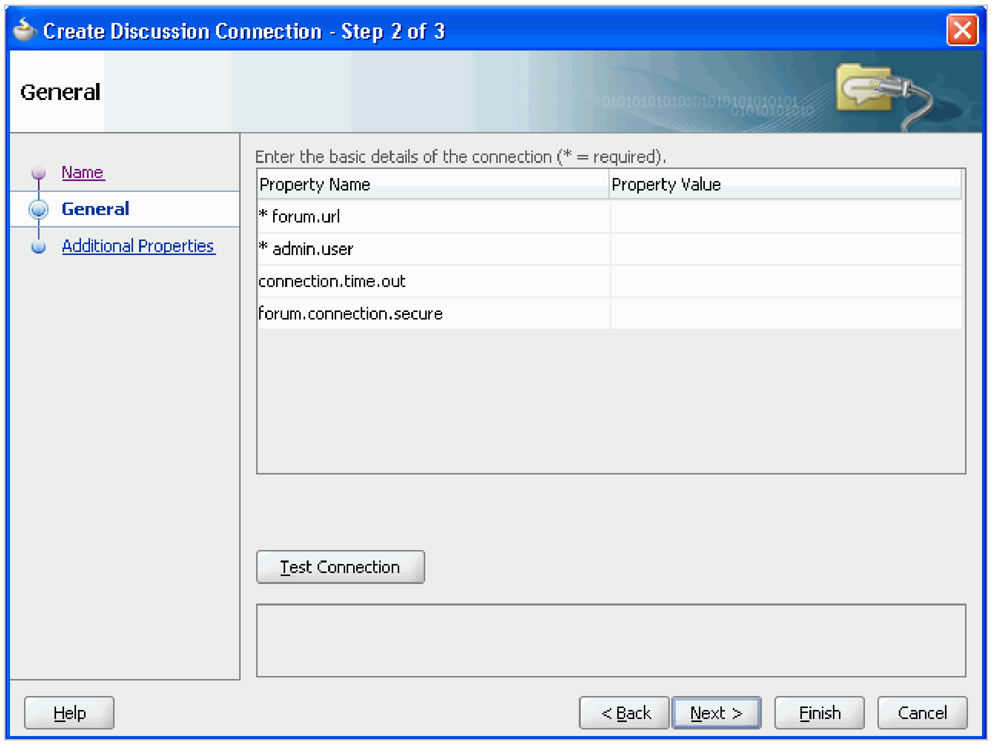
Create Connection in: [✓ Application Resources] IDE Connections

Connection Name: MyDiscussions

[✓] Set as default connection

6. Click **Next**.
7. On the **General** page, enter values for the required parameters:
   - `forum.url`: The URL for your discussions server instance. For example: `http://discussions.example.com:8888/owc_discussions`.
   - `admin.user`: The user name of your discussions server administrator; for example, `admin`. This account is used by the Discussions and Announcement services to perform administrative operations on behalf of WebCenter users.
8. Optionally, enter values for the other parameters:
   - `connection.time.out`: The time (in seconds) the service should wait for the server to respond while making the connection. If the discussions server does not respond in the given time, then it will abort the connection and report an error.
   - `forum.connection.secure`: Whether the connection should be secured. If it should be secured, then additional configuration about WS-Security is required on the next page. The accepted value for this parameter is true or false.

Figure 12–3 Create Discussions Connection, Step 2



Enter the basic details of the connection (* = required).

Property Name	Property Value
* forum.url	
* admin.user	
connection.time.out	
forum.connection.secure	

9. Click **Test Connection**, and if it is successful, then click **Next**.
10. On the **Create Discussion Connection - Step 3 of 3** page, you can enable client-side WS-Security (Figure 12–4).

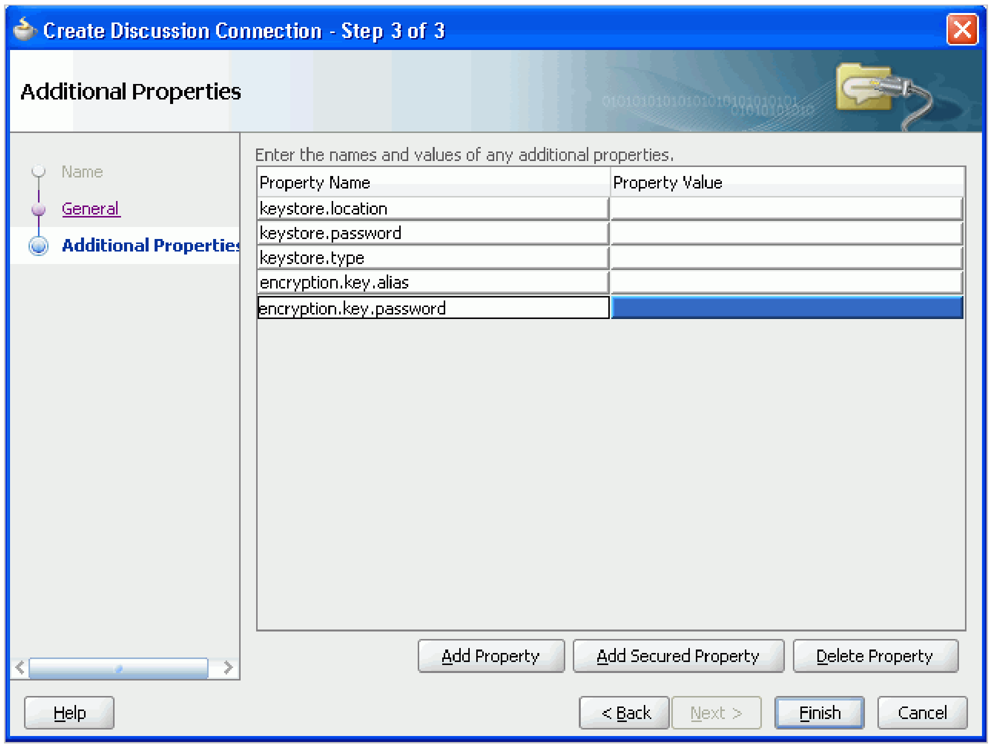
    First obtain the client-side certificate and store it in a local directory. Then click the **Add Secured Property** button to add each of the following additional parameters in your WebCenter Discussion connection to secure WS-Service calls:

    - `keystore.location`: The certificate file path in your local directory.
    - `keystore.password`: The keystore password.
    - `keystore.type`: The keystore type associated with the certificate, for example, JKS.
    - `encryption.key.alias`: The key alias to be used for encryption.
    - `encryption.key.password`: The password for accessing the encryption key.

    Make sure your property key and value are the same as in the certificate.

    Note: If values should be secured, then add them by clicking the **Add Secured Property** button.

Figure 12–4 Create Discussions Connection, Step 3



Enter the names and values of any additional properties.

Property Name	Property Value
keystore.location	
keystore.password	
keystore.type	
encryption.key.alias	
encryption.key.password	

11. Click **Finish**. Your connection should now appear as a node under **Application Resources - Connections**.

12.2.2 Adding the Announcements Service at Design Time

This section explains a basic incorporation of the Announcement service.

12.2.2.1 Announcements Task Flows

The announcement task flows let you add a main view or a sidebar view of announcements to your page.

Table 12–1 Announcement Task Flows

Task Flow	Required Parameters	Description
Announcements	`parentId`	This task flow displays a view with that allows the user to see all current announcements and perform operations based on their privileges. For a Moderator, all command buttons are shown, but for a Reader, only the refresh and personalization options are shown. The personalization option lets users select the number of days to display announcements. The `parentId` parameter is the forum ID in the discussions server under which announcements are maintained. If this parameter is not specified, then global announcements will be shown.

Table 12–1 (Cont.) Announcement Task Flows

Task Flow	Required Parameters	Description
Announcements - Sidebar View	`parentId`	This task flow displays a view that shows various categories of quick links to announcements. The `parentId` parameter is the forum ID in the discussions server under which announcements are maintained. If this parameter is not specified, then global announcements in the discussions server will be shown. The look and feel of this view changes with the optional parameter values given for rendering the task flow region. For information on how to add this task flow, see Section 12.3.1, "How to Add the Announcements - Sidebar View Task Flow."

12.2.2.2 How to Add Announcements to Your Application

The **Announcements** task flow provides a complete view of your announcements. To add the **Announcements** task flow to your custom WebCenter application:

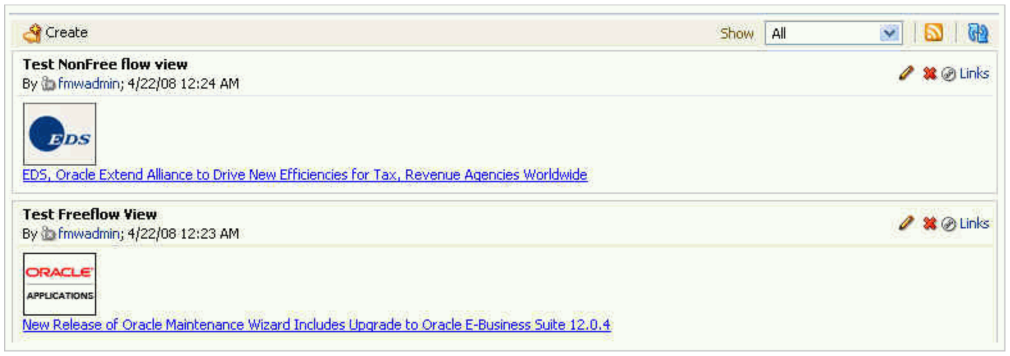
1. Follow the steps described in Chapter 3, "Preparing Your Development Environment" to implement security and create a new ustomizable page in your application.
2. Open the page on which you want to add the service.
3. In the Resource Palette, expand **IDE Connections**, **File System**, **webcenter_ services_root**, **Announcements,** and **ADF Task Flows**.
4. Drag **Announcements** from the Resource Palette and drop it onto the page inside of the `af:form` `begin` and `end` tags.
5. When prompted, select **Region** as the way to create the task flow (and confirm with **Add Library**). This operation may take a moment to complete.
6. In the Edit Task Flow Binding dialog, you can enter a value for the `parentId` parameter.
   - If global announcements are required, then do not specify a value for `parentId`.
   - If announcements from a specific forum are required (for example, when a forum is reserved for each group), then enter a value for `parentId`; for example, `${2}`.
7. Click **OK**.

   Note: If you created a connection in IDE and not in the application, then the connection must be added to the application. For example, in the Resource Palette under IDE Connections, right-click the connection and select **Add to Application**.

8. Save and run your page.

   Figure 12–5 shows the **Announcements** view at runtime. The runtime view is based on the value that was provided for the `parentId` parameter.

Figure 12–5 Announcements View at Runtime



Test NonFree flow view
By fmwadmin; 4/22/08 12:24 AM
EDS, Oracle Extend Alliance to Drive New Efficiencies for Tax, Revenue Agencies Worldwide

Test Freeflow View
By fmwadmin; 4/22/08 12:23 AM
New Release of Oracle Maintenance Wizard Includes Upgrade to Oracle E-Business Suite 12.0.4

Note: All instances of the **Announcements** task flow in an application run against the active discussions server, and it serves no purpose to add more than one **Announcements** task flow instance. This is true for all service task flows that require connections to back-end servers, such as task flows from the Discussions and Mail services.

12.2.3 Setting Security for Announcements

In a non-secured custom WebCenter application, identity propagation cannot happen. A user will be a *guest* (or *anonymous*) user who can view only public categories and forums. In a non-secured application, the `parentId` (forum ID) from which announcements are fetched should be a public forum. If the forum is not public, then an error is reported. If the `parentId` parameter is not specified, then WebCenter cannot fetch global announcements (that is, announcements not scoped to a forum): global announcements are not available for public users.

In an ADF-secured custom WebCenter application, identity propagation is enabled. Based on the identity, appropriate permissions are matched and corresponding actions are enabled.

The user name that you use to log in to the application will be used to log in to the discussions server. The recommended approach is to have the discussions server and the custom WebCenter application point to the same identity store. To simplify the login process for your custom WebCenter application, you should use the same login credentials for the application as you do for the discussions server. This way, your users can log into the application once and automatically connect to the discussions server.

Note: The Announcements services requires that the identity store be LDAP-based; that is, not file-based with `jazn-data.xml`.

For information about configuring ADF security, see Section 3.5, "Implementing Security in Your Application."

12.3 Advanced Information for the Announcements Service

This section describes optional features available with this service.

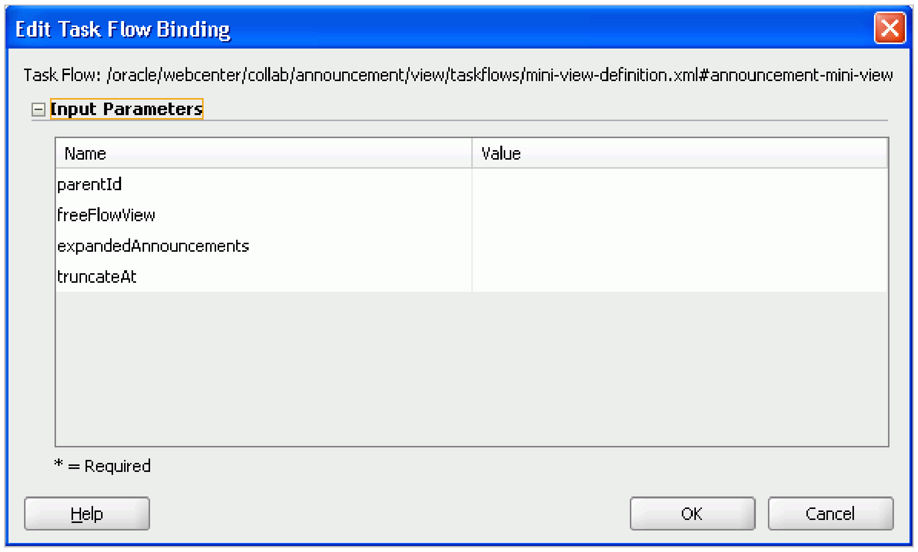
12.3.1 How to Add the Announcements - Sidebar View Task Flow

The **Announcements - Sidebar View** is similar to the **Announcements** view in that it displays announcements and provides tools to create, edit, and delete announcements. It also provides controls for determining when an announcement is published and when it expires. After you create an announcement, you can select the **Mail** icon next to the announcement to mail the announcement to anyone you choose.

The **Announcements - Sidebar View** task flow lets you present announcements to end users where manage controls are not needed. Based on the content, it can render in a default view or a custom view.

To add the **Announcements - Sidebar View** to your custom WebCenter application, follow the same instructions that you did for the **Announcements** task flow in Section 12.2.2, "Adding the Announcements Service at Design Time," but drag and drop **Announcements - Sidebar View** onto the page.

Figure 12–6 Edit Task Flow Binding Dialog for Announcements - Sidebar View



Task Flow: /oracle/webcenter/collab/announcement/view/taskflows/mini-view-definition.xml#announcement-mini-view

Input Parameters

Name	Value
parentId	
freeFlowView	
expandedAnnouncements	
truncateAt	

* = Required

Table 12–2 describes the **Announcements - Sidebar View** parameters. The look and feel of the **Announcements - Sidebar View** changes with the values provided for these parameters. For more information, see Section 12.3.2, "Customizing Announcements Views."

Table 12–2 Announcements - Sidebar View Task Flow Parameters

Parameter	Description
`parentId`	The forum ID in the discussions server under which announcement objects are maintained. For example, `${2}`. If this parameter is not specified, then announcements default to global announcements.

Table 12–2 (Cont.) Announcements - Sidebar View Task Flow Parameters

Parameter	Description
`freeFlowView`	A Boolean value representing whether to remove the announcement title (subject) and show the announcement body as is. The default value is `${false}`, which means that the announcement list displays the title and body in plain text and is controlled by the `truncateAt` and `expandedAnnouncements` parameters. Enter `${true}` to remove the announcement title and ignore the values for `truncateAt` and `expandedAnnouncements`.
`expandedAnnouncements`	The number of announcements to display announcement details in plain text (that is, the body of the announcement). Users can click the title to display the full announcement content in rich text mode. The value you enter for `expandedAnnouncements` is ignored if `freeFlowView` is set to `true`.
`truncateAt`	For announcements that display announcement body in plain text, this value specifies how many characters to display for each announcement. Enter an Expression Language (EL) expression. For example, when the value is set to the EL expression `${50}`, following their titles, announcements display no more than 50 characters. Users can click announcement titles to display the full announcement. If no value is specified, then it will display 200 characters. If a non-valid positive integer value is specified, then it will display all characters in plain text. This parameter takes effect in conjunction with `expandedAnnouncements`. The value you enter for `truncateAt` is ignored if `freeFlowView` is set to `true`.

12.3.2 Customizing Announcements Views

You can change the look and feel of the **Announcements - Sidebar View** with the parameter values. This section shows two examples of this.

Figure 12–7 shows the **Announcements - Sidebar View** at runtime with the `freeFlowView` parameter set to `${true}`.

Figure 12–7 Announcements - Sidebar View with freeFlowView Parameter

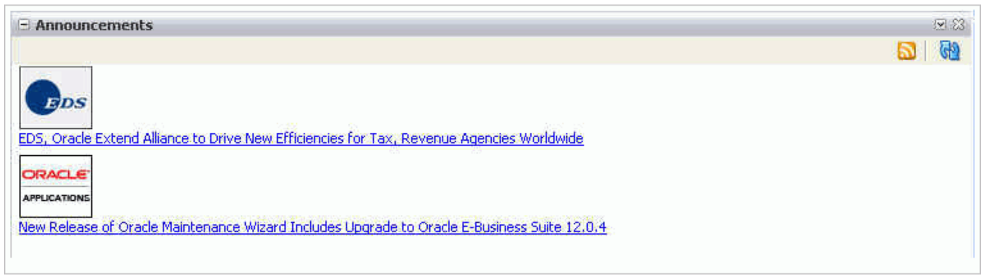
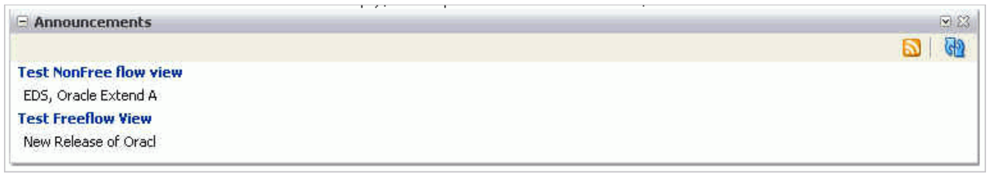

Announcements
EDS, Oracle Extend Alliance to Drive New Efficiencies for Tax, Revenue Agencies Worldwide
New Release of Oracle Maintenance Wizard Includes Upgrade to Oracle E-Business Suite 12.0.4

Figure 12–8 shows the **Announcements - Sidebar View** at runtime with the `freeFlowView` parameter set to false (or empty), the `expandedAnnouncements` parameter set to 2, and the `truncateAt` parameter set to 20.

Figure 12–8 Announcements - Sidebar View with Optional Parameters



Test NonFree flow view

EDS, Oracle Extend A

Test Freeflow View

New Release of Oracl

13 Integrating the Discussions Service

This chapter explains how to integrate the Discussions service in a custom WebCenter application.

This chapter includes the following sections:

13.1 Introduction to Discussions

The Discussions service lets you expose discussion forums on your application pages. Users can use the Discussions service to create forums, post questions, and search for answers.

For information about the Discussions taxonomy with categories, forums, topics, and messages, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

13.1.1 Understanding the Discussions Service

With the Discussions service, you can do the following (according to your permissions):

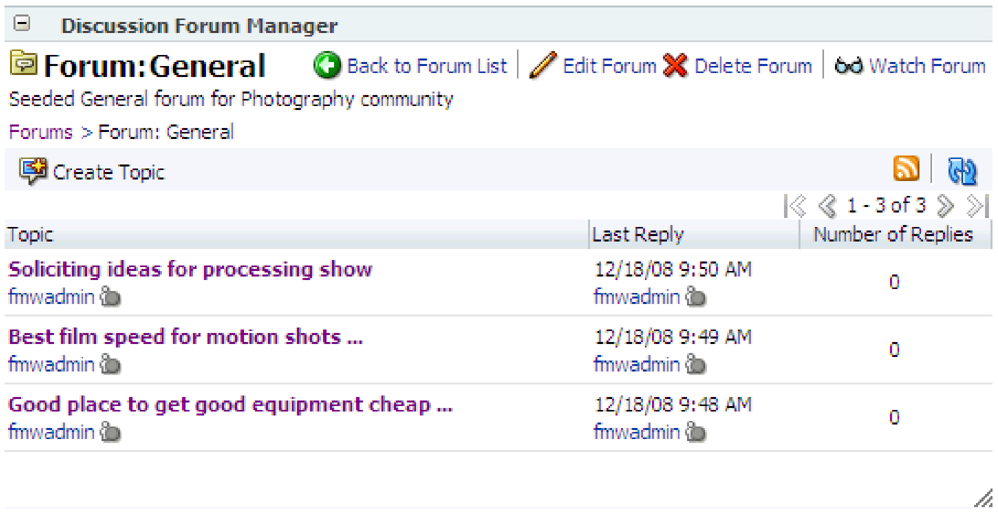
- Create a new discussion forum or a new topic.
- Navigate into a forum from a list of available forums.
- Edit an existing discussion forum, topic, or message by clicking the **Edit** icon.
- Delete a forum, topic, or message by clicking the **Delete** icon.
- Reply to an existing topic or message by clicking the **Reply** icon.
- View the number of replies for a topic on the main forum view.
- Drill into a topic to read all replies by clicking the topic name.
- Add a watch to a topic or forum by clicking the **Watch** icon.
- View the following in the sidebar:
  - Watched topics
  - Watched forums
  - Most popular topics
  - Most recent topics

The Discussions service is integrated with many WebCenter Web 2.0 services, such as Instant Messaging and Presence, RSS, and Search (to search within forums) services. Use the Mail service to archive mails into discussions as threads. Use the Links service to link to a discussion from any WebCenter object, or, link to a WebCenter object.

Note: In a secured custom WebCenter application, Discussions permissions are allotted according to an individual user's assigned user role. A user can be a moderator, participant, or viewer. Some activities require moderator or participant roles. For more information, see Section 13.2.3, "Setting Security for Discussions."

Figure 13–1 shows a sample discussion forum topic list.

Figure 13–1 Discussion Forum Topics List


Discussion Forum Manager

Forum: General — Back to Forum List | Edit Forum | Delete Forum | Watch Forum

Seeded General forum for Photography community

Forums > Forum: General

Create Topic

1 - 3 of 3

Topic	Last Reply	Number of Replies
Soliciting ideas for processing show fmwadmin	12/18/08 9:50 AM fmwadmin	0
Best film speed for motion shots ... fmwadmin	12/18/08 9:49 AM fmwadmin	0
Good place to get good equipment cheap ... fmwadmin	12/18/08 9:48 AM fmwadmin	0

For more information about the service at runtime, see *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

Note: You can see the Discussions service in action in the sample application, as described in Chapter 2, "Introduction to the WebCenter Sample Application."

13.1.2 Requirements for Discussions

The Discussions service requires a discussions server. Install and configure the discussions server that comes with Oracle Fusion Middleware.

For information on installing the Discussions server that comes with Oracle Fusion Middleware, see *Oracle Fusion Middleware Installation Guide for Oracle WebCenter*.

13.2 Basic Configuration for the Discussions Service

This section describes required steps for adding the Discussions service to your application. It contains the following subsections:

- Section 13.2.1, "Setting up Connections for Discussions"

- Section 13.2.2, "Adding the Discussions Service at Design Time"
- Section 13.2.3, "Setting Security for Discussions"

13.2.1 Setting up Connections for Discussions

You must create a connection to the discussions server in your custom WebCenter application.

The Discussions service requires a WebCenter Discussion connection to the discussions server. When you create a WebCenter Discussion connection or set a connection as active, both the Announcements and Discussions services will use this same connection. If you already have an existing connection, then you can skip this section and see Section 13.2.2, "Adding the Discussions Service at Design Time."

If you do not have an existing connection, then you must create a new WebCenter Discussion connection.

13.2.1.1 Discussions Connections

The Discussions service requires a WebCenter Discussion connection to the discussions server.

Note: While you can set up the connections to back-end servers at design time in Oracle JDeveloper, you can later add, delete, or modify connections in your deployed environment using Enterprise Manager Fusion Middleware Control. For more information, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

13.2.1.2 How to Set Up Connections for Discussions

Follow these steps to set up the discussions connection.

1. Follow the steps in Chapter 3.5, "Implementing Security in Your Application."
2. In Oracle JDeveloper, open the application in which you plan to consume discussion forums.

   Note: If you already created a WebCenter Discussion connection for the Announcements service, then that will be used by default for the Discussions service. No extra configuration is required.

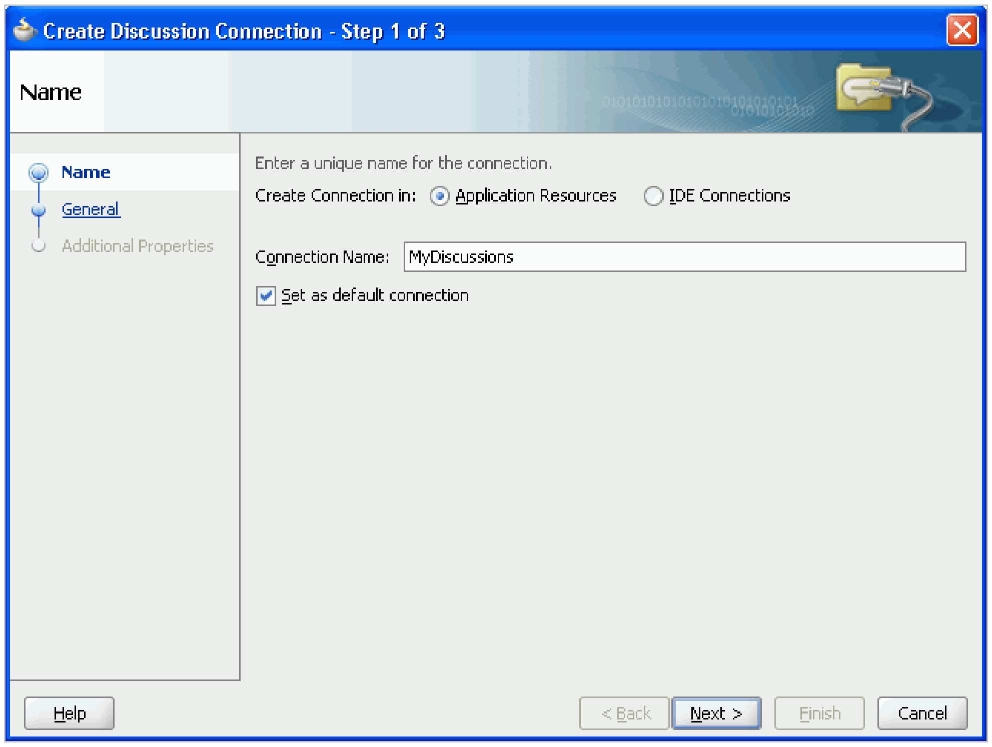
   However, if you want to use different WebCenter Discussion connection for the Announcements service and the Discussions service, then a manual entry in `adf-config.xml` is required.

3. In the Resource Palette, click the **New** icon and select **New Connection**, **WebCenter Discussion Connection**.
4. On the **Name** page, select to create the connection in **Application Resources**. (A connection in Application Resources will be available only for that application, while a connection in IDE Connections will be available for all applications you create. If you plan to use the connection in other applications, then select IDE Connections so you do not need to re-create it.)
5. For **Connection Name**, enter a unique name; for example, `MyDiscussions`.

6. Select the **Set the default connection** checkbox. You can have more than one connection, but only one can be active (default). If you have different discussions servers (for example, for each page in a space), then do not select the checkbox, but the service requires that one connection be marked as the default connection (Figure 13–2).

> **Note:** After you create a connection as the default connection, you cannot edit it so that it is *not* the default. To use a different default connection, you must create a new connection and mark that as the default connection.

Figure 13–2 Create Discussion Connections, Step 1



Enter a unique name for the connection.

Create Connection in: [✓ Application Resources] IDE Connections

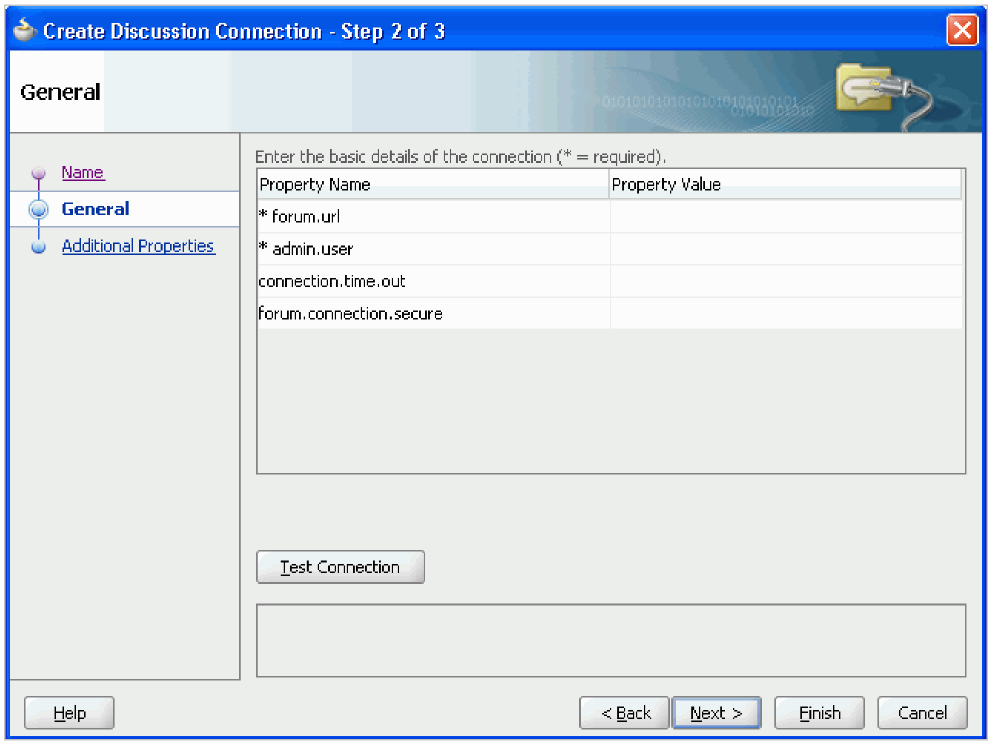
Connection Name: MyDiscussions

[✓] Set as default connection

7. Click **Next**.
8. On the **General** page, enter values for the required parameters:
   - `forum.url`: The URL for your discussions server instance. For example: `http://discuss-server.com:8888/owc_discussions`.
   - `admin.user`: The user name of your discussions server (Jive) administrator; for example, `admin`. This account is used by the Discussions and Announcement services to perform administrative operations on behalf of WebCenter users.
9. Optionally, enter values for the other parameters:
   - `connection.time.out`: The time (in seconds) the service should wait for the server to respond while making the connection. If the discussions server does not respond in the given time, then it will abort the connection and report an error.

- `forum.connection.secure`: Indicates whether a secure communication is required between the custom WebCenter application and the discussions server. The default value is false. When set to `true` (secured mode), all Web Services calls from the custom WebCenter application are sent with a user name token and client certificate.

Figure 13–3 Create Discussions Connection, Step 2



Enter the basic details of the connection (* = required).

Property Name	Property Value
* forum.url	
* admin.user	
connection.time.out	
forum.connection.secure	

For more information about options for securing your custom WebCenter application, see Section 24, "Securing Your WebCenter Application."

10. Click **Test Connection**, and if it is successful, then click **Next**.

11. On the **Create Discussion Connection - Step 3 of 3** page, you can enable client-side WS-Security (Figure 13–4).

    First obtain the client-side certificate and store it in a local directory. Then click the **Add Secured Property** button to add each of the following additional parameters in your WebCenter Discussion connection to secure WS-Service calls:

    - `keystore.location`: The certificate file path in your local directory

      Note: If the application is deployed to WebLogic Server, then this path should be accessible on the deployed server. Make sure that the path is available to the application on the deployed server.

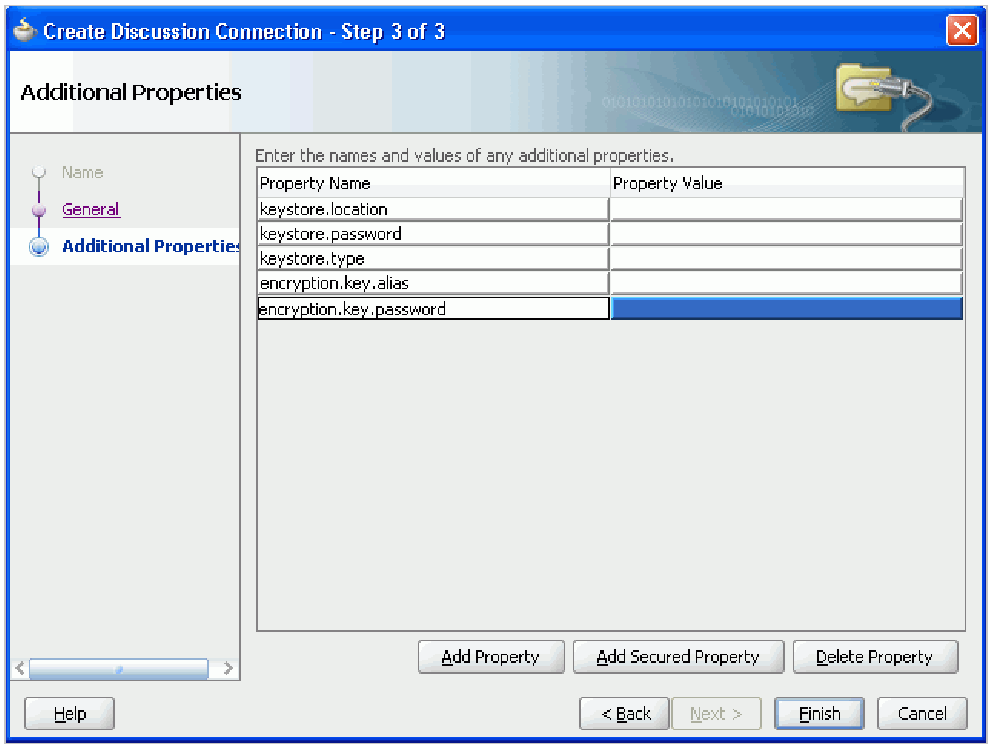
    - `keystore.password`: The keystore password.
    - `keystore.type`: The keystore type associated with the certificate. Valid values are: `jks` (Java Key Store) and `pks`.
    - `encryption.key.alias`: The key alias to be used for encryption.

- `encryption.key.password`: The password for accessing the encryption key.

Make sure your property key and value are the same as in the certificate.

Note: If values should be secured, then add them with the **Add Secured Property** button.

Figure 13–4 Create Discussions Connection, Step 3



Enter the names and values of any additional properties.

Property Name	Property Value
keystore.location	
keystore.password	
keystore.type	
encryption.key.alias	
encryption.key.password	

12. Click **Finish**. Your connection should now appear as a node under **Application Resources - Connections**.

13.2.2 Adding the Discussions Service at Design Time

This section explains a basic incorporation of the Discussions service.

13.2.2.1 Discussions Task Flows

The Discussions service provides several task flows (Table 13–1) to enable you to include the service in a form that best suits your needs.

Table 13–1 Discussions Service Task Flows

Task Flow	Description
Discussion Forums	This task flow displays the Oracle WebCenter Discussions Manager View, which allows the user to see all the discussions and their respective replies. It also allows users to perform various operations based on their privileges. A Moderator can perform create, read, update, and delete operations on all objects. A Participant can create a topic, edit a topic that has been created by him, and reply to a topic. A Viewer can only view objects. The parameters alter the way the view appears. For more information, see Section 13.3.1, "Customizing Discussions Views."
Discussions - Popular Topics	This task flow provides a view that allows users to see all the popular topics under a given category ID or forum ID. For more information, see Section 13.3.2, "Adding the Discussions - Popular Topics Task Flow."
Discussions - Recent Topics	This task flow displays a view that allows users to see all the recent topics given a category ID or forum ID. For more information, see Section 13.3.3, "Adding the Discussions - Recent Topics Task Flow."
Discussions - Watched Forums	This task flow displays a view that allows users to see all of their watched forums under a given category ID. For more information, see Section 13.3.4, "Adding the Discussions - Watched Forums Task Flow."
Discussions - Watched Topics	This task flow displays a view that allows users to see all their watched topics under a given category ID or forum ID. For more information, see Section 13.3.5, "Adding the Discussions - Watched Topics Task Flow."
Discussions - Sidebar View	This task flow displays a combined view of the Popular Topics, Recent Topics, Watched Topics, and Watched Forums task flows. Instead of adding four separate task flows, this single task flow presents all four views with a drop-down list so that end users can personalize it For more information, see Section 13.3.6, "Adding the Discussions - Sidebar View Task Flow."

13.2.2.2 How to Add Discussions to your Application

The **Discussion Forums** task flow provides a complete view of your discussions. To add the Discussion Forums task flow to your WebCenter application, follow these steps.

1. Follow the steps described in Section 11.1.1, "How to Prepare Your Application to Consume Services" to implement security and create a new customizable page in your application.
2. Open the page on which you want to add the Discussions service.
3. In the Resource Palette, expand **My Catalogs**, **WebCenter Services Catalog**, and **Task Flows**.
4. Drag **Discussions** from the Resource Palette and drop it onto the page.
5. When prompted, select **Region** as the way to create the task flow.
6. You may be prompted to add the discussions library to your project. If so, then click **Add Library**. This operation may take a moment to complete.

7. In the Edit Task Flow Binding dialog box, enter values for the parameters. Table 13–2 describes the parameters.

> **Note:** The parameters alter the way the view appears. For more information, see Section 13.3.1, "Customizing Discussions Views."

Table 13–2 Parameters for Discussion Forums Task Flow

Parameters	Description
`categoryId`	This optional parameter is an identifier for an existing category in your Oracle WebCenter Discussions instance to which the view should be scoped. When no value is supplied, it defaults to the appropriate root category of the discussions server. (This root category ID can be overridden by supplying an additional property named `application.root.category.id` in the connection.) For testing purposes, you may want to create a new category through the Oracle WebCenter Discussions administrator interface and then reference that category identifier here.
`forumId`	This parameter is an identifier for an existing forum that resides inside the given `categoryId`. It is optional. You should be certain about the specific forum for which you want to see data before entering this value.
`isCategoryView`	A means of showing the forums grouped under the Category ID category or the topics specified under the Forum ID forum. True means you want the task flow to display the forums classified under Category ID; false, the default value, means you want the task flow to display the topics associated with the specified Forum ID. This parameter value works in combination with other parameters.
`showRecursiveForums`	This parameter determines if you show forums either in a category only or in subcategories. True means all forums under a given category/subcategory are shown; false means only the category's direct child forums are shown. The default value is false. Note: A value of true can impact performance.

8. Click **OK**.

> **Note:** If you created a connection in IDE and not in the application, then the connection must be added to the application. For example, in the Resource Palette under IDE Connections, right-click the connection and select **Add to Application**.

9. Save and run your page.

> **Note:** All instances of the **Discussion Forums** task flow in an application run against the same discussions server: it is unnecessary to add more than one **Discussion Forums** task flow instance. This is true for all service task flows that require connections to back-end servers, such as task flows from the Announcements or Mail services.

The **Discussion Forums** main view contains some features that require other services in your application.

- For the links in the main view to work, you must have configured the Links service. For more information, see Chapter 16, "Integrating the Links Service."
- For users' presence indicators to work, you must have configured the Oracle WebCenter Presence service. For more information, see Chapter 15, "Integrating the Instant Messaging and Presence Service."
- Oracle recommends that the discussions server and WebCenter point to the same identity store. If the custom WebCenter application and the discussions page are configured to be secure, then upon logging in to the application, then the Discussions service respects those credentials in the discussions server.

13.2.3 Setting Security for Discussions

In a non-secured custom WebCenter application, identity propagation cannot happen. A user will be a *guest* (or *anonymous*) user who can view only public categories and forums.

In an ADF-secured custom WebCenter application, identity propagation is enabled. Based on the identity, appropriate permissions are matched and corresponding actions are enabled.

The user name that you use to log in to the application will be used to log in to the discussions server. The recommended approach is to have the discussions server and the custom WebCenter application point to the same identity store. When you run the page, you are presented with a login page for user credentials. Enter the credentials with the required privileges in the discussions server.

Note: The Discussions services requires that the identity store be LDAP-based; that is, not file-based with `jazn-data.xml`.

For information about configuring ADF security, see Section 3.5, "Implementing Security in Your Application."

13.3 Advanced Information for the Discussions Service

This section describes optional features available with this service.

13.3.1 Customizing Discussions Views

You can customize the look and feel of **Discussion Forums** views by changing parameter values. The following combinations are possible:

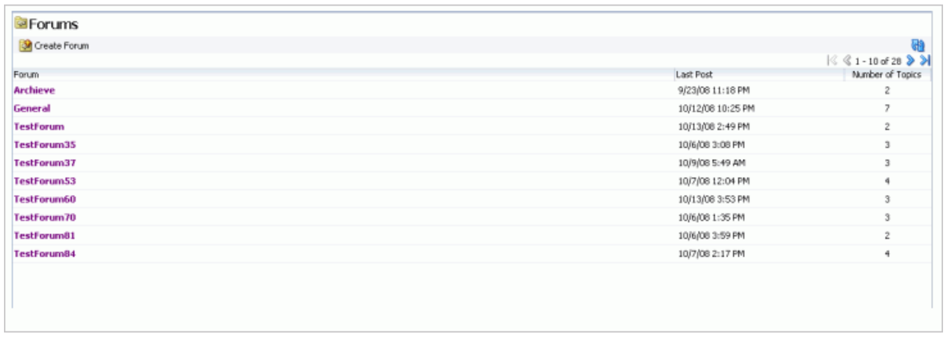
Note: A *bidirectional link* connects back and forth. For example, when you create a link from a discussion topic to a document, a link from the document back to the topic also is created. Similarly, when you delete the link from the discussion topic to a document, the link from the document back to the topic automatically is deleted.

- `categoryId`: This displays the forums list view if there is more than one forum. If there is only one forum, then it will drill into the forum and list all topics with a bidirectional link enabled.

- `categoryId` and `forumId`: This displays the topics list view with a bidirectional link enabled.
- `isCategoryView` is set to `true`: This displays the topics list view with a bidirectional link enabled.
- `categoryId` and `forumId` and `isCategoryView` is set to `false`: This displays the topics list view, but a bidirectional link is not enabled.
- `categoryId` and `isCategoryView` is set to `true`: This displays the forums list view if there is more than one forum. If there is only one forum, then it displays that forum with a bidirectional link enabled.
- `forumId` and `isCategoryView` is set to `false`: This displays the topics list view, but a bidirectional link is not enabled.
- `categoryId` and `isCategoryView` is set to `false`: This is similar to when `categoryId` alone is given.
- `forumId` and `isCategoryView` is set to `true`: This is similar to when `forumId` and `isCategoryView` is set to false.
- `isCategoryView` = `true` or `false`: This is ignored. It goes with the default scope in a custom WebCenter application.
- `forumId`: This is similar to `forumId` and `isCategoryView` is set to false.

Figure 13–5 shows the **Discussion Forums** view with the `categoryId` parameter set.

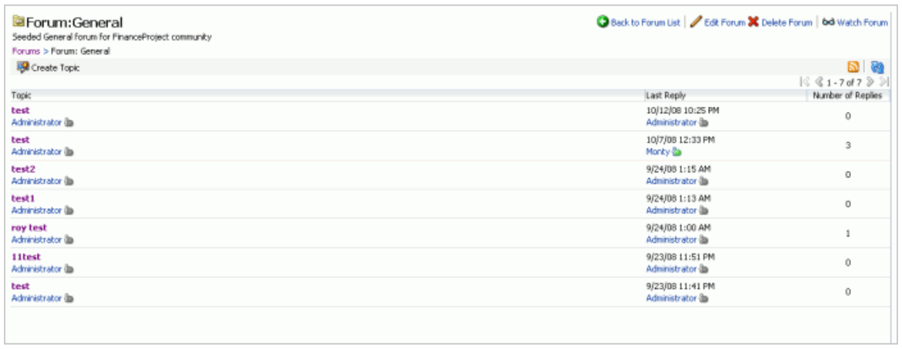
Figure 13–5   Discussions Forum View with categoryId Set



Forum	Last Post	Number of Topics
Archieve	9/23/08 11:18 PM	2
General	10/12/08 10:25 PM	7
TestForum	10/13/08 2:49 PM	2
TestForum35	10/6/08 3:08 PM	3
TestForum37	10/9/08 5:49 AM	3
TestForum53	10/7/08 12:04 PM	4
TestForum60	10/13/08 3:53 PM	3
TestForum70	10/6/08 1:35 PM	3
TestForum81	10/6/08 3:59 PM	2
TestForum84	10/7/08 2:17 PM	4

With the `categoryId` parameter set, you can click any forum or any topic, as shown in Figure 13–6 and Figure 13–7.

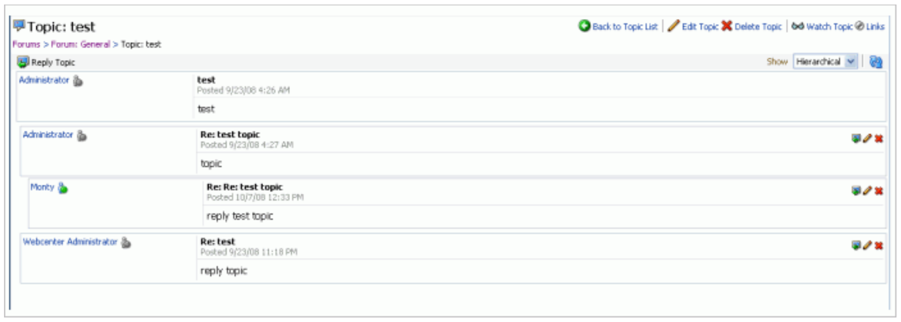
Figure 13–6   Click any Forum in the Discussion Forums View



Seeded General forum for FinanceProject community

Topic	Last Reply	Number of Replies
test Administrator	10/12/08 10:25 PM Administrator	0
test Administrator	10/7/08 12:33 PM Monty	3
test2 Administrator	9/24/08 1:15 AM Administrator	0
test1 Administrator	9/24/08 1:13 AM Administrator	0
roy test Administrator	9/24/08 1:00 AM Administrator	1
11test Administrator	9/23/08 11:51 PM Administrator	0
test Administrator	9/23/08 11:41 PM Administrator	0

Figure 13–7 Click any Topic in the Discussion Forums View



Topic: test

Forums > Forum: General > Topic: test

Administrator — **test** — Posted 9/23/08 4:26 AM — test

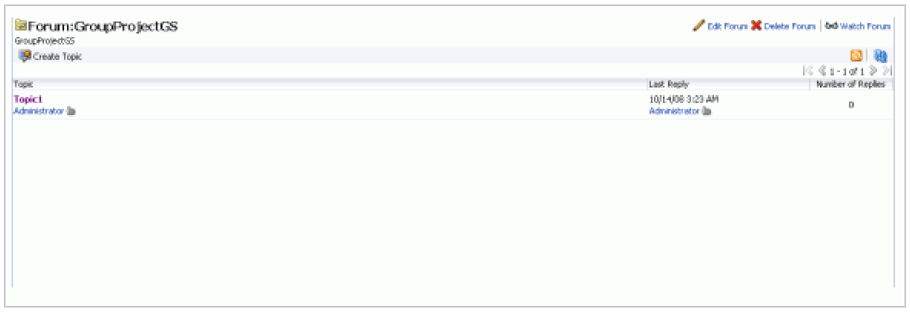
Administrator — **Re: test topic** — Posted 9/23/08 4:27 AM — topic

Monty — **Re: Re: test topic** — Posted 10/7/08 12:33 PM — reply test topic

Webcenter Administrator — **Re: test** — Posted 9/23/08 11:18 PM — reply topic

Figure 13–8 shows the **Discussion Forums** view with the `forumId` parameter set.

Figure 13–8 Discussion Forums View with the forumId Parameter Set



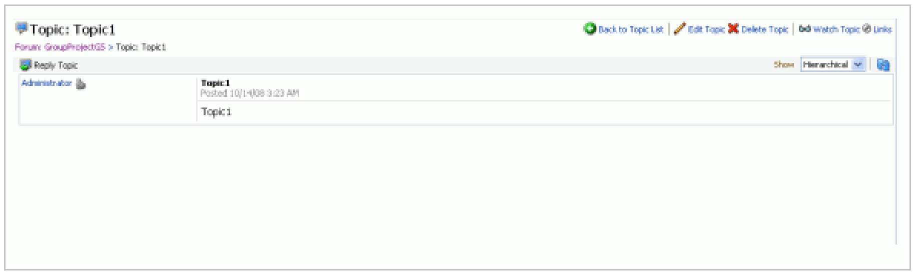
Forum:GroupProjectGS

GroupProjectGS

Topic	Last Reply	Number of Replies
Topic1 Administrator	10/14/08 3:23 AM Administrator	0

With the `forumId` parameter given, you can click any topic, as shown in Figure 13–9.

Figure 13–9 Click any Topic in the Discussion Forums View



Topic: Topic1

Forum: GroupProjectGS > Topic: Topic1

Administrator — **Topic1** — Posted 10/14/08 3:23 AM — Topic1

13.3.2 Adding the Discussions - Popular Topics Task Flow

The **Discussions - Popular Topics** task flow provides a view that allows users to see all the popular topics under a given category ID or forum ID.

To add the **Discussions - Popular Topics** task flow to your custom WebCenter application, follow the same instructions that you did for the **Discussion Forums** task flow in Section 13.2.2.2, "How to Add Discussions to your Application," but drag and drop **Discussions - Popular Topics** onto the page.

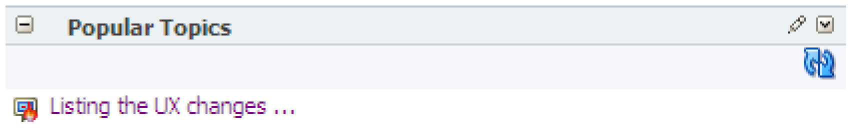
Table 13–3 describes the available parameters for this task flow.

Table 13–3 Parameters for Discussions - Popular Topics Task Flow

Parameters	Description
`categoryId`	This optional parameter is an identifier for an existing category in your Oracle WebCenter Discussions instance to which the view should be scoped. When no value is supplied, it defaults to the appropriate root category of the discussions server. (This root category ID can be overridden by supplying an additional property named `application.root.category.id` in the connection.) For testing purposes, you may want to create a new category through the Oracle WebCenter Discussions administrator interface and then reference that category identifier here.
`forumId`	This parameter is an identifier for an existing forum in your Oracle WebCenter Discussions instance for which popular topics should be fetched. If both `categoryId` and `forumId` are given, then only `categoryId` is honored.
`disableToolbar`	This parameter indicates whether the toolbar, including the Refresh icon, should be displayed. It can be `${true}`, `${false}`, or undefined. If you leave this parameter undefined, then the default behavior (false) is to show the toolbar.

Figure 13–10 shows how the **Discussions - Popular Topics** task flow looks at runtime.

Figure 13–10 Discussions - Popular Topics View at Runtime



13.3.3 Adding the Discussions - Recent Topics Task Flow

The **Discussions - Recent Topics** task flow displays a view that allows users to see all the recent topics given a category ID or forum ID.

To add the **Discussions - Recent Topics** task flow to your custom WebCenter application, follow the same instructions that you did for the **Discussion Forums** task flow in Section 13.2.2.2, "How to Add Discussions to your Application," but drag and drop **Discussions - Recent Topics** onto the page.

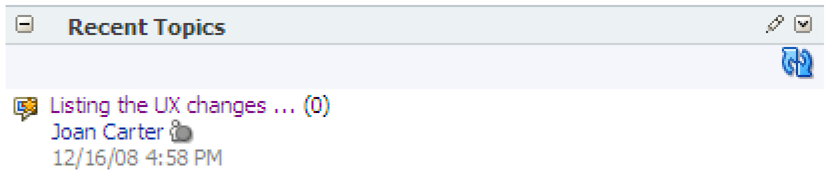
Table 13–4 describes the available parameters for this task flow.

Table 13–4 Parameters for Discussions - Recent Topics Task Flow

Parameters	Description
`categoryId`	This optional parameter is an identifier for an existing category in your Oracle WebCenter Discussions instance to which the view should be scoped. When no value is supplied, it defaults to the appropriate root category of the discussions server. (This root category ID can be overridden by supplying an additional property named `application.root.category.id` in the connection.) For testing purposes, you may want to create a new category through the Oracle WebCenter Discussions administrator interface and then reference that category identifier here.
`forumId`	This parameter is an identifier for an existing forum in your Oracle WebCenter Discussions instance for which popular topics should be fetched. If both `categoryId` and `forumId` are given, then only `categoryId` is honored.
`disableToolbar`	This parameter indicates whether the toolbar, including the Refresh icon, should be displayed. It can be `${true}`, `${false}`, or undefined. If you leave this parameter undefined, then the default behavior is to show the toolbar.

Figure 13–11 shows how the **Discussions - Recent Topics** task flow looks at runtime.

Figure 13–11 Discussions - Recent Topics View at Runtime


Recent Topics

Listing the UX changes ... (0)
Joan Carter
12/16/08 4:58 PM

13.3.4 Adding the Discussions - Watched Forums Task Flow

The **Discussions - Watched Forums** task flow displays a view that allows users to see all of their watched forums under a given category ID.

To add the **Discussions - Watched Forums** task flow to your custom WebCenter application, follow the same instructions that you did for the **Discussion Forums** task flow in Section 13.2.2.2, "How to Add Discussions to your Application," but drag and drop **Discussions - Watched Forums** onto the page.

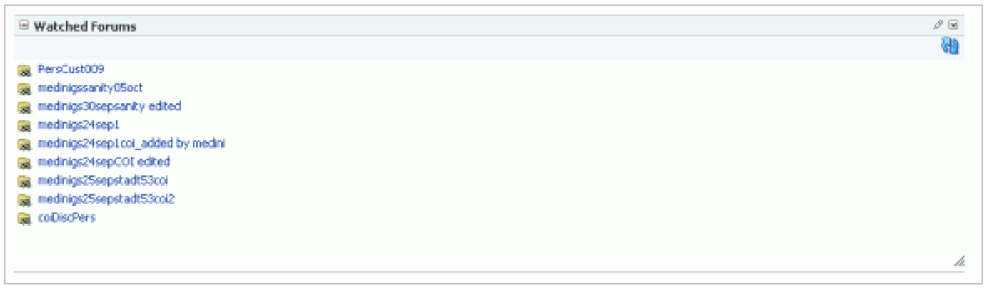
Table 13–5 describes the available parameters for this task flow.

Table 13–5   Parameters for Discussions - Watched Forums Task Flow

Parameters	Description
`categoryId`	This optional parameter is an identifier for an existing category in your Oracle WebCenter Discussions instance to which the view should be scoped. When no value is supplied, it defaults to the appropriate root category of the discussions server. (This root category ID can be overridden by supplying an additional property named `application.root.category.id` in the connection.) For testing purposes, you may want to create a new category through the Oracle WebCenter Discussions administrator interface and then reference that category identifier here.
`disableToolbar`	This parameter indicates whether the toolbar, including the Refresh icon, should be displayed. It can be `${true}`, `${false}`, or undefined. If you leave this parameter undefined, then the default behavior is to show the toolbar.

Figure 13–12 shows how the **Discussions - Watched Forums** task flow looks at runtime.

Figure 13–12   Discussions - Watched Forums View at Runtime



Watched Forums

PersCust009
medinigssanity05oct
medinigs30sepsanity edited
medinigs24sep1
medinigs24sep1coi_added by medini
medinigs24sepCOI edited
medinigs25sepstadt53coi
medinigs25sepstadt53coi2
coiDiscPers

13.3.5 Adding the Discussions - Watched Topics Task Flow

The **Discussions - Watched Topics** task flow displays a view that allows users to see all their watched topics under a given category ID or forum ID.

To add the **Discussions - Watched Topics** task flow to your custom WebCenter application, follow the same instructions that you did for the **Discussion Forums** task flow in Section 13.2.2.2, "How to Add Discussions to your Application," but drag and drop **Discussions - Watched Topics** onto the page.

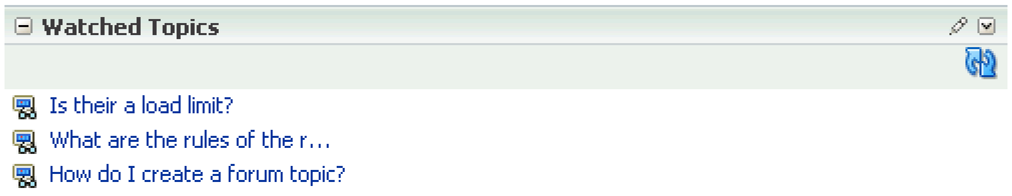
Table 13–6 describes the available parameters for this task flow.

Table 13–6 Parameters for Discussions - Watched Topics Task Flow

Parameters	Description
`categoryId`	This optional parameter is an identifier for an existing category in your Oracle WebCenter Discussions instance to which the view should be scoped. When no value is supplied, it defaults to the appropriate root category of the discussions server. (This root category ID can be overridden by supplying an additional property named `application.root.category.id` in the connection.) For testing purposes, you may want to create a new category through the Oracle WebCenter Discussions administrator interface and then reference that category identifier here.
`forumId`	This parameter is an identifier for an existing forum in your Oracle WebCenter Discussions instance for which watched topics should be fetched. If both `categoryId` and `forumId` are given, then only `categoryId` is honored.
`disableToolbar`	This parameter indicates whether the toolbar, including the Refresh icon, should be displayed. It can be `${true}`, `${false}`, or undefined. If you leave this parameter undefined, then the default behavior is to show the toolbar.

Figure 13–13 shows how the **Discussions - Watched Topics** task flow looks at runtime.

Figure 13–13 Discussions - Watched Topics View at Runtime


Watched Topics
Is their a load limit?
What are the rules of the r...
How do I create a forum topic?

13.3.6 Adding the Discussions - Sidebar View Task Flow

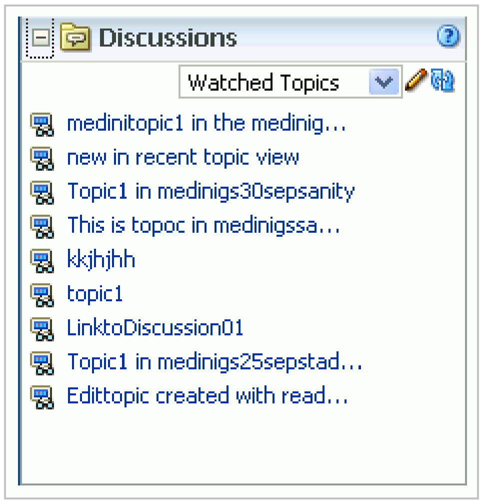
The **Discussions - Sidebar View** task flow displays a combined view of the Popular Topics, Recent Topics, Watched Topics, and Watched Forums task flows. Instead of adding four separate task flows, this single task flow presents all four views with a dropdown list so that end users can personalize it.

By default, Watched Topics are displayed. This task flow takes only one parameter: `categoryId`.

To add the **Discussions - Sidebar View** task flow to your custom WebCenter application, follow the same instructions that you did for the **Discussion Forums** task flow in Section 13.2.2.2, "How to Add Discussions to your Application," but drag and drop **Discussions - Sidebar View** onto the page.

Figure 13–14 shows how the **Discussions - Sidebar View** task flow looks at runtime.

Figure 13–14 Discussions - Sidebar View at Runtime


Discussions
Watched Topics
medinitopic1 in the medinig...
new in recent topic view
Topic1 in medinigs30sepsanity
This is topoc in medinigssa...
kkjhjhh
topic1
LinktoDiscussion01
Topic1 in medinigs25sepstad...
Edittopic created with read...

13.3.7 Using Custom Discussions APIs

Oracle WebCenter Discussions provides APIs, so you can further customize your application. To learn more about them, see the Jive Forums documentation on the WebCenter product page in the Fusion Middleware library.

13.3.8 Troubleshooting the Discussions Service

This section describes common problems and solutions for the Discussions service.

Problem

It can take several minutes to fetch data from recursive forums. For example, if you drop the **Discussions** task flow on the page with the `categoryId` and `forumId` parameters left blank and you set the `showrecursive` parameter to `${'true'}`, then it could take several minutes to scroll through all the forums.

Solution

The option to show recursive forums is not recommended for performance reasons.

Problem

Users can log on to Oracle WebCenter but cannot log on to Oracle WebCenter Discussions.

Solution

Make sure that Oracle WebCenter Discussions is configured to use the same LDAP provider as the custom WebCenter application. Contact your administrator for details. For information about configuring LDAP, see the "Configuring the Discussions Server to Share the Identity Store LDAP" section in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

Problem

When you access a discussions forum from your custom WebCenter application, the following error message is displayed:

```
No default or active connection available for: Discussion Forum
```

Solution

Ensure the following:

- Ensure that you have created a WebCenter Discussion connection.
- Ensure the required WebCenter Discussion connection is marked as the default connection.

Problem

While performing certain operations on Discussions Manager task flows, you encounter error messages like "Forum not found" or "Topic not found."

Solution

Check the following:

- Ensure that the Oracle WebCenter Discussions server is up and running.
- Ensure the forum or topic exist on the discussions server.
- Ensure that you have the required privileges for the forum or topic that you are editing.

If all of these settings are fine and the problem persists, contact your administrator for details.

14 Integrating the Documents Service

Oracle WebCenter Framework enables content integration through:

- Content Repository data controls, which enable read-only access to a content repository, and maintain tight control over the way the content displays in the application.
- The Documents service, which enables users to view and manage documents in your organization's content repositories.

This chapter describes how to integrate the Documents service in your custom WebCenter applications at design time to create a user-friendly interface for managing documents at runtime. For more information about managing and including content in your custom WebCenter applications, see:

- Chapter 8, "Integrating Content" to include content in your custom WebCenter applications at design time using JCR data controls.
- Chapter 11, "Managing Content Repositories" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter* to configure and manage content repositories used by custom WebCenter applications.
- Chapter 23, "Working with the Documents Service" in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter* to work with the Documents service and task flows at runtime in any custom WebCenter application.

This chapter includes the following sections:

- Section 14.1, "Introduction to the Documents Service"
- Section 14.2, "Basic Configuration for the Documents Service"
- Section 14.3, "Advanced Information for the Documents Service"

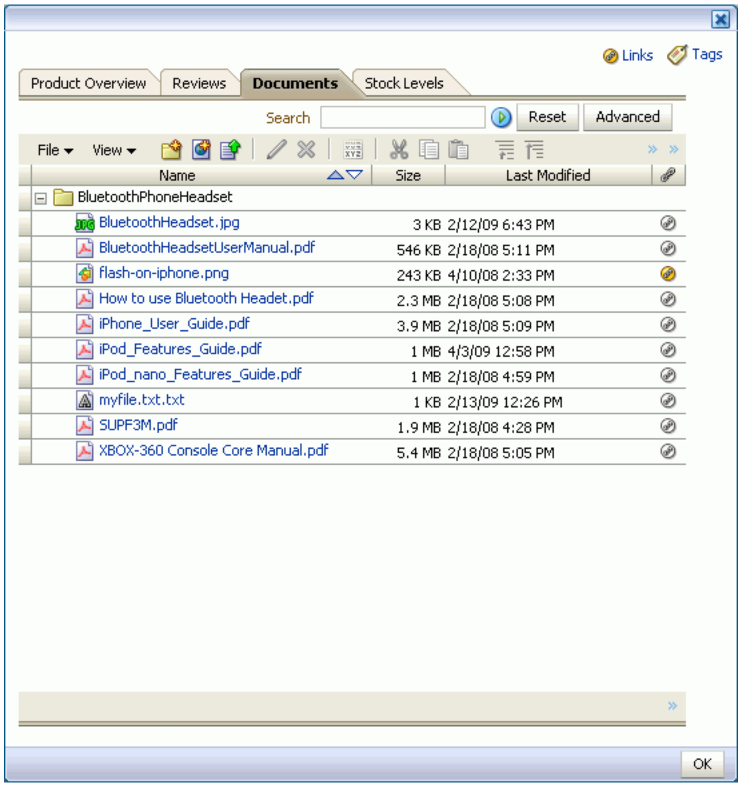
14.1 Introduction to the Documents Service

The Documents service enables users to view and manage content located in a content repository, such as Oracle Content Server, Oracle Portal, or the file system, from a custom WebCenter application.

14.1.1 Understanding the Documents Service

Depending on the content repository, the Documents service enables users to perform a variety of tasks on content in the repository. Using Oracle Content Server as the repository, for example, users can view, add, and manage files and configure file and folder properties. Figure 14–1 shows an example of one of the Documents service task flows at runtime.

Figure 14–1 Document Library View at Runtime



Name	Size	Last Modified
BluetoothPhoneHeadset		
BluetoothHeadset.jpg	3 KB	2/12/09 6:43 PM
BluetoothHeadsetUserManual.pdf	546 KB	2/18/08 5:11 PM
flash-on-iphone.png	243 KB	4/10/08 2:33 PM
How to use Bluetooth Headet.pdf	2.3 MB	2/18/08 5:08 PM
iPhone_User_Guide.pdf	3.9 MB	2/18/08 5:09 PM
iPod_Features_Guide.pdf	1 MB	4/3/09 12:58 PM
iPod_nano_Features_Guide.pdf	1 MB	2/18/08 4:59 PM
myfile.txt.txt	1 KB	2/13/09 12:26 PM
SUPF3M.pdf	1.9 MB	2/18/08 4:28 PM
XBOX-360 Console Core Manual.pdf	5.4 MB	2/18/08 5:05 PM

Note: You can see the Documents service in action in the sample application, as described in Chapter 2, "Introduction to the WebCenter Sample Application."

14.1.2 Requirements for the Documents Service

To use the Documents service, you must have a content repository that contains the documents you want to manage. If the content repository you wish to use requires authentication, ensure that you set up an external application when you configure the connection to your content repository, as mentioned in Section 14.2.1, "Setting Up Connections."

14.2 Basic Configuration for the Documents Service

This section describes the steps to add the Documents service to your application.

14.2.1 Setting Up Connections

Before you can use the Documents service, you must first set up the connection to your content repository. You can reuse connections you've created or create new ones. For information on creating a connection, refer to Section 8.2, "Configuring Content Repository Connections."

Once you set up a connection to a content repository, you can set the connection as active for the Documents service by selecting **Set as primary connection for Documents service**. You can only set one active connection at a time. If you create another connection and set the new connection as active, the first connection will no longer be active. You can always return to the Properties of the connection (in the Application Resources pane, under Connections, right-click the connection name and choose **Properties**) to toggle whether the connection is active.

Note: While you can set up the connections to back-end servers at design time in Oracle JDeveloper, you can later add, delete, or modify connections in your deployed environment using Enterprise Manager Fusion Middleware Control. For more information, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

When you use Oracle Content Server as your content repository, the security settings depend on the connection configuration. If the connection uses External Application Service Public Credentials, there is no need to set up application security.

If the connection uses Identity Propagation, follow the steps described in Section 11.1.1.1, "Implementing Security for Services."

For information on installing the back-end content repository, see Section 4.3.1 "Oracle Content Server Requirements" in the *Oracle Fusion Middleware Installation Guide for Oracle WebCenter*.

14.2.2 Adding the Documents Service at Design Time

Once you have configured the Documents service with your application, you can add the task flows to your application. Oracle WebCenter Services includes the WebCenter Services Catalog, which contains the task flows you can use with your application at design time.

14.2.2.1 Documents Service Task Flows and Task Flow Parameters

The Documents service contains three task flows: Document Library, Document Library - List View, and Document Library - Recent Documents. Each task flow has its own set of parameters. You configure these values when you add the task flow to your page:

Table 14–1 describes the task flow, the name that displays in the WebCenter Services Catalog, and the content repository with which you can use the task flow. This table also describes the different capabilities a user has for each type of content repository.

Table 14–1 Documents Service Task Flows

Task Flow	Content Repository	Parameters (* = required)	Description

Document Library	■ Oracle Content Server
	■ Oracle Portal (read and search only)
	■ File system (read and write only)
		`${connectionName}`
		`${startFolderPath}`
			Provides comprehensive document management functionalities, such as copying, moving, pasting, and deleting files and folders. Section 14.2.2.1.1, "Document Lilbrary Task Flow Parameters" describes the parameters.

Table 14–1 (Cont.) Documents Service Task Flows

Task Flow	Content Repository	Parameters (* = required)	Description

Document Library - List View	■ Oracle Content Server
	■ Oracle Portal
	■ File system (read and write only)
		`${taskFlowInstId}`
		`${showFolders}`
		`${connectionName}`
		`${startFolderPath}`
		`${creator}`
		`${lastModifier}`
		`${createdBefore}`
		`${createdAfter}`
		`${lastModifiedBefore}`
		`${lastModifiedAfter}`
			Shows a simple list view of folders and documents, allows users to navigate a folder hierarchy, and enables users to customize search queries. Section 14.2.2.1.2, "Document Library - List View Task Flow Parameters" describes the parameters.

Document Library - Recent Documents	■ Oracle Content Server	`${connectionName}`
		`${lastModifier}`
		`${lastModifiedAfter}`
		`${lastModifiedBefore}`
		`${maxDocuments}`
		`${mostRecentFirst}`
			Shows the most recently used documents by the current user. Section 14.2.2.1.3, "Document Library - Recent Documents Task Flow Parameters" describes the parameters.

14.2.2.1.1 Document Lilbrary Task Flow Parameters

Table 14–2 describes the Document Library task flow parameters.

Table 14–2 Document Library Task Flow Parameters

Task Flow Parameter	Type	Description	Default Value
`connectionName`	String	Optional. The name of the connection to your content repository. If no value is entered, the default connection is used. Example: `${'MyContent'}`	The connection you selected as default in the Create Content Repository Connection dialog box. To select a default, check the **Set as primary connection for Documents service** checkbox.
`startFolderPath`	String	Optional. The root folder for the document library. Example: `${'/Manuals'}`	The root folder of the content repository you've configured with the specified connection

14.2.2.1.2 Document Library - List View Task Flow Parameters

Table 14–3 describes the Document Library - List View task flow parameters.

Table 14–3 Document Library - List View Task Flow Parameters

Task Flow Parameter	Type	Description	Default Value
`taskFlowInstId`	String	A unique identifier for the task flow that is automatically generated when you add the task flow to your application, and used internally for customization. Do not edit this value.	None
`showFolders`	Boolean	Sets whether the listing should show documents and folders (`${true}`), or documents only (`${false}`).	`${false}`
`connectionName`	String	Optional. Connection Name. The name of the connection to your content repository.	The connection you selected as default in the Create Content Repository Connection dialog box. To select a default, check the **Set as primary connection for Documents Service** checkbox.
`startFolderPath`	String	Optional. The root folder for the document library.	The root folder of the content repository you've configured with the specified connection
`creator`	String	Optional. The user who created a document. If no value is entered, then documents created by any user display. For example: ${`'monty'}`	None
`lastModifier`	String	Optional. The user who last modified a document. If no value is entered, then no filtering based on the user who last modified the document is performed. For example: `${'monty'}`	None
`createdBefore`	String	Optional. Limit the display of task flow content to files and folders created before a specified date. If no value is entered, then no "before" filtering using the creation date is applied. The value uses the ISO 8601 format: `${'yyyy-mm-ddThh:mm:ss .zzzTZ'}`[1] For example: `${'2008-11-14T06:08:02 .000+01:00'}`	None

Table 14–3 (Cont.) Document Library - List View Task Flow Parameters

Task Flow Parameter	Type	Description	Default Value
`createdAfter`	String	Optional. Limit the display of task flow content to files and folders created after a specified date. If no value is entered, then no "after" filtering using the creation date is applied. The value uses the ISO 8601 format: `${'yyyy-mm-ddThh:mm:ss.zzzTZ'}`[1] For example: `${'2008-11-14T06:08:02.000+01:00'}`	None
`lastModifiedBefore`	String	Optional. Limit the display of task flow content to files and folders modified before a specified date. If no value is entered, then no "before" filtering using the last modification date is applied. The value uses the ISO 8601 format: `${'yyyy-mm-ddThh:mm:ss.zzzTZ'}`[1] For example: `${'2008-08-07T18:24:36.000Z'}`	None
`lastModifiedAfter`	String	Optional. Limit the display of task flow content to files and folders modified after a specified date. The value uses the ISO 8601 format: `${'yyyy-mm-ddThh:mm:ss.zzzTZ'}`[1] For example: `${'2008-08-07T18:24:36.000Z'}`	None

[1] "TZ" is the time zone indicator. If the time being described is in UTC (Coordinated Universal Time), then the time zone indicator is "Z". If the time is from any other time zone, then TZ describes the offset from UTC of the time zone. For example, if the time is in California in December (Pacific Standard Time, PST), then the TZ indicator would be "-08:00".

14.2.2.1.3 Document Library - Recent Documents Task Flow Parameters Table 14–4 describes the Document Library - Recent Documents task flow parameters.

Table 14–4 Document Library - Recent Documents Task Flow Parameters

Task Flow Parameter	Type	Description	Default Value
`connectionName`	String	Optional. The name of the connection to your content repository. If no value is entered, the default connection is used.	The connection you selected as default in the Create Content Repository Connection dialog box. To select a default, check the **Set as primary connection for Documents service** checkbox.
`lastModifier`	String	Optional. The user who last modified a document. If no value is entered, then no filtering based on the user who last modified the document is performed. For example: `${'monty'}`	None
`lastModifiedAfter`	String	Optional. Limit the display of task flow content to files and folders modified after a specified date. If no value is entered, then documents modified in the last three months are shown. The value uses the ISO 8601 format: `${'yyyy-mm-ddThh:mm:ss.zzzTZ'}`[1] For example: `${'2008-08-07T18:24:36.000Z'}`	3 months prior to the current date
`lastModifiedBefore`	String	Optional. Limit the display of task flow content to files and folders modified before a specified date. If no value is entered, then no "before" filtering using the last modification date is applied. The value uses the ISO 8601 format: `${'yyyy-mm-ddThh:mm:ss.zzzTZ'}`[1] For example: `${'2008-08-07T18:24:36.000Z'}`	None
`maxDocuments`	Long	Optional. The maximum number of documents to show. For example: `${10}`. Note that there is no single quote surrounding 10.	`${null}`

Table 14–4 (Cont.) Document Library - Recent Documents Task Flow Parameters

Task Flow Parameter	Type	Description	Default Value
`mostRecentFirst`	Boolean	Optional. The sort order of documents. Use a Boolean type, for example `${true}`. Note that there is no single quote surrounding true. If no value is entered, then the most recent documents display first.	`${true}`

[1] "TZ" is the time zone indicator. If the time being described is in UTC (Coordinated Universal Time), then the time zone indicator is "Z". If the time is from any other time zone, then TZ describes the offset from UTC of the time zone. For example, if the time is in California in December (Pacific Standard Time, PST), then the TZ indicator would be "-08:00"

14.2.2.2 How to Add the Documents Service Task Flows at Design Time

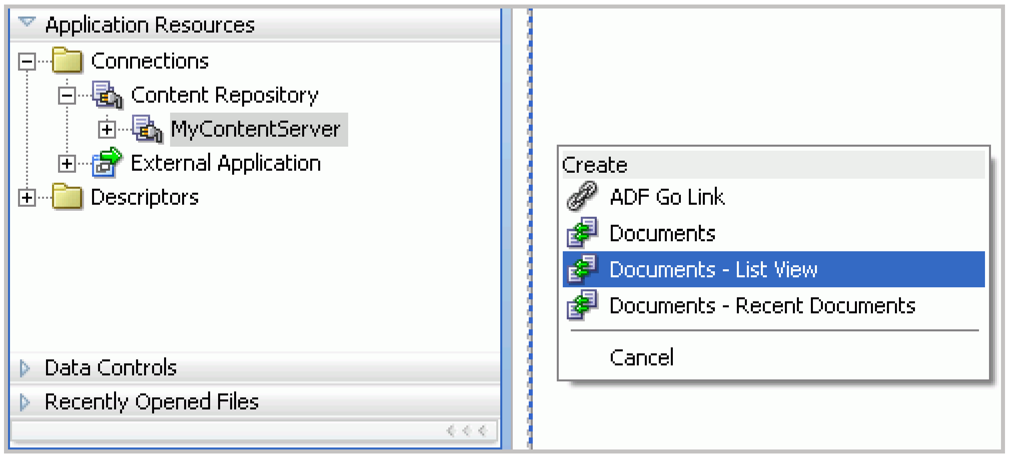
To add the Documents service task flows to your custom WebCenter application:

1. Follow the steps in Section 11.1.1, "How to Prepare Your Application to Consume Services" to implement security and create a customizable page in your application.
2. Open the customizable page.
3. Ensure a connection exists to your content repository, or set up the connection to the content repository you wish to use as your document library.

   Note: For more information on creating and setting up connections to your content repository, see Chapter 8, "Integrating Content."

4. In the Application Resource panel, navigate to **Connections**, then **Content Repository**.
5. Locate the name of your connection, then drag and drop it onto your page.

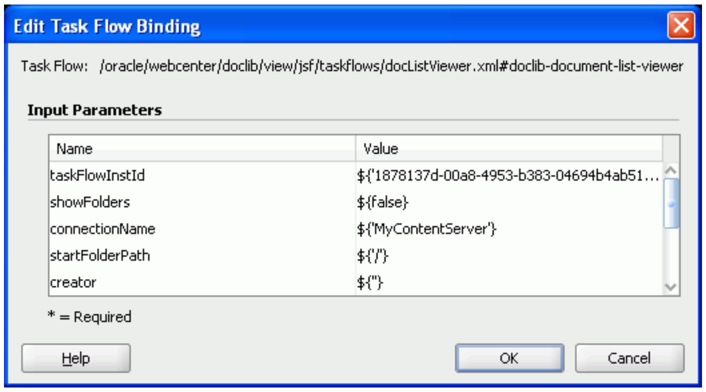
Figure 14–2 Documents Pop-Up Menu



6. In the menu that displays, choose **Documents**, **Documents - List View**, or **Documents - Recent Documents**, depending on which task flow you wish to use. If the connection you are using does not support a particular task flow, the task flow does not display in this menu. Refer to Table 14–1 for information on the content repositories and the task flows they support.

7. In the Edit Task Flow Binding dialog box, configure the parameter values. The appearance of this dialog box depends on the task flow you've chosen. For more information on these parameters, see Section 14.2.2.1, "Documents Service Task Flows and Task Flow Parameters."

Figure 14–3 Task Flow Binding Parameters for a Document Library View



Edit Task Flow Binding

Task Flow: /oracle/webcenter/doclib/view/jsf/taskflows/docListViewer.xml#doclib-document-list-viewer

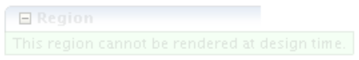
Input Parameters

Name	Value
taskFlowInstId	${'1878137d-00a8-4953-b383-04694b4ab51...
showFolders	${false}
connectionName	${'MyContentServer'}
startFolderPath	${'/'}
creator	${''}

* = Required

8. Click **OK**. A new region displays in the Design view of your page.

Figure 14–4 New Document LIbrary Task Flow Region on the Page



Region

This region cannot be rendered at design time.

Note: By default, for the Document Library view, users can upload files to the content repository at runtime. For more information, see Section 14.3.2.2, "Uploading Files to Content Repositories."

---

9. Save your page and the page definition file, then run your page to your browser. You can do this by right-clicking the page (not the page definition file) and choosing **Run**.

   The Document Library view should display on your page. You can also view the Document Library view in a sample application in the Fusion Order Demo. For more information, see Section 2.2, "Browsing the Oracle WebCenter Framework Components in the Fusion Order Demo for WebCenter at Runtime."

Note: By default, the task flow displays as full screen at runtime. You can use ADF layout components to modify this layout. To learn more about ADF layout components, refer to the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*.

14.2.3 Setting Security for the Documents Service

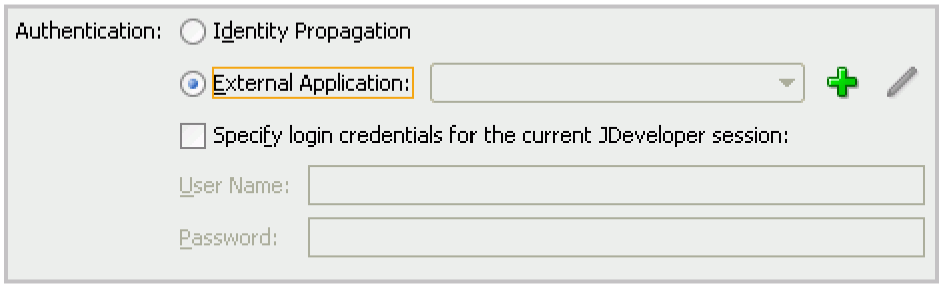
The Documents service task flows support the same security options that are supported by the content repository connections. This service can also use an external application with dedicated user accounts to access remote content repositories, such as an Oracle Content Server or Oracle Portal. For more information about using security with content repositories, see Section 8.1, "Introduction to Content Integration." For information about using external applications, see Section 24.2, "Working with External Applications."

If you are using a content repository that handles its own authentication, such as Oracle Portal or Oracle Content Server, you can associate that content repository with an external application definition to allow for credential provisioning. You can modify your connection to your content repository to use an external application without shared or public credentials. When you do this, the External Application - Change Password task flow is automatically integrated to allow your end users to provide their credentials.

To register an external application for an existing content repository connection:

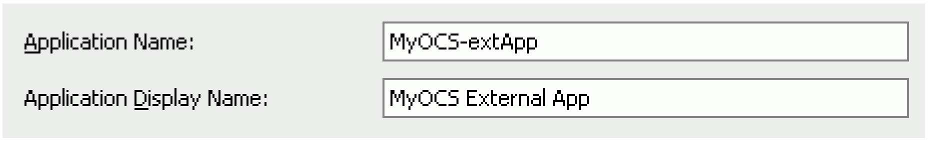
1. In the Application Resources pane, right-click your connection name and choose **Properties**.
2. In the Edit Content Repository Connection dialog box, under **Authentication**, select **External Application**.

Figure 14–5 Selecting External Application in the Content Repository Connection Dialog Box


Authentication: ( ) Identity Propagation
(●) External Application:
☐ Specify login credentials for the current JDeveloper session:
User Name:
Password:

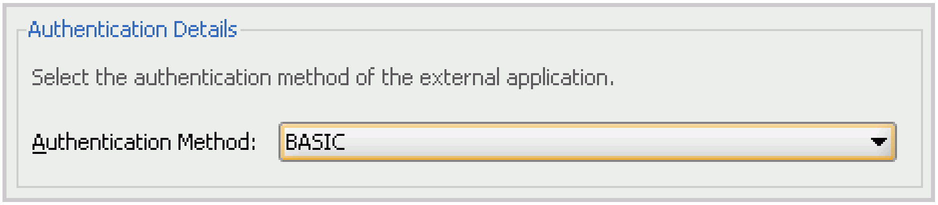
3. Click the green + (plus) sign to start the Register External Application wizard.
4. On the Name page, enter an application name and application display name, then click **Next**.

Figure 14–6 Naming the External Application


Application Name: MyOCS-extApp
Application Display Name: MyOCS External App

5. On the General page, under **Authentication Details**, choose **BASIC** from the **Authentication Method** list, then click **Next**.

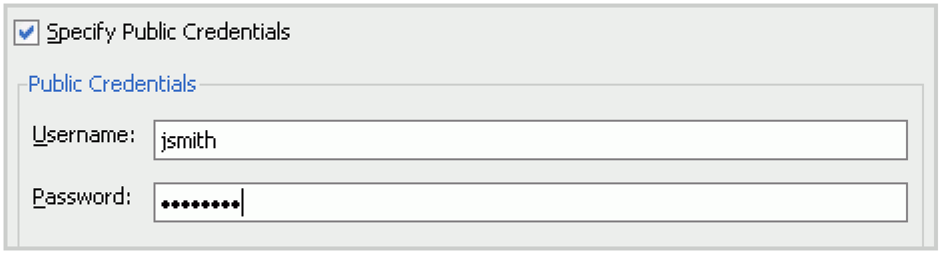
Figure 14–7 Choosing the BASIC Authentication Method


Authentication Details
Select the authentication method of the external application.
Authentication Method: BASIC

6. On the Additional Fields page, click **Next**.

7. On the Shared Credentials page, click **Next**.
8. On the Public Credentials page, select **Specify Public Credentials**, then enter the username and password for the content repository.

Figure 14–8 Specify the Public Credentials



[✓] Specify Public Credentials

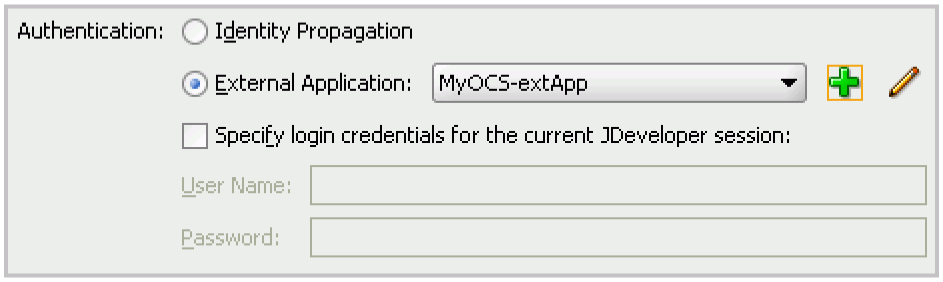
Public Credentials

Username: jsmith

Password: ••••••••

9. Click **Finish**. You'll notice that your connection now uses the external application for authentication.

Figure 14–9 Using the New External Application



Authentication: ○ Identity Propagation [✓ ● External Application:] MyOCS-extApp

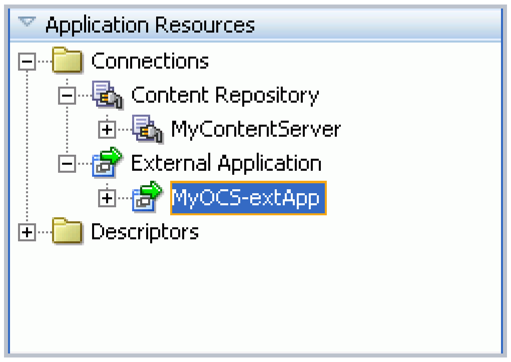
☐ Specify login credentials for the current JDeveloper session:

User Name:

Password:

10. In the Edit Content Repository Connection dialog box, click **OK**. In the Application Resources panel, you'll notice the external application now displays.

Figure 14–10 External Application in the Application Resources Panel



Application Resources
- Connections
  - Content Repository
    - MyContentServer
  - External Application
    - MyOCS-extApp
- Descriptors

If you do not apply security, and the document library requires a login to access the content, the user will not be able to authenticate, and thus will not see any content at runtime.

14.3 Advanced Information for the Documents Service

This section describes how you can use the Documents service with other WebCenter Web 2.0 Services in your application. You will also learn how to use the task flow parameters to modify your Documents service views.

14.3.1 Using the Documents Service with Other WebCenter Web 2.0 Services

You can use the Documents service with a variety of other Oracle WebCenter Web 2.0 Services. For example, you can add tags to documents in your document library, search across your application and retrieve documents in the results, and track recent changes to the document library.

Note: You can use the Tags service with the Document Library task flow only.

You can see an example of how to use the Documents service with the Tags service in Section 21.3.1, "Optional Way to Show Tags on Pages."

To learn more about how you can use these services together, refer to Section 1.1.4, "Introduction to Oracle WebCenter Web 2.0 Services."

Note: When you integrate the Documents service with the Search service, the Search service returns results from *all* content repository connections.

14.3.2 Modifying the Documents Service Task Flow Parameters

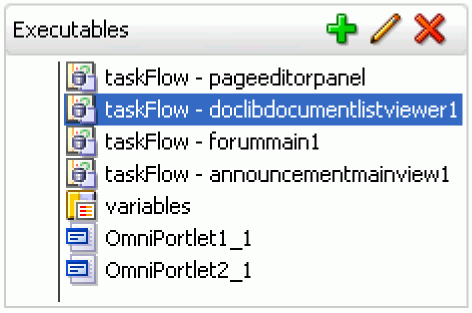
The Documents service contains optional parameters that you can set to further enhance the service. You can also modify the parameters after you have added the region to your page.

14.3.2.1 Modifying the Task Flow Binding Parameters

To modify the task flow binding information after you have created the region:

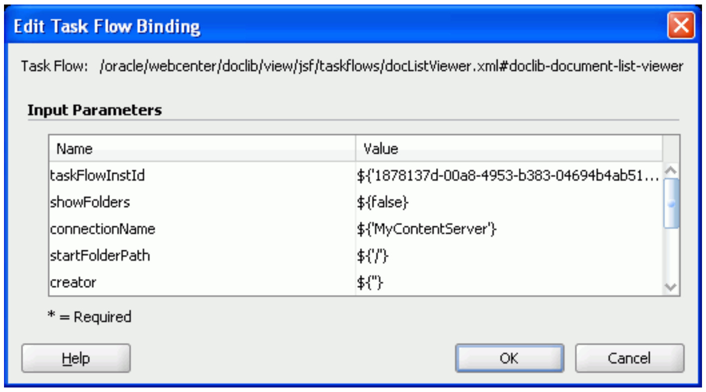
1. Go to the Bindings view of your page (click the **Bindings** tab next to the **Source** tab).
2. Under **Executables**, you'll see the task flow you added. Select this task flow. Figure 14–11 shows an example of a Document Library - List View task flow in the **Executables** section.

Figure 14–11 Document Library - List View Task Flow in the Bindings View


Executables
taskFlow - pageeditorpanel
taskFlow - doclibdocumentlistviewer1
taskFlow - forummain1
taskFlow - announcementmainview1
variables
OmniPortlet1_1
OmniPortlet2_1

3. Next to the **Executables** heading, click the pencil icon to display the Edit Task Flow Binding dialog box
4. Next to the **connectionName** parameter, enter the value for your connection. For example, `${'MyContentServer'}`.
5. Next to the **startFolderPath** parameter, enter the value for your starting path. For example, `${'/'}`.

Figure 14–12 Task Flow Binding Parameters



Task Flow: /oracle/webcenter/doclib/view/jsf/taskflows/docListViewer.xml#doclib-document-list-viewer

Input Parameters

Name	Value
taskFlowInstId	${'1878137d-00a8-4953-b383-04694b4ab51...
showFolders	${false}
connectionName	${'MyContentServer'}
startFolderPath	${'/'}
creator	${''}

* = Required

6. Click **OK** to add the parameters.

14.3.2.2 Uploading Files to Content Repositories

The Document Library enables you to allow users to upload files into content repositories. Oracle WebCenter Framework uses Apache MyFaces Trinidad to handle file upload from a browser to the application server.

To change the default settings of Apache MyFacesTrinidad, you can add three parameters to the `web.xml` file. To edit this file, open the **ViewController** project of your application. Under Web Content, open the `web.xml` file. In the Overview, navigate to **Application > Context Initialization Parameters**, then click the green plus (+) sign to add the parameters and their values (as described in Table 14–5) or simply update the code in the Source view. After you've made your changes, save the `web.xml` file, then restart Oracle JDeveloper

Table 14–5 Apache MyFaces Trinidad Parameters

Parameter	Description
`UPLOAD_MAX_MEMORY`	The maximum amount of memory in kilobytes that a single file can use when uploaded.
`UPLOAD_MAX_DISK_SPACE`	The maximum amount of disk space in kilobytes that a single file can use when uploaded.
`UPLOAD_TEMP_DIR`	The directory in which the file being uploaded is temporarily stored.

For more information, see the Apache MyFaces Trinidad documentation, located at `http://myfaces.apache.org/trinidad/devguide/fileUpload.html`.

14.3.3 Customizing Documents Service Views

You can customize the Documents service views by changing the values of the task flow parameters at design time. This section contains a few examples of how you can adjust the task flow parameters to change the display of the documents at runtime.

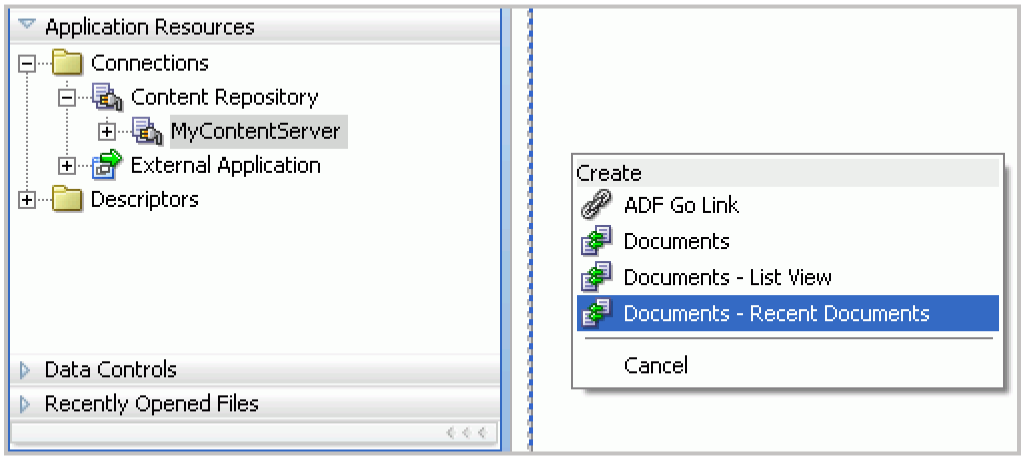
14.3.3.1 Displaying Only Recent Documents for the Currently Authenticated User

For example, you can set the task flow binding parameters for the Document Library - Recent Documents task flow to only display recent documents for the currently authenticated user.

To display only recent documents:

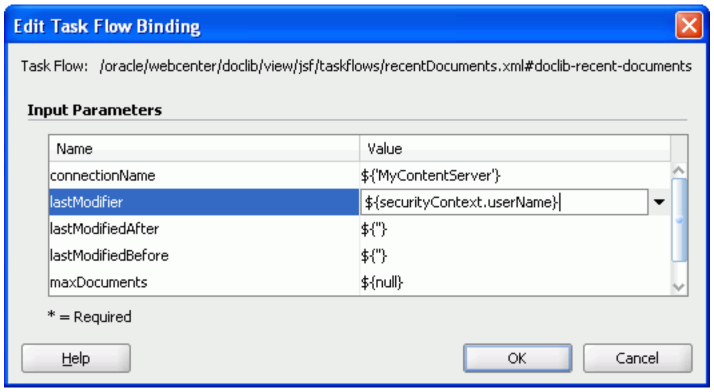
1. Ensure that you've implemented security for your application. For more information, refer to Section 11.1.1.1, "Implementing Security for Services."
2. Ensure that you have a connection to an Oracle Content Server so that you can use the Document Library - Recent Documents task flow.
3. From the Application Resources pane, drag and drop the connection name onto your page, and select **Documents - Recent Documents View** from the menu.

Figure 14–13 Dragging and Dropping the Documents - Recent Documents Task Flow



4. In the Edit Task Flow Binding dialog box, change the value for the `lastModifier` parameter to `${securityContext.userName}`, as shown in Figure 14–14.

Figure 14–14 lastModifier Task Flow Parameter



Task Flow: /oracle/webcenter/doclib/view/jsf/taskflows/recentDocuments.xml#doclib-recent-documents

Input Parameters

Name	Value
connectionName	${'MyContentServer'}
lastModifier	${securityContext.userName}
lastModifiedAfter	${''}
lastModifiedBefore	${''}
maxDocuments	${null}

* = Required

5. Click **OK**, then save your page.

14.3.3.2 Customizing the Document Library - List View with Oracle Composer

You and your end users can also customize these views at runtime for the Document Library - List view task flow. To learn more about customizing views at runtime, refer to Chapter 23, "Working with the Documents Service" in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

14.3.4 Using Adapters with the Documents Service

To learn more about using existing adapters for the content repository connections, see Chapter 8, "Integrating Content."

15 Integrating the Instant Messaging and Presence Service

This chapter explains how to integrate the Instant Messaging and Presence (IMP) service in a WebCenter application. It contains the following sections:

- Section 15.1, "Introduction to the IMP Service"
- Section 15.2, "Basic Configuration for the IMP Service"
- Section 15.3, "Advanced Information for the IMP Service"

15.1 Introduction to the IMP Service

The IMP service enables you to observe the presence status (online, offline, busy, or away) of other authenticated application users. Additionally, it provides instant access to interaction options, such as instant messages, mails, and phone calls. Further, through the IMP service users can receive notifications from configured voicemail systems.

Any WebCenter Web 2.0 service that has a user name with the same identity can integrate with the IMP service; for example, Discussions, Documents, or Mail.

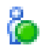
This section provides an overview of IMP features and requirements. It contains the following subsections:

- Section 15.1.1, "Understanding the IMP Service"
- Section 15.1.2, "Requirements for IMP"

15.1.1 Understanding the IMP Service

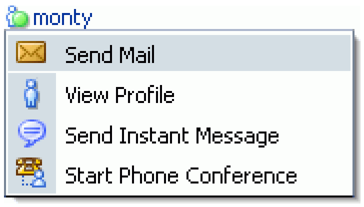
Figure 15–1 shows the Presence icon indicating a user who is online.

Figure 15–1 Presence Icon (Online)



Wherever a user is indicated, for example as the author of a document in the document library, you can click the icon to invoke a context menu (Figure 15–2).

Figure 15–2 Presence Icon Context Menu



The context menu can include the following actions:

- Send Mail (This opens the Mail Compose window.)
- View Profile (This opens the selected user's profile page, with information such as mail ID and contact numbers.)
- Send Instant Message (This opens an Oracle Presence Instant Messenger window.)
- Start Phone Conference (For presence servers that support the phone facility, this initiates a call between you and the selected user. There is no UI indication of the action, unless an exception occurs.)

You can enable your users to pull in existing instant messaging contacts (or buddies) and communicate with those contacts within the context of your application. Next to a contact name is an icon that indicates the presence state of each contact. Users can sort contacts can by name or online status. For more information, see Section 15.2.2.3, "Adding the Buddies Task Flow."

For detailed information about the IMP service at runtime, including screen shots and descriptions of the presence status options, see *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

15.1.2 Requirements for IMP

The IMP service requires a back-end presence server. Oracle WebLogic Communications Services (OWLCS) 11*g* is bundled with Oracle Fusion Middleware, but WebCenter also is certified with Microsoft Office Live Communications Server (LCS) 2005 and can integrate with other presence servers.

For information on OWLCS installation, see *Oracle Fusion Middleware Installation Guide for Oracle WebCenter*.

15.2 Basic Configuration for the IMP Service

This section describes the steps required for adding the IMP service to your application. It contains the following subsections:

- Section 15.2.1, "Setting up Connections for the IMP Service"
- Section 15.2.2, "Adding the IMP Service at Design Time"
- Section 15.2.3, "Setting Security for the IMP Service"

15.2.1 Setting up Connections for the IMP Service

After the presence server is properly installed and running, you must add a connection to it. This section describes how. It contains the following subsections:

- Section 15.2.1.1, "IMP Service Connections"
- Section 15.2.1.2, "How to Set Up OWLCS Connections for the IMP Service"

- Section 15.2.1.3, "How to Set Up LCS Connections for the IMP Service"

15.2.1.1 IMP Service Connections

The IMP service requires an **Instant Messaging and Presence** connection to the presence server (OWLCS or LCS).

When a WebCenter Web 2.0 service interacts with an application that handles its own authentication, you can associate that application with an external application definition to allow for credential provisioning. For LCS connections, an external application is mandatory. For OWLCS connections, use identity propagation (instead of an external application). This enables you to set additional security with WS-Security.

The connection appends the domain information from the connection to the user name to create the SIP address.

Note: While you can set up the connections to back-end servers at design time in Oracle JDeveloper, you can later add, delete, or modify connections in your deployed environment using Enterprise Manager Fusion Middleware Control. For more information, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

15.2.1.2 How to Set Up OWLCS Connections for the IMP Service

To set up the connection to the OWLCS presence server:

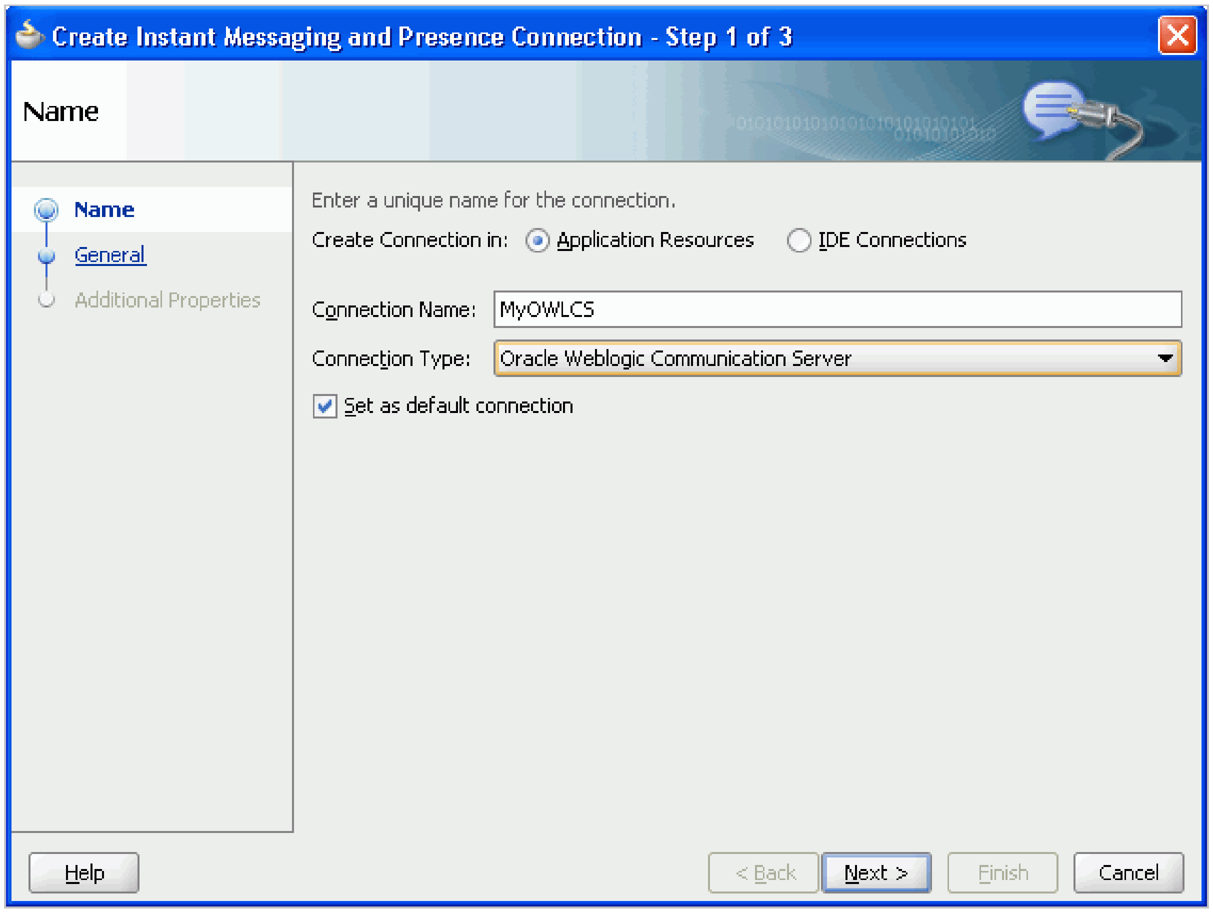
1. In Oracle JDeveloper, open the application in which you plan to consume the Instant Messaging and Presence service.
2. Ensure that security is enabled in your application according to the steps in Chapter 3.5, "Implementing Security in Your Application."
3. In the Application Navigator, under Application Resources, right-click **Connections**, then select **Instant Messaging and Presence** from the list.
4. In the Create Instant Messaging and Presence Connection dialog box, select to create the connection in **Application Resources**.

   A connection in Application Resources is available only for that application, while a connection in IDE Connections is available for all applications you create. If you plan to use the connection in other applications, then select IDE Connections to avoid having to recreate it.
5. On the **Name** page, for **Connection Name**, enter a unique name for your connection.

   No other connection should have the same name.
6. From the **Connection Type** list, select **Oracle WebLogic Communication Server**.
7. Select the check box to use this as the default connection.

   The service requires that one connection be marked as the default connection.
8. Click **Next** (Figure 15–3).

Figure 15–3 Create OWLCS Instant Messaging and Presence Connection, Step 1



Enter a unique name for the connection.

Create Connection in: [✓ Application Resources] IDE Connections

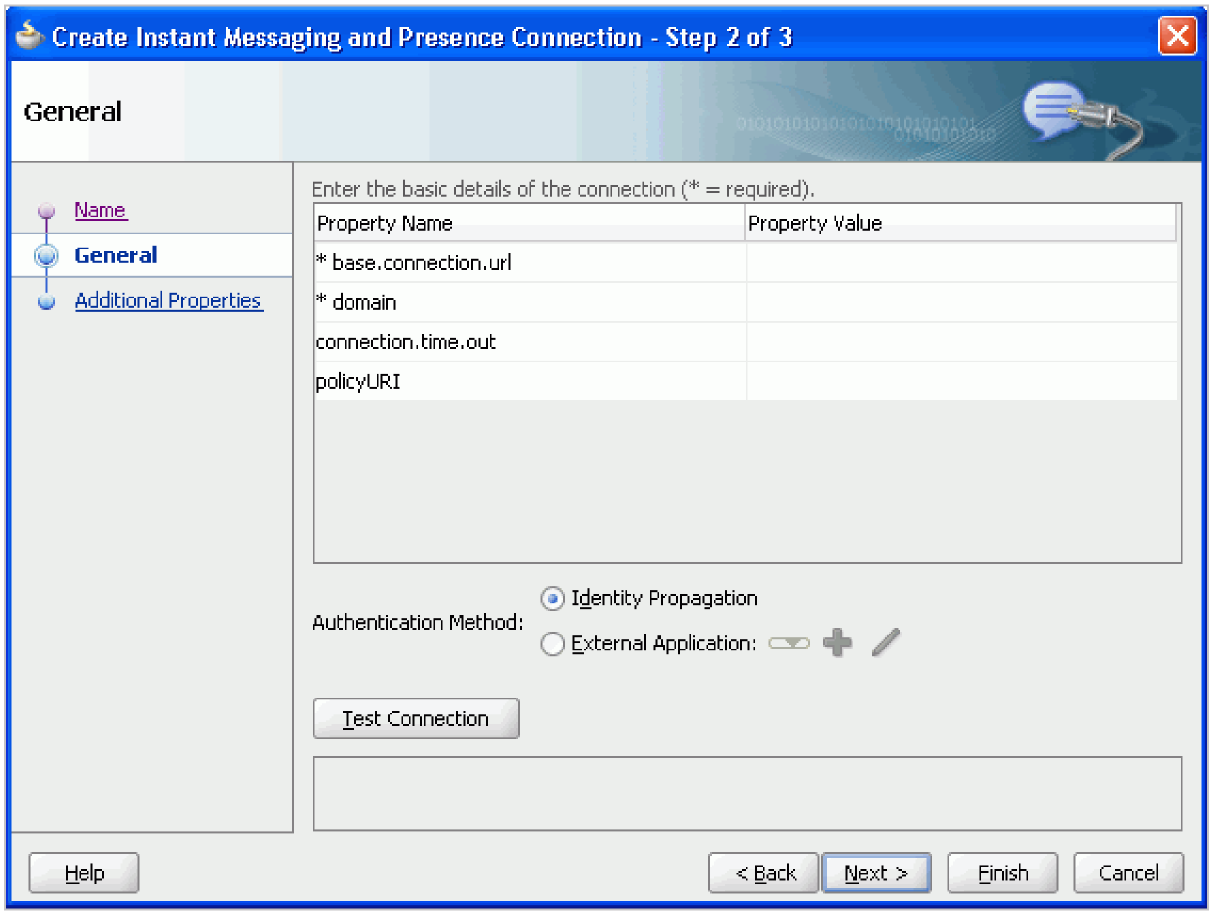
Connection Name: MyOWLCS

Connection Type: Oracle Weblogic Communication Server

[✓] Set as default connection

9. On the **General** page (Figure 15–4), for `base.connection.url` and `domain`, enter the location of your OWLCS instance.

Figure 15–4 Create OWLCS Instant Messaging and Presence Connection, Step 2



Enter the basic details of the connection (* = required).

Property Name	Property Value
* base.connection.url	
* domain	
connection.time.out	
policyURI	

Authentication Method: [✓ Identity Propagation] External Application:

For example, the `base.connection.url` could be `http://host:port`, and the `domain` could be `example.com`.

The following properties are optional:

- `connection.time.out`: The time (in seconds) the service should wait for the server to respond while making the connection. If the presence server does not respond in the given time, then it will abort the connection and report an error.
- `policyURI`: URI to the security policy that is required for authentication on the OWLCS.

For **Authentication Method**:

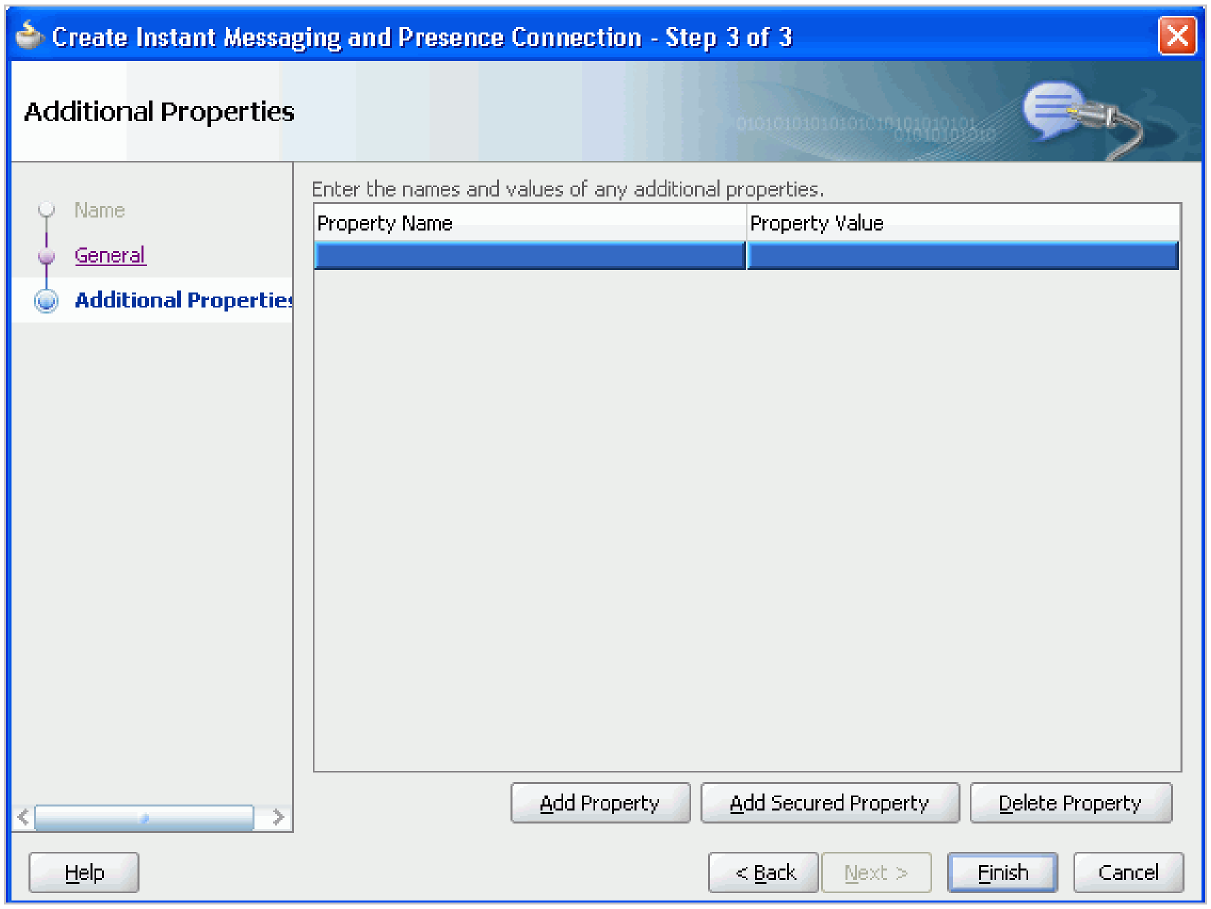
- Select **Identity Propagation** to provide secure identity propagation through the use of WS-Security.

  Note: Identity propagation is the recommended authentication method for the IMP service. For more information about identity propagation, see Section 24.11, "Identity Propagation Mechanisms."

- Select **External Application** to leverage the authentication mechanism (user names and passwords) on the presence server. The application will map the presence server user to the application user such that end users do not have to enter their user names and passwords each time they need information. For more information on configuring an external application for the IMP service, see Section 15.2.3, "Setting Security for the IMP Service."

10. Click **Test Connection** to confirm that the connection is good, then click **Next** to create the connection.
11. On the **Additional Properties** page, add any additional, optional parameters (Figure 15–5).

Figure 15–5 Create OWLCS Instant Messaging and Presence Connection, Step 3



Additional properties are necessary only for changing the default configuration of the Click2Dial call service.

Note: If values should be secured, then add them with the **Add Secured Property** button.

You can add the following parameters:

- `presence.url`: URL to the presence service. This must be supplied if the presence service is deployed on a separate server.
- `contacts.url`: URL to the contact management service. This must be supplied if the contact management service is deployed on a separate server.

- `call.url`: URL for the third-party call server. If no value is supplied, then this uses the same value as `base.connection.url`.
- `call.method`: Supports values `sip` and `pstn`:
  - When set to `sip`, the IMP service forwards the user's SIP address to the third-party call service. The third-party call service must decide on the routing of the call.
  - When set to `pstn`, the user's phone number is based on the user's profile attribute (`BUSINESS_PHONE`). You can use the connection property `call.number.attribute` to change this default profile attribute (`BUSINESS_PHONE`) to any other attribute.
- `call.domain`: The domain name of the `pstn` gateway. If no domain name is supplied, then this uses the domain value specified when the connection was created. Supply a domain name only when `call.method` is set to `pstn`.
- `contact.number.attribute`: The attribute used to read users' phone numbers from the user profile. The default is `BUSINESS_PHONE`. Supply this attribute value only when `call.method` is set to `pstn`.
- `primary.domain`: If the WebCenter user identity is qualified with a domain (for example, `john.doe@oracle.com`), and if the presence server domain is different (for example, `john.doe@example.com`) then specify the primary domain `oracle.com` here. If the user identity is qualified with a domain and the presence server uses the same `oracle.com` domain, then it is not necessary that you specify the `primary.domain`.

12. Click **Finish**.

You can see the new IM and presence connection under **Application Resources - Connections**.

15.2.1.3 How to Set Up LCS Connections for the IMP Service

To set up the connection to the LCS presence server:

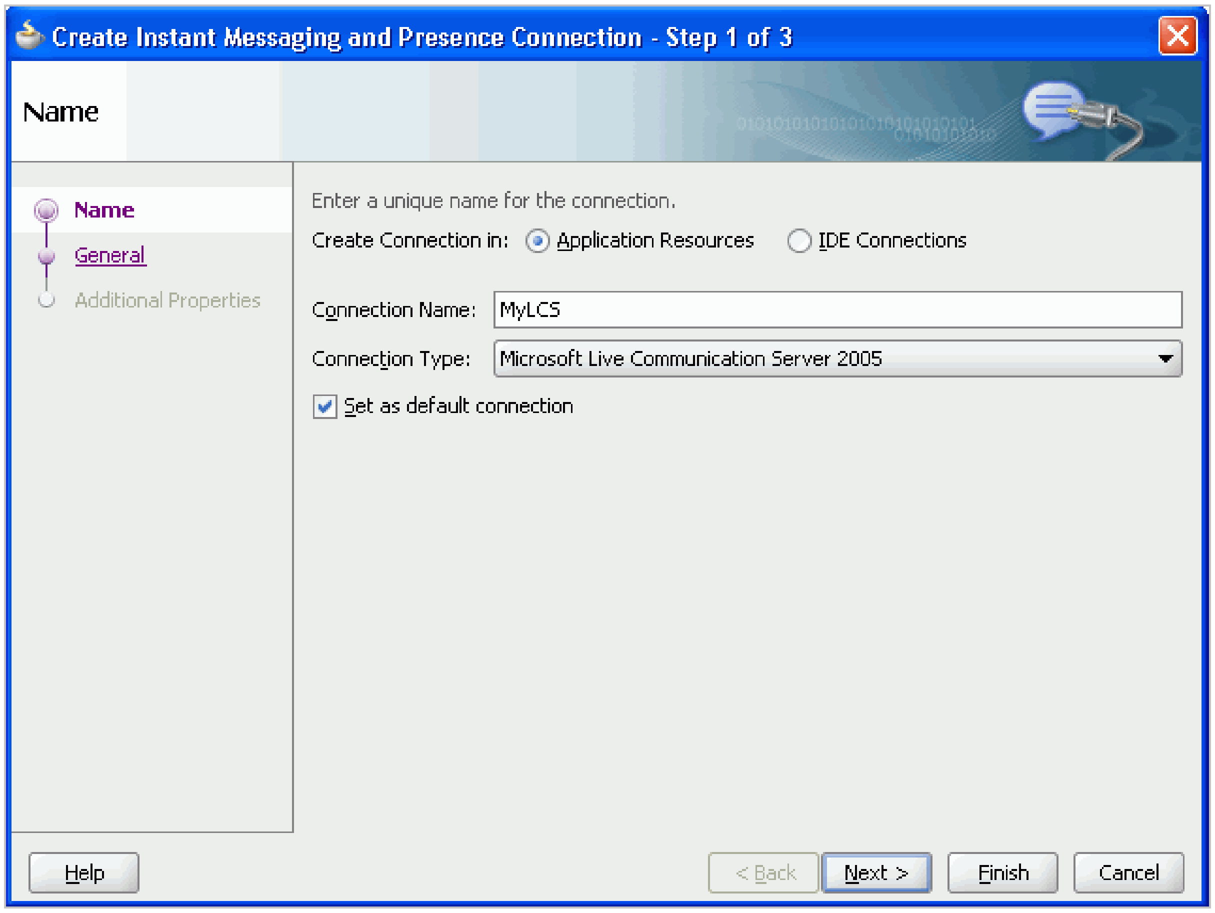
1. In Oracle JDeveloper, open the application in which to consume the Instant Messaging and Presence service.
2. Ensure that security is enabled in your application according to the steps in Chapter 3.5, "Implementing Security in Your Application."
3. In the Application Navigator, under Application Resources, right-click **Connections**, then select **Instant Messaging and Presence** from the list.
4. In the Create Instant Messaging and Presence Connection dialog box, select to create the connection in **Application Resources**.

   A connection in Application Resources is available only for that application, while a connection in IDE Connections is available for all applications you create. If you plan to use the connection in other applications, then select IDE Connections to avoid having to re-create it.
5. On the **Name** page, for **Connection Name**, enter a unique name for your connection.

   No other connection should have the same name.
6. From the **Connection Type** list, select **Microsoft Live Communication Server 2005**.

7. Select the **Set as default connection** check box to use this as the default connection, as shown in Figure 15–3.

Figure 15–6 Create LCS Instant Messaging and Presence Connection, Step 1



Enter a unique name for the connection.

Create Connection in: [✓ Application Resources] IDE Connections

Connection Name: MyLCS

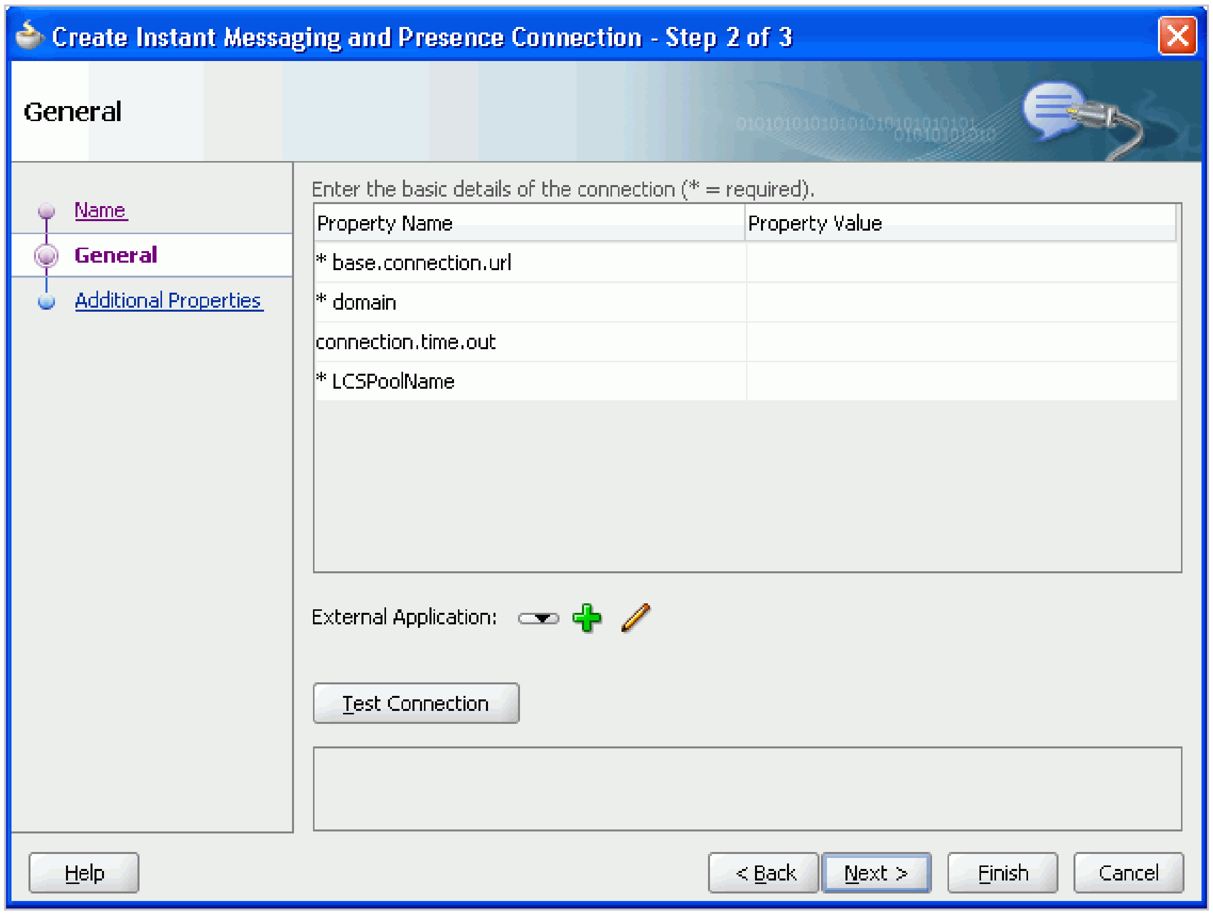
Connection Type: Microsoft Live Communication Server 2005

[✓] Set as default connection

8. Click **Next**

9. On the **General** page (Figure 15–7), for `base.connection.url` and `domain`, enter the location of your LCS instance.

Figure 15–7 Create LCS Instant Messaging and Presence Connection, Step 2


Create Instant Messaging and Presence Connection - Step 2 of 3

General

Enter the basic details of the connection (* = required).

Property Name	Property Value
* base.connection.url	
* domain	
connection.time.out	
* LCSPoolName	

External Application:

For example:

- The `base.connection.url` could be `http://host:port/RTC` where RTC is the virtual directory name under which the server side module is deployed. (See the Microsoft Live Communications Server 2005 documentation for more information.)
- The domain could be `example.com`.
- The connection.time.out property is optional. It represents the time (in seconds) the service should wait for the server to respond while making the connection. If the presence server does not respond in the given time, then it aborts the connection and reports an error.
- For `LCSPoolName`, enter the name of the pool under which Microsoft Office Communication Server components are deployed. (See the Microsoft Live Communication Server documentation for more information.)

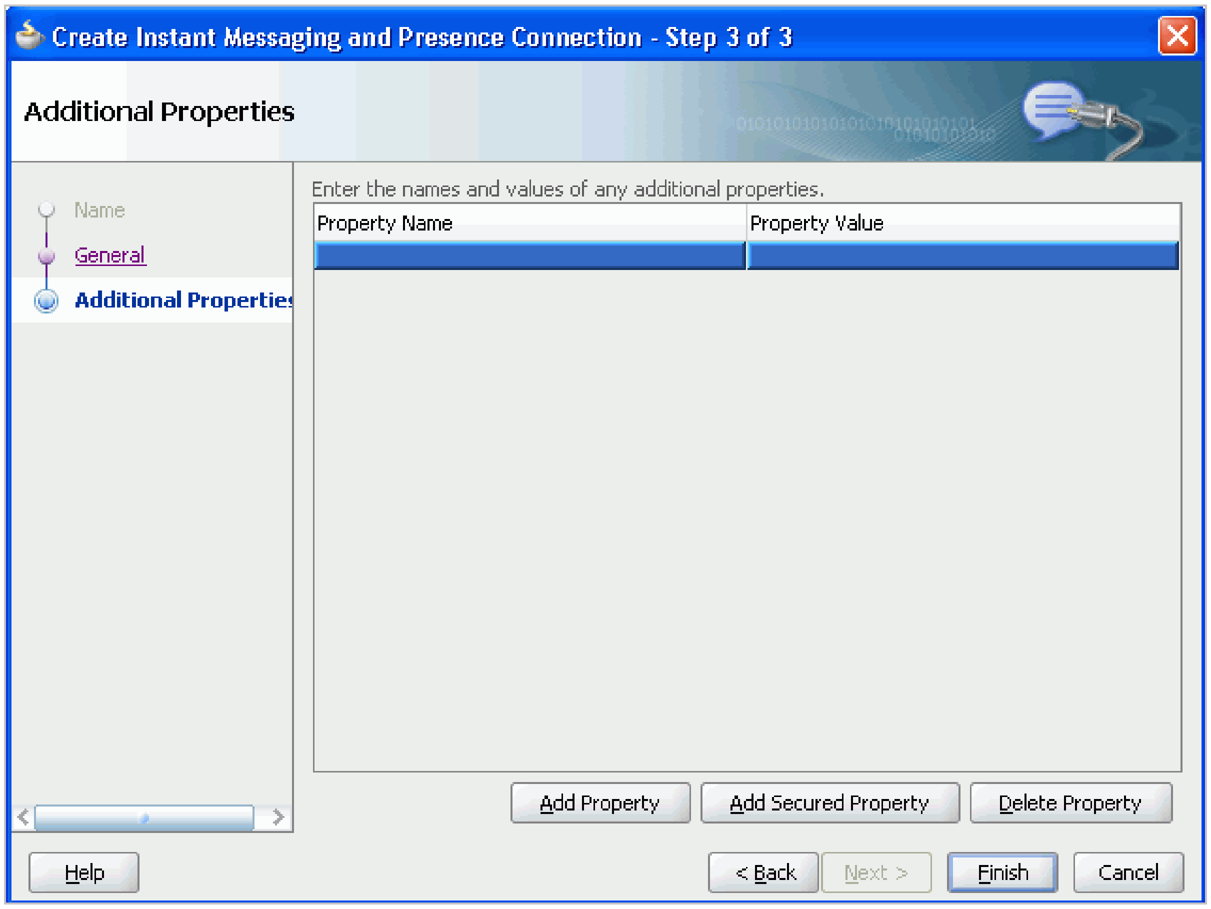
10. Select an **External Application** to leverage the authentication mechanism (user names and passwords) on the presence server.

    For LCS connections, an external application is mandatory. The application maps the presence server user to the application user such that end users do not have to enter their user names and passwords each time they need information. For detailed information about configuring an external application for the IMP service, see Section 15.2.3, "Setting Security for the IMP Service."

11. Click **Test Connection** to confirm that the connection is good.

12. Click **Next** to create the connection.
13. On the **Additional Properties** page (Figure 15–8), you can optionally add the `primary.domain` parameter.

Figure 15–8 Create LCS Instant Messaging and Presence Connection, Step 3



If the WebCenter user identity is qualified with a domain (for example, `john.deo@oracle.com`), and if the presence server domain is different (for example, `john.deo@example.com`) then specify the primary domain `oracle.com` here. If the user identity is qualified with a domain and the presence server uses the same `oracle.com` domain, then it is not necessary that you specify the `primary.domain`.

14. Click **Finish**.

You can see the new IM and presence connection under **Application Resources - Connections**.

15.2.2 Adding the IMP Service at Design Time

This section explains a basic incorporation of the IMP service into your application. It contains the following subsections:

- Section 15.2.2.1, "IMP Service Task Flows"
- Section 15.2.2.2, "How to Add the IMP Service to your Application"
- Section 15.2.2.3, "Adding the Buddies Task Flow"

15.2.2.1 IMP Service Task Flows

The IMP service includes the **Buddies** task flow.

15.2.2.2 How to Add the IMP Service to your Application

To add the IMP service to your WebCenter application:

1. Follow the steps described in Chapter 3, "Preparing Your Development Environment" to implement security and create a new customizable page in your application.
2. Open the page on which you want to add the IMP service.
3. Ensure that you have configured your application to connect to the presence server.
4. In the Component Palette, drag and drop **Presence** to the page to add the Presence icon.

   You can use the **Presence** component anywhere you want to display a user, for example, the author of a discussion topic, the sender/recipient of a mail, or the owner of a document.
5. In the resulting dialog box, enter the user name of a user that exists in the back-end server that is linked to in the IM and Presence connection.

   You can add numerous presence components to the application page.

   For information about optional **Presence** component parameters, see Section 15.3.1, "Customizing IMP Views."
6. Click **Finish**.
7. From the Component Palette, drag and drop **Presence Data** to the end of the page (that is, before the `<af:document>` tag).

   This component does not have any attributes.

   The **Presence Data** component provides the status information for all **Presence** components on the page, such as online, offline, or busy. It verifies that all presence information corresponding to the user on the whole page shows consistent status information. Without this component, all users appear offline.

   Because the **Presence Data** component makes a call to the backend server, for best performance, ensure that this is the last tag on the page. To avoid adding this tag to every page in your application, consider using a page template with **Presence Data** as the last component; that is, before the end of the `</af:form>` tag.

   Note: The Page service enables you to create new pages at runtime on which you can add components, such as forums, mail, or documents. Many components have **Presence** tags, but users do not have a handle to add the **Presence Data** tag to the page. To see presence on custom pages, you must manually add the **Presence Data** tag to the underlying template.

8. If **External Application** in IDE Connections was selected when you created the connection, then drag and drop the **External Application - Change Password** task flow into your application from the Resource Palette or Component Palette.

   This enables the end user to set the appropriate user name and password for the external application.

9. Save your project, and then run your page to a browser.

15.2.2.3 Adding the Buddies Task Flow

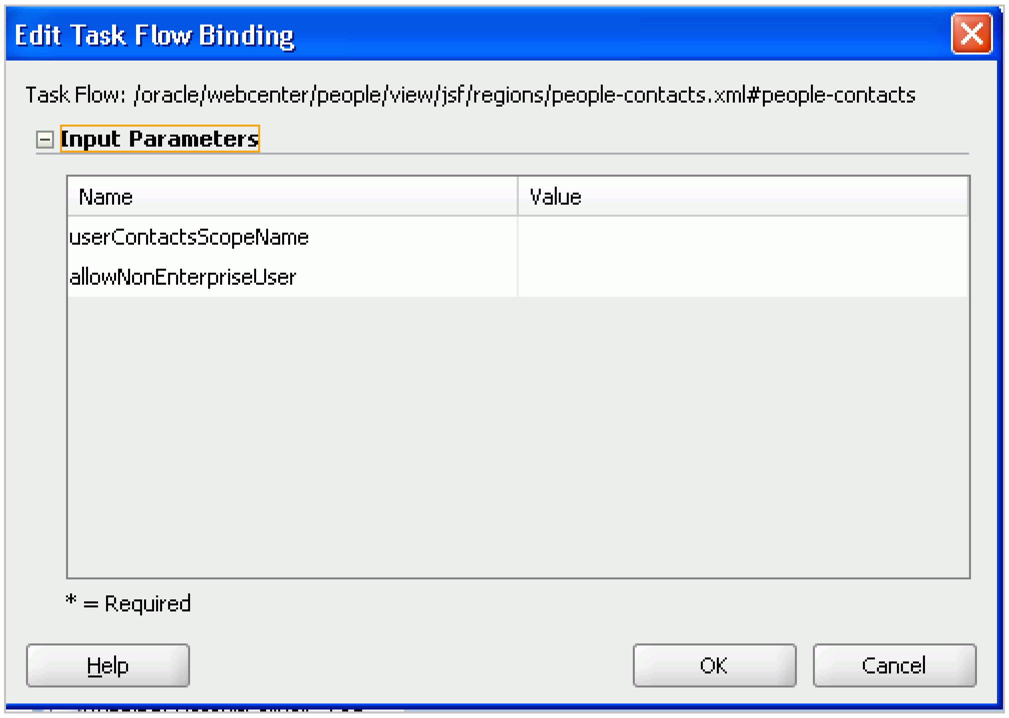
Optionally, you can add the **Buddies** task flow to an application page. For the **Buddies** task flow to work, you must have the IMP service configured in your application. Additionally, ensure that you have a **Presence Data** component tag at the end of the page to see the presence status of buddies.

Note: To add or remove buddies from your account, you must use the OWLCS/LCS client. In WebCenter applications, you can see buddies but you cannot add or remove buddies. For more information about OWLCS, see the *Oracle WebLogic Communication Services Administrator's Guide*.

With OWLCS, it is possible to see someone's online presence even if they are not your buddy. To get this functionality, you must provision the IMP service in group spaces.

Table 15–9 describes the **Buddies** task flow parameters (Figure 15–9).

Figure 15–9 Buddies Task Flow Parameters


Edit Task Flow Binding
Task Flow: /oracle/webcenter/people/view/jsf/regions/people-contacts.xml#people-contacts
Input Parameters
* = Required
Help
OK
Cancel

Name	Value
userContactsScopeName	
allowNonEnterpriseUser	

Table 15–1 Buddies Task Flows

Parameter	Description
`userContactScopeName`	The name of the group space for which to display members. Members of the current group space are listed by default. Use this parameter to display some other group space's members. In WebCenter Spaces, the group space name is available on the General tab of the group space Settings page.

Table 15–1 (Cont.) Buddies Task Flows

Parameter	Description
`allowNonEnterpriseUsers`	A Boolean value for enabling or disabling the display of users outside the scope of the enterprise. `True` means non-enterprise users are also displayed; `false` means non-enterprise users are not displayed.

15.2.3 Setting Security for the IMP Service

The IMP service requires user identity: it runs only on secured pages. You must make the WebCenter application secure using the ADF Security wizard. Additionally, you must make pages secure in their page definition files.

The user name you use to log in to the application is also used to log in to the presence server. If you do not apply ADF security, then users cannot authenticate and do not see any content at runtime.

Note: The presence server and the WebCenter application should point to the same identity store. The identity store must be LDAP-based; that is, not file-based with `jazn-data.xml`.

To access the presence server, the IMP service can use an external application with dedicated user accounts:

- OWLCS supports both identity propagation and external application-based connections. With OWLCS, user creation and deletion is manual. Any time a new user is added to (or removed from) the application's identity store, the same user must be created in (or removed from) the OWLCS user store.
- LCS supports external application connections. With a secured application, users get buddies and presence status. LCS provides an option for changing external credentials, which works as an alternative to using an external application. A logged-in user can click any Presence tag and select **Change Credentials** from the menu.

  With LCS, if security is required, then LCS should be on a private trusted network.

To use an external application for authentication:

1. On the General page of the Create Instant Messaging and Presence Connection wizard, click the **+** icon next to External Application.

   This brings up the Register External Application wizard. The application maps the presence server user to the application user such that end users do not have to enter their user names and passwords each time.

   Note: External application credential provisioning is built into the IMP connection. There is no need to drop **External Application - Change Password** task flow on a page.

2. On the Name page:

   - For **Application Name**, enter a unique name to identify the application. This name must be unique within the WebCenter application, and among other connections as well. Note that you cannot edit this field afterwards.

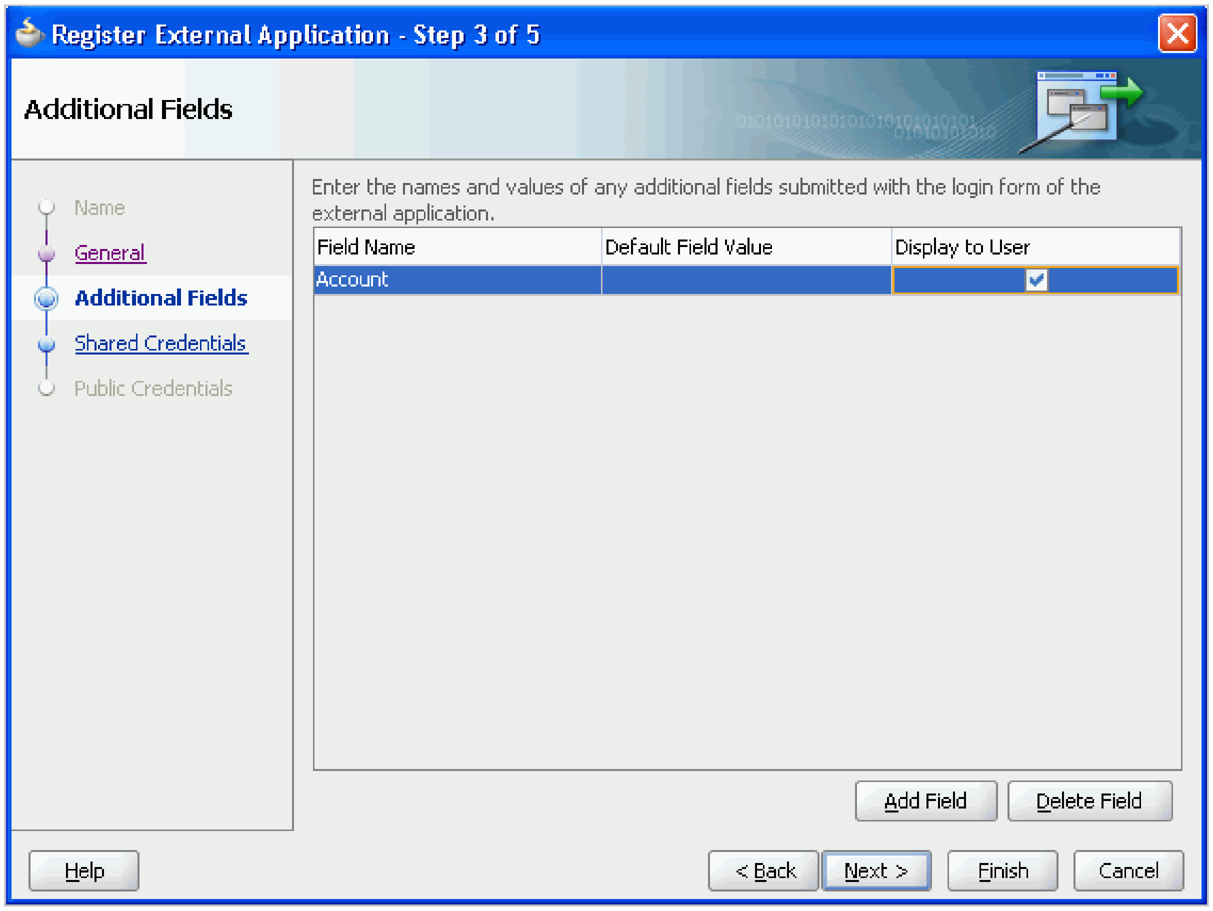
- For **Display Name**, enter a name for the application that end users will see in the credential provisioning screens.

3. Click **Next**.
4. On the General page:
   - For **Login URL**, enter the URL to which the HTML login page is submitted. View the HTML source of the application's login form to retrieve this URL.
   - For **User Name/ID Field Name**, enter the label that the application uses for the user name field, for example, User Name.
   - For **Password Field Name** field, enter the label that the application uses for the password field, for example Password.
5. From the **Authentication Method** list, select **POST**. This submits login credentials within the body of a form. The external application for the IMP service requires this authentication method.
6. Click **Next**.
7. On the Additional Fields page:

   Click **Add Field**, and add an extra field with the name "Account". Make sure to select the **Display to User** check box, as shown in Figure 15–10.

   Note: The external application for the IMP service requires this additional field. It must be displayed to users.

Figure 15–10 Account Additional Field



Enter the names and values of any additional fields submitted with the login form of the external application.

Field Name	Default Field Value	Display to User
Account		✓

8. The External Application service allows different types of credentials to be associated with a connection:

   - When shared credentials are specified, every *authenticated* user uses the same credentials to access the external application; that is, the user name and password you can define here. A single presence session is created for all logged-in users. This is not accessible to public users.
   - With public credentials, without authenticating your WebCenter application, you can view presence from a certain presence ID for all *unauthenticated* (public) users. Public credentials are used whenever an application is not secured or the user has not yet logged in. A single presence session is created for all public users.
   - With private credentials, each user must authenticate to an *individual* ID (that is, each application user must specify his own credentials). One presence session is created for each user.

9. Click **Finish** to have the external application use private credentials, or click **Next** to set up shared or public credentials.
10. *For Shared Credentials Only*: On the Shared Credentials page, ensure that **Specify Shared Credentials** is selected, then enter the shared user credentials and ID.
11. *For Public Credentials Only*: On the Public Credentials page, ensure that **Specify Public Credentials** is selected, then enter the user credentials and ID for public use.

12. Click **Finish** to register the external application.
13. In the IMP connection wizard, ensure that this newly-created external application connection for IMP is selected.

For information about using external applications, see Section 24.2, "Working with External Applications."

For information about configuring ADF security, see Section 3.5, "Implementing Security in Your Application."

15.3 Advanced Information for the IMP Service

This section describes optional features available with the IMP service. It contains the following subsections:

- Section 15.3.1, "Customizing IMP Views"
- Section 15.3.2, "Troubleshooting the IMP Service"

15.3.1 Customizing IMP Views

Table 15–2 lists the attributes supported by the **Presence** component. Only the `username` attribute is required; all other attributes are optional. You can update these attributes in the Property Inspector.

Table 15–2 Presence Component Description

Attribute	Description
`username`	The user whose presence information you want to add to the application page. This attribute is required.
`display`	How the component should display. Takes one of the following values: ■ `icon`: Display only the Presence icon; do not render the name. ■ `name`: Display the user name; do not display the Presence icon. ■ `both` (default): Display both the icon and the user name.
`displayName`	By default, the **Presence** component displays the user name. If the flag `get.display.name.from.user.profile` is set to `true` in `service-config.xml`, then the **Presence** component tries to look up a user's display name from the application's JPSContext.
`iconPosition`	The position for the icon. Possible values are `begin` and `end`. The default value is `begin`.
`controlsEnabled`	A Boolean value that defines whether the component should provide rich interactions to the end user. If this attribute is set to `false`, then the Presence component does not do anything when clicked.
`id`	A unique identifier for the component on the page. The identifier must follow a subset of the syntax allowed in HTML: ■ Must not be a zero-length String. ■ First character must be an ASCII letter (A-Z, a-z) or an underscore ('_'). ■ Subsequent characters must be an ASCII letter or digit (A-Z, a-z, 0-9), an underscore ('_'), or a dash ('-').
`smallIcon`	A Boolean value that defines whether to use the small 12x12 icon set (`true`) or to use the default 16x16 icon set (`false`). The default value is `false`.

Table 15–2 (Cont.) Presence Component Description

Attribute	Description
`rendered`	A Boolean value that defines whether or not to render this component. The default value is `true`.
`binding`	An EL reference to store the component instance on a bean. Use this to give programmatic access to a component from a backing bean or to move creation of the component to a backing bean.
`inlineStyle`	The CSS styles to use for this component. Manually enter any style in compliance with CSS version 2.0 or later, or expand this node to specify style elements. This is intended for basic style changes.
`styleClass`	A CSS style class to use for this component.

15.3.2 Troubleshooting the IMP Service

This section describes common problems and solutions for the IMP service.

Problem

The Presence icon is not visible in your custom WebCenter application.

Solution

Ensure that an IMP connection exists in your application and has been set as active.

Problem

Buddies are not visible from within your custom WebCenter application. Further, the presence status of users is not available.

Solution

This problem may arise due to various reasons. Ensure the following:

- Ensure that the IMP connections are configured properly and the base URL and domain values are correct.
- Ensure that your back-end presence server (OWLCS or Microsoft LCS) is up and running. A quick way to verify this is to ensure that you can connect to the presence server using a supported SIP client (Oracle Communicator or Microsoft Communicator).
- Ensure that you have logged-on with valid user credentials and the user exists on the presence server. For Microsoft Live Communications Server, verify that you have provided correct credentials in the external application.
- Ensure that you have buddies on the presence server.
- If all these settings are working fine, then check with your administrator to ensure that Web Services for the presence server is installed properly and is running.

In addition to verifying these settings, ensure the following if the presence status of a user is not visible:

- Ensure that the user, whose presence you cannot view, has been added as your buddy in the application. You cannot view presence of a user who is not your buddy.
- Ensure that the user is online on the presence server.
- Ensure that the application page contains the **Presence Data** component. The **Presence Data** component must be added as the last tag on the page.

Problem

Changes in the presence status of users are not visible in your custom WebCenter application.

Solution

For each logged-in user's session, the IMP service fetches the presence information from the presence server and stores it in the presence cache. For any presence or buddy requests, the service returns the data from the cache until the cache expires. The default cache expiry period is 60 seconds.

To view the updated presence status, you can wait for the cache to expire and retrieve the latest presence status.

You can also change the cache expiry time by setting the `rtc.cache.time` service configuration property to the desired value (in seconds). To do this, update `adf-config.xml` to include the highlighted entry, which shows the sample value as 30 seconds:

Example 15–1 Setting the rtc.cache.time Expiration Value in adf-config.xml

```
<adf-collaboration-config xmlns="http://xmlns.oracle.com/webcenter/collab/config">
<service-config serviceId="oracle.webcenter.collab.rtc">
<property name="rtc.cache.time" value="30"/></service-config>
</adf-collaboration-config>
```

Problem

A number of options, such as Start Phone Conference and Send Instant Message, are not available in the context menu of the IMP service in your custom WebCenter application.

Solution

Ensure that IMP service is configured in your custom WebCenter application. Also, ensure the following settings for the various context menu options:

- **Start Phone Conference** option unavailable: Ensure that you are using the OWLCS connection type. For Microsoft LCS, this option is not supported.
- **Send Mail** option unavailable: Ensure that Mail service is configured in your application.
- **View Profile** option unavailable: Ensure that your application is secured.
- **Send Instant Message** option unavailable: Ensure that the IMP service is configured in the application.

Problem

You are unable to send a message from your custom WebCenter application. Clicking the **Send Instant Message** option returns an error.

Solution

Ensure that your SIP client is supported by the presence server and you have logged on as an authenticated user. For OWLCS, the supported SIP client is Oracle Communicator. For Microsoft LCS, the supported SIP client is Microsoft Communicator.

Problem

You are not able to start a phone conference. Clicking the **Start Phone Conference** option does not initiate any action.

Solution

Ensure the following:

- Ensure that you are starting the phone conference with a user other than the logged-in user. The caller and the user being called up should be different users, with different phone numbers.
- Ensure that you have provided correct telephone numbers, in the supported format, in the user profile.
- If you specified the `contact.number.attribute` attribute in the IMP connection, then ensure that the contact numbers are indeed provided under this attribute for users. Otherwise, ensure that contact numbers are provided under the `BUSINESS_PHONE` attribute.

16 Integrating the Links Service

This chapter explains how to integrate the Links service in a WebCenter application. It contains the following sections:

- Section 16.1, "Introduction to the Links Service"
- Section 16.2, "Basic Configuration for the Links Service"

16.1 Introduction to the Links Service

This section provides overview information about the Links service features and requirements. It contains the following subsections:

- Section 16.1.1, "Understanding the Links Service"
- Section 16.1.2, "Requirements for the Links Service"

16.1.1 Understanding the Links Service

The Links service enables users to easily connect pieces of information, producing context between items. For example, suppose you are viewing discussion threads about a problem with a particular task. You know of a document that provides a detailed description of how to perform that task. You can link from the discussion thread to the document so that other users who view the thread can immediately view the linked document.

The Links service provides a means for the application developer to set up source objects (for example, the Discussions service) and target objects (for example, a document), thus enabling your users to create links between the two objects.

There are three actions associated with the Links service: create, delete, and manage. The manage action includes the create and delete actions.

The following custom JSF components are included in the service:

- **Links Detail Button**: This displays an icon and (optionally) a hyperlink that users click to open the Links Panel. To use the Links Detail Button, you must also include the Links - Dialog task flow as a region on the page.
- **Links Detail Menu Item**: This adds a menu item that opens the Links Panel. You can embed this item in an ADF menu. To use the Links Detail Menu Item, you must also include the Links - Dialog task flow as a region on the page.
- **Links Status Icon**: This has two versions:

  The gray **Links** icon (Figure 16–1) indicates that no links are present in the Links dialog.

Figure 16–1 The Links Icon (No Links Present)

The gold **Links** icon (Figure 16–2) indicates that links are present in the Links dialog.

Figure 16–2 The Links Icon (Links Present)

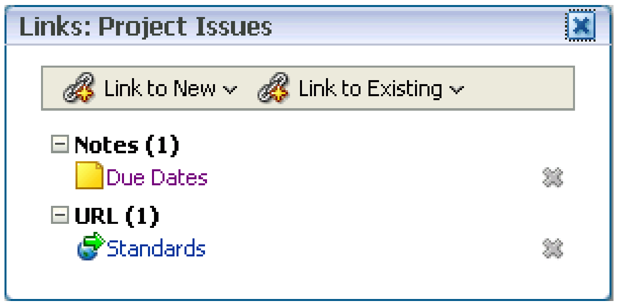
The Links service provides a way to view, access, and associate related information. For example, from an event details window you can link to the event's agenda. In a list of project assignments, you can link to the specifications relevant to each assignment.

The Links service exposes its features through a Links dialog, accessible wherever the **Links** icon displays in your application.

With the Links service, you can do the following:

- Link an object (such as a page) to an existing object (such as a discussion topic) by clicking the **Link** icon, selecting **Link to Existing**, choosing the resource **Discussions**. Optionally, you can choose a specific forum and click the topic title to choose a link (Figure 16–3).

Figure 16–3 Links Dialog



Links: Project Issues

Link to New ⌄ Link to Existing ⌄

⊟ Notes (1)
Due Dates

⊟ URL (1)
Standards

- Link an object (such as a discussion topic) to a new object (such as a new note or URL) by clicking the **Link** icon for discussion topic, selecting **Link to New**, and choosing either **Note** or **URL**.
- Create multiple links from one object.
- Delete a link.

The Links service supports bidirectional links between objects. For example, when you create a link from a discussion topic to a document, a link from the document back to the discussion topic also is created. Similarly, when you *delete* the link from the discussion topic to a document, the link from the document back to the discussion topic is automatically deleted. Bidirectional linking is not available for URLs, notes, and specific list rows.

For more information about the Links service at runtime, see *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

Note: You can see the Links service in action in the sample application, as described in Chapter 2, "Introduction to the WebCenter Sample Application."

You can link **from** the following objects:

- Announcements
- Discussions
- Documents
- Events
- Lists
- Pages
- Any object to which you bind the custom JSF components, such as the Links Detail Button

You can link **to** the following **new** objects:

- Discussions
- Documents
- Events
- Notes
- URLs

You can link **to** the following **existing** objects:

- Announcements
- Discussions
- Documents
- Events

Links are *not* available for the following WebCenter Web 2.0 services:

- Mail
- Worklist
- Search

Note: The Lists and Notes services are available in WebCenter Spaces only.

16.1.2 Requirements for the Links Service

Links automatically recognize any WebCenter service in your application. After you have configured a service in your application, you can add links. However, links work only on secured pages. Links icons do not appear on unsecured pages. For more information, see Section 16.2.3, "Setting Security for the Links Service."

16.2 Basic Configuration for the Links Service

This section describes the steps required for adding the Links service to your application. It contains the following subsections:

- Section 16.2.1, "Setting up Connections for the Links Service."
- Section 16.2.2, "Adding the Links Service at Design Time."
- Section 16.2.3, "Setting Security for the Links Service."
- Section 16.2.4, "Troubleshooting the Links Service."

16.2.1 Setting up Connections for the Links Service

The Links service requires a connection to the database where the WebCenter schema is installed. The link map (that is, relationship information such as what object is linked to what other object) is stored in the database.

> **Note:** For details about installing the database and the WebCenter schema, see *Oracle Fusion Middleware Installation Guide for Oracle WebCenter*.

To create the database connection:

1. In the Application Navigator, expand the **Application Resources** pane.
2. Right-click **Connections**, then click **New Database.**
3. Enter the following information for your database connection:
   - **Connection Name**: `WebCenter`
   - **Connection Type**: `Oracle (JDBC)`
   - **Username:** `username`
   - **Password**: `password`
   - **Host**: `<host where you will install the WebCenter schema>` (for example, `localhost`)
   - **JDBC Port:** `<port>` (for example, `1521`)
   - **SID**: `<system identifier for the database with the same JDBC port>` (for example, `ORCL`)

   You must enter the **Connection Name** exactly as "WebCenter". There are cases when you may want to leverage an existing database connection for WebCenter services, and it may not be possible to change the database connection name to "WebCenter".

   To allow WebCenter services to use another database connection by a different name, you must add the following `<data-source>` tag as a child of the `<wpsC:adf-service-config>` element in the `adf-config.xml` file. (`adf-service-config` is a child of `adf-config`, and `data-source` is a child of `adf-service-config` or sibling of `extension-registry-config`.)

   For example:

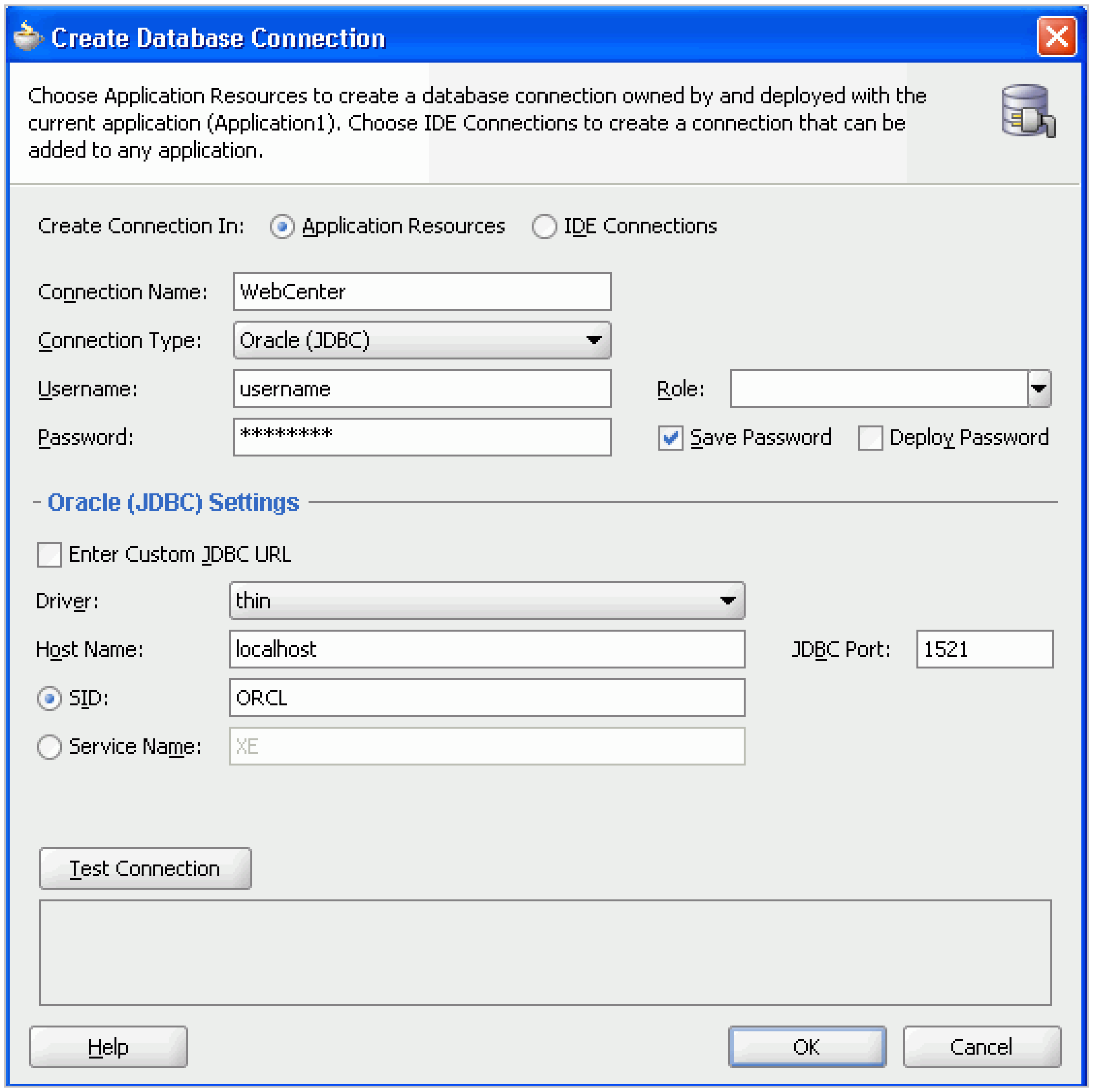
Example 16–1 Adding a data-source Tag in adf-config.xml

```
<adf-config xmlns="http://xmlns.oracle.com/adf/config"
```

```
            xmlns:mdsC="http://xmlns.oracle.com/adf/mds/config"
            xmlns:wpsC="http://xmlns.oracle.com/webcenter/framework/service"
            xmlns:jndiC="http://xmlns.oracle.com/adf/jndi/config">

    <wpsC:adf-service-config
xmlns="http://xmlns.oracle.com/webcenter/framework/service">
        <data-source jndi-name="java:/comp/env/jdbc/NewDatabaseConnDS"/>
   </wpsC:adf-service-config>
</adf-config>
```

Note: The syntax for `jndi-name` is `"java:/comp/env/jdbc/NewDatabaseConnDS"`. This is derived from the example name `NewDatabaseConn` used to create the database connection in the creation wizard.

Figure 16–4 Database Connection



Create Database Connection

Choose Application Resources to create a database connection owned by and deployed with the current application (Application1). Choose IDE Connections to create a connection that can be added to any application.

Create Connection In: (● Application Resources) ○ IDE Connections

Connection Name: WebCenter

Connection Type: Oracle (JDBC)

Username: username

Role:

Password: *********

[✓ Save Password] ☐ Deploy Password

Oracle (JDBC) Settings

☐ Enter Custom JDBC URL

Driver: thin

Host Name: localhost

JDBC Port: 1521

(● SID:) ORCL

○ Service Name: XE

4. Click **OK**.

Note: While you can set up the connections to back-end servers at design time in Oracle JDeveloper, you can later add, delete, or modify connections in your deployed environment using Enterprise Manager Fusion Middleware Control. For more information, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

16.2.2 Adding the Links Service at Design Time

This section explains a basic incorporation of the Links service. It contains the following subsections:

- Section 16.2.2.1, "Links Service Task Flows."
- Section 16.2.2.2, "How to Add the Links Service to your Application."

16.2.2.1 Links Service Task Flows

The Links service includes one task flow: **Links Dialog**.

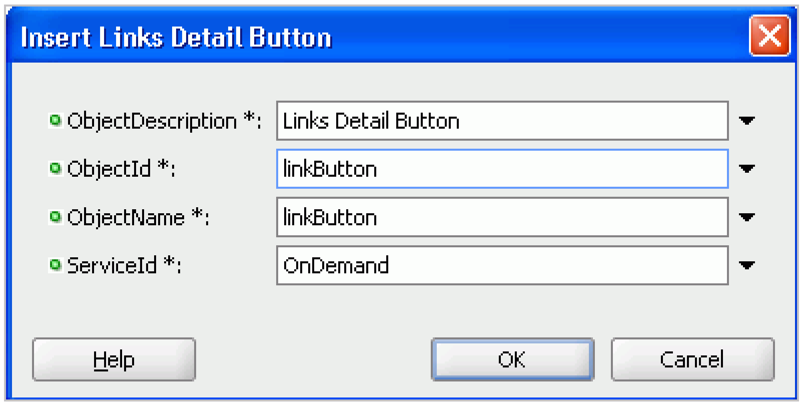
16.2.2.2 How to Add the Links Service to your Application

To add the Links Dialog task flow to your WebCenter application:

1. Follow the steps described in Chapter 3, "Preparing Your Development Environment" to implement security and create a new customizable page in your application.
2. Ensure that you have set up the database connection to a database with the WebCenter schema installed.
3. Open the page on which you want to add the Links service.
4. In the Component Palette, click **WebCenter Links Service**.
5. Drag and drop the **Links Detail Button** component onto your page inside the `panelGroupLayout`.

   The button is placed inside of a `panelGroupLayout` for the purposes of this example only. It is not required that you always place the button inside a `panelGroupLayout`.
6. In the Insert Links Detail Button dialog (Figure 16–5), enter a unique object description, ID, and name.

Figure 16–5 Insert Links Detail Button Wizard



Insert Links Detail Button

ObjectDescription *:	Links Detail Button
ObjectId *:	linkButton
ObjectName *:	linkButton
ServiceId *:	OnDemand

The properties in this dialog include:

- **objectDescription**: The description of the object to which you are binding the Links Detail Button
- **objectId**: A unique ID that identifies the object to which you are binding the Links Detail Button
- **objectName**: The name of the object to which you are binding the Links Detail Button
- **serviceId**: An application-wide ID that identifies your application

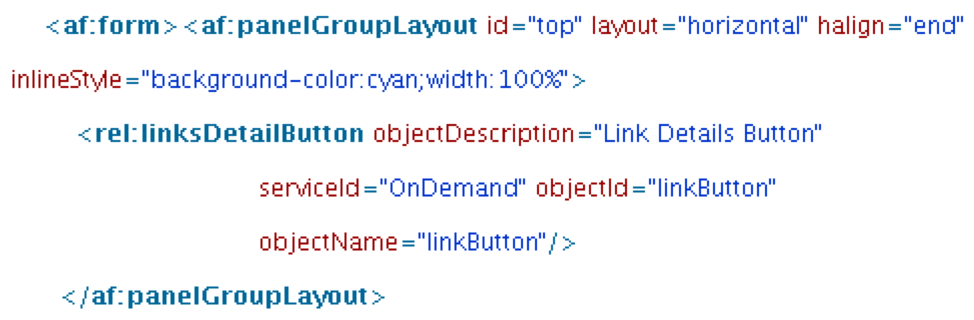
7. In the **ServiceId** field, enter `OnDemand`.

> **Note:** The Links service combines the `serviceId` and `objectId` to uniquely identify the object to which you bind the Links Detail Button.

8. Click **OK**.

   The new button displays in your page source (Figure 16–6).

Figure 16–6 The Link Detail Button in Your Page Source



```
   <af:form><af:panelGroupLayout id="top" layout="horizontal" halign="end"
inlineStyle="background-color:cyan;width:100%">
      <rel:linksDetailButton objectDescription="Link Details Button"
                         serviceId="OnDemand" objectId="linkButton"
                         objectName="linkButton"/>
    </af:panelGroupLayout>
```

9. In the Resource Palette, open **My Catalogs**, then open the **Task Flows** folder.
10. Drag and drop the **Links Dialog** task flow next to the Link Detail button on your page, and select **Region** from the context menu.
11. Save and run your page to the browser.

16.2.3 Setting Security for the Links Service

Links work only on secured pages. Links icons do not appear on unsecured pages. For information about configuring ADF security, see Section 3.5, "Implementing Security in Your Application."

Table 16–1 shows the permissions granted when you add Links components to a page. Task flow permissions, such as `RelationshipPermission` are automatically granted to authorized users when a service is consumed.

Table 16–1 Links Service Permissions

Name	Class	Action
`*` (This allows the logged-in user to create and delete links; otherwise, the user will see only links already available.)	`oracle.webcenter.relationship.model.security.RelationshipPermission`	manage
`/oracle/webcenter/relationship/view/jsf/resources/links-detail.xml#links-detail`	`oracle.adf.controller.security.TaskFlowPermission`	view
`/oracle/webcenter/relationship/view/jsf/resources/links-detail-popup.xml#links-detail-popup`	`oracle.adf.controller.security.TaskFlowPermission`	view

16.2.4 Troubleshooting the Links Service

This section describes common problems and solutions for the Links service.

Problem

The **Links** icon does not appear.

Solution

The Links service requires a database connection to the WebCenter schema, where links information is stored. Make sure that you have created the connection to the database and made it available in the Application Resources panel of the Application Navigator. If the connection is available in the Resource Palette but not in Application Resources, then simply drag the connection from the Resource Palette to the Connections folder in Application Resources.

Problem

Existing links show up, but you are not able to create new links or delete existing links.

Solution

The `RelationshipPermission` task flow permission is automatically granted to authorized users when a service is consumed. Verify that this permission has been granted. For information, see Section 11.1.3, "Automated Task Flow Grants."

17

Integrating the Mail Service

This chapter explains how to integrate the Mail service in a WebCenter application. It contains the following sections:

- Section 17.1, "Introduction to the Mail Service"
- Section 17.2, "Basic Configuration for the Mail Service"
- Section 17.3, "Advanced Information for the Mail Service"

Note: The Mail service supports only the Inbox. No other folders (or moving of messages) are supported.

17.1 Introduction to the Mail Service

This section provides an overview of Mail service features and requirements. It contains the following subsections:

- Section 17.1.1, "Understanding the Mail Service"
- Section 17.1.2, "Requirements for the Mail Service"

17.1.1 Understanding the Mail Service

The Mail service enables users to access the inbox of a mail server that supports Internet Message Access Protocol4 (IMAP4) and Simple Mail Transfer Protocol (SMTP). Additionally, it enables users to compose a new mail message from within the application (with attachments) and delete, reply to, and forward messages.

The Mail service does not replace users' mail clients; it simply enables users to access and compose mail in a single, collaborative environment.

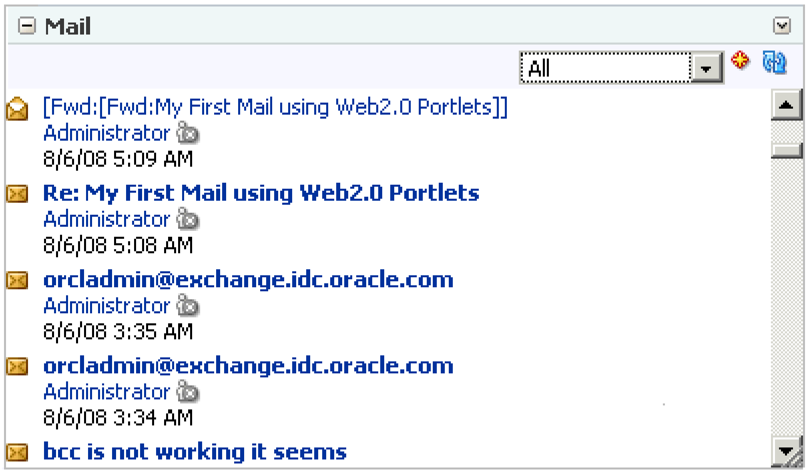
With the Mail service, you can do the following:

- Read a message by clicking the linked mail subject.
- View sender details (including date) by expanding the **From** field.
- Scroll down to see the other messages not in the view. You can navigate among the already fetched messages as cached messages.
- Attach a file to a message by expanding the attachment section within the mail dialog and clicking **Attach**. Specify an attachment in the new dialog pop-up. Remove an attachment from the mail by clicking the **Remove Attachment** icon.
- Reply to a message by clicking the **Reply** or **Reply All** icon.
- Forward a message by clicking the **Forward** icon.

- Cancel an operation (for example, sending a mail) by clicking **Cancel**.

Figure 17–1 shows the Mail task flow at runtime. At the top of the view are three elements: a list that shows the number of mails to display (here, **All**), the **Compose** icon, and the **Refresh** icon. The **Refresh** icon provides a means of manually checking for new messages in the inbox.

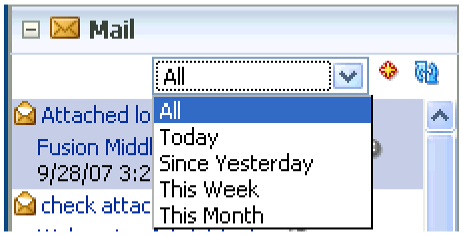
Figure 17–1 Mail Service at Runtime



[Fwd:[Fwd:My First Mail using Web2.0 Portlets]]
Administrator
8/6/08 5:09 AM
Re: My First Mail using Web2.0 Portlets
Administrator
8/6/08 5:08 AM
orcladmin@exchange.idc.oracle.com
Administrator
8/6/08 3:35 AM
orcladmin@exchange.idc.oracle.com
Administrator
8/6/08 3:34 AM
bcc is not working it seems

The list provides filters to focus the view on messages received today, messages received since yesterday, messages received this week, and messages received this month (Figure 17–2).

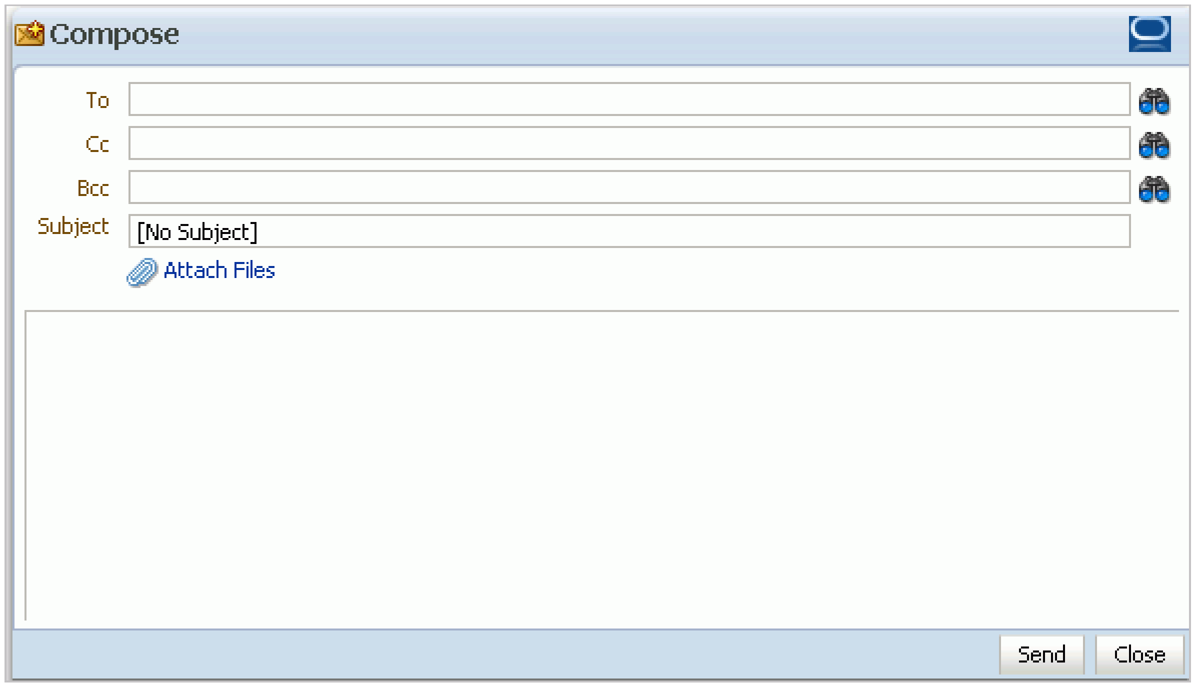
Figure 17–2 Message Filter



Note: By default, `All` displays the 50 most recent messages. For information on how to increase this amount, see Section 17.3.2, "Configuring the Number of Mails Displayed."

Click the **Compose** icon next to the list to compose a new message right from your application. Clicking this icon displays the Compose page, as shown in Figure 17–3.

Figure 17–3 Compose Mail



Use the **Search** icons to find mail addresses and contacts of users in the LDAP store that the WebCenter application uses. For any user not in the LDAP store, you must enter an explicit mail address.

For more information about the Mail service at runtime, see *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

17.1.2 Requirements for the Mail Service

The Mail service requires a mail server that supports IMAP4 and SMTP protocols.

In the WebCenter Spaces application, the Microsoft Exchange mail server is required for automatic creation of distribution lists when group spaces are created. In a custom WebCenter application, this feature may not be desirable. To disable it, do not provide the LDAP (Active Directory) server details in the mail connection.

17.2 Basic Configuration for the Mail Service

This section describes the steps required for adding the Mail service to your application. It contains the following subsections:

- Section 17.2.1, "Setting up Connections for the Mail Service"
- Section 17.2.2, "Adding the Mail Service at Design Time"
- Section 17.2.3, "Setting Security for the Mail Service"

17.2.1 Setting up Connections for the Mail Service

Before you can use the Mail service, you must first set up the connection to your mail server. The Mail service supports any mail server based on IMAP4 and SMTP protocol.

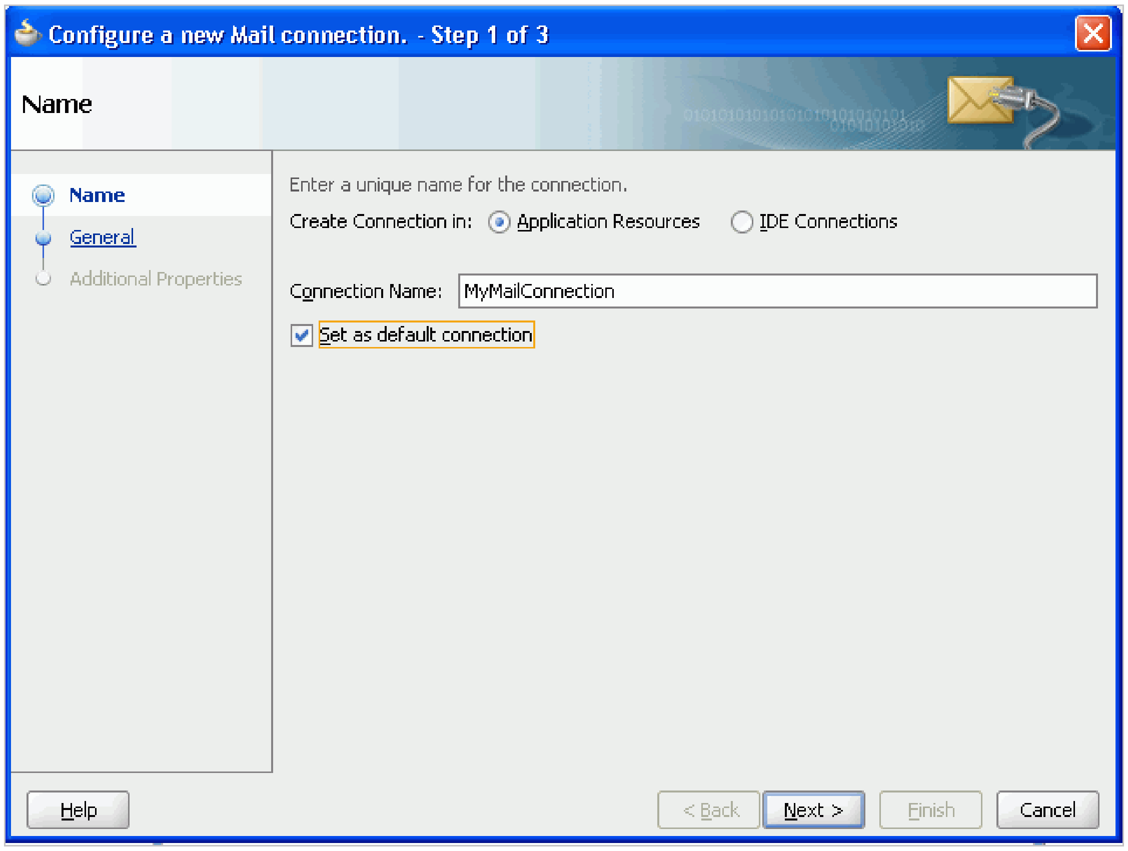
Note: While you can set up the connections to back-end servers at design time in Oracle JDeveloper, you can later add, delete, or modify connections in your deployed environment using Enterprise Manager Fusion Middleware Control. For more information, *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

To create a connection to your mail server from the application:

1. In Oracle JDeveloper, open the application in which to consume the Mail service.
2. In your application, under Application Resources, right-click **Connections**, then select **Mail** from the list.
3. Select to create the Mail service connection in **Application Resources**.

   A connection in Application Resources is available only for that application; while a connection in IDE Connections is available for all applications you create. If you plan to use the connection in other applications, then select IDE Connections to avoid having to recreate it.
4. In the **Connection Name** field, enter a unique name for the connection.
5. When configuring a single mail account, select the **Set as default connection** check box to use this as the active connection (Figure 17–4).

Figure 17–4 Configure a New Mail Connection, Step 1



Enter a unique name for the connection.

Create Connection in: [✓ Application Resources] IDE Connections

Connection Name: MyMailConnection

[✓] Set as default connection

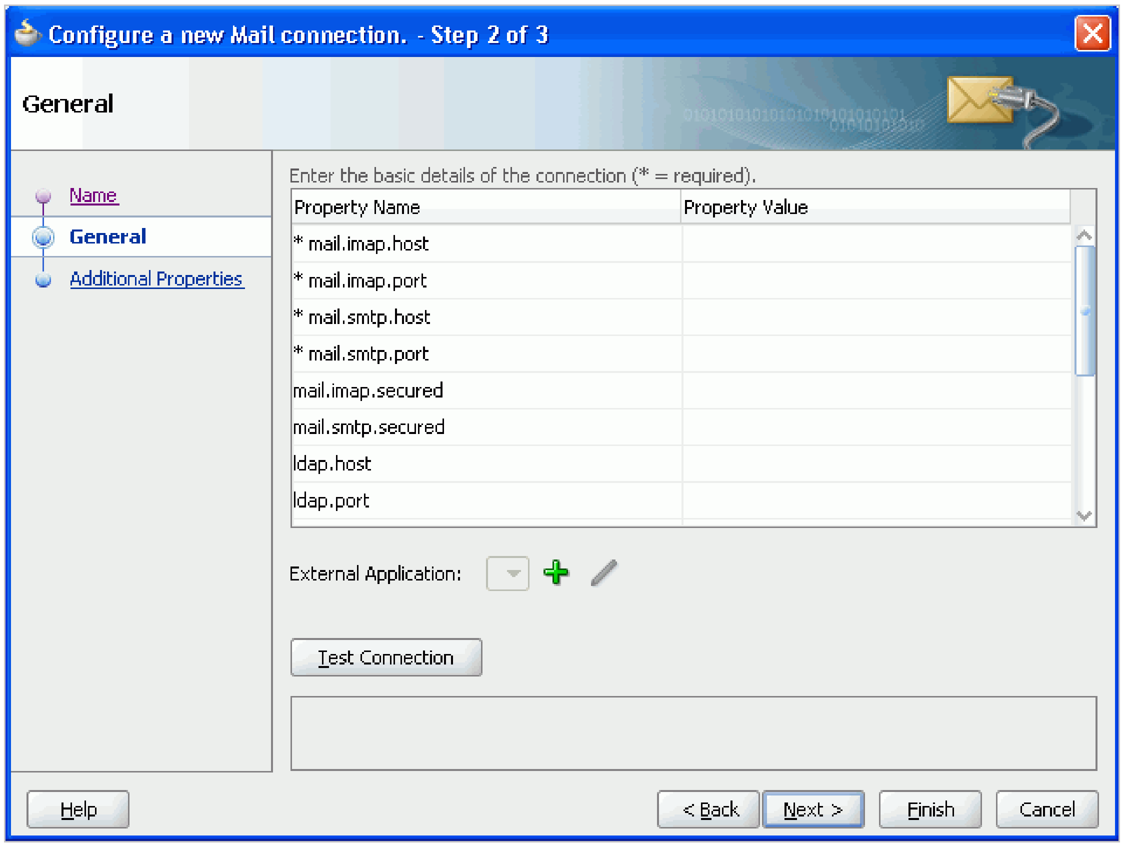
When configuring multiple mail accounts, you are not necessarily required to select this as the default connection. Keep in mind, however, that the service requires that one connection be marked as the default connection.

Note: After you create a connection as the default connection, you cannot edit it so that it is *not* the default. To use a different default connection, you must create a new connection and mark that as the default connection.

---

6. On the **General** page, enter values for the following required parameters (Figure 17–5).

Figure 17–5 Configure a New Mail Connection, Step 2



Enter the basic details of the connection (* = required).

Property Name	Property Value
* mail.imap.host	
* mail.imap.port	
* mail.smtp.host	
* mail.smtp.port	
mail.imap.secured	
mail.smtp.secured	
ldap.host	
ldap.port	

- `mail.imap.host`: The location of your IMAP server
- `mail.imap.port`: The IMAP server port number (default is -1)
- `mail.smtp.host`: The location of your SMTP server
- `mail.smtp.port`: The SMTP server port number (default is -1)

Optionally, enter values for the other parameters:

- `mail.imap.secured`: Indicates (true/false) a secure SSL connection (default is false)
- `mail.smtp.secured`: Indicates (true/false) a secure SSL connection (default is false)

- `ldap.host` and `ldap.port`: These, and all other LDAP values, are required only if Microsoft Exchange is the mail server. For the Mail service to create distribution lists reliably, the WebCenter Spaces application should use the same Active Directory server as Microsoft Exchange.

7. Click **Test Connection** to check that the host and port are available.
8. You must select an existing external application or create a new external application to proceed:

   a. For **External Application**, click the **+** icon to open the Register External Application wizard

      The application maps the mail server user to the application user so that end users are not required to enter their user names and passwords each time.

      For more information on external applications, see Section 24.2, "Working with External Applications."

      Note: External application credential provisioning is built into the mail connection. There is no need to drop **External Application - Change Password** task flow on a page.

   b. On the Name page:

      For **Application Name**, enter a unique name to identify the application. This name must be unique not only within the WebCenter application, but also among other connections. Note that you cannot edit this field afterwards.

      For **Display Name**, enter a name for the application that end users will see in the credential provisioning screens.

   c. Click **Next**.

   d. On the General page, you can *optionally* enter values if you want the external application for the Mail service to participate in Click Through Login.

      For **Login URL**, enter the URL to which the HTML login page is submitted. View the HTML source of the application's login form to retrieve this URL.

      For **User Name/ID Field Name**, enter the label that the application uses for the user name field, for example, User Name.

      For **Password Field Name** field, enter the label that the application uses for the password field, for example Password.

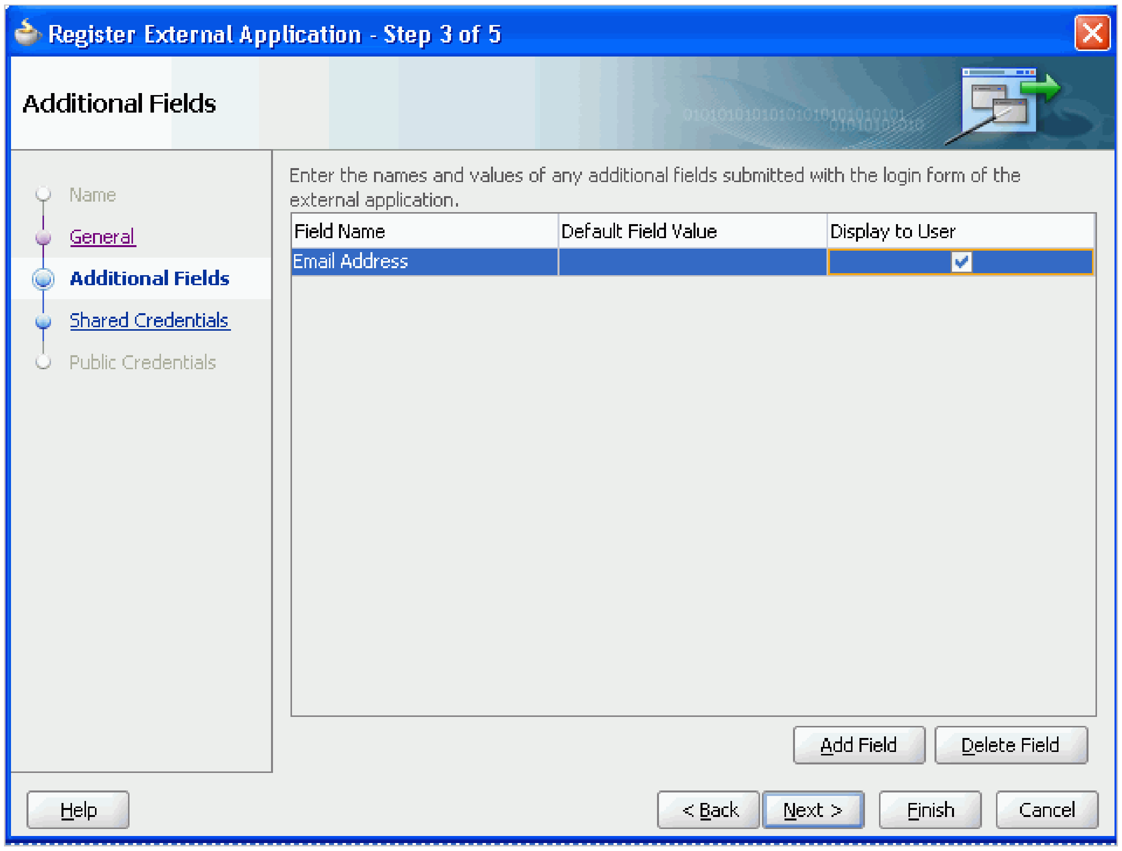
      From the **Authentication Method** list, select **POST**. This submits login credentials within the body of a form. The external application for the Mail service requires this authentication method.

   e. Click **Next**.

   f. On the Additional Fields page:

      Click **Add Field**, and add an extra field with the name `Email Address`. Make sure to select the **Display to User** check box, as shown in Figure 17–6.

Figure 17–6 Email Address Additional Field


Register External Application - Step 3 of 5

Additional Fields

Enter the names and values of any additional fields submitted with the login form of the external application.

Field Name	Default Field Value	Display to User
Email Address		✓

Note: The external application for the Mail service requires this additional field. It must be shown to users. This field will capture the user's email address, so that when the user sends mail, the sender address will be this email address.

---

g. The External Application service allows different types of credentials to be associated with a connection.

When shared credentials are specified, every *authenticated* user uses the same credentials to access the external application; that is, the user name and password you define here.

With public credentials, without authenticating your WebCenter application, you can view mail from a certain mail ID for all *unauthenticated* (public) users. Public credentials are used whenever an application is not secured or the user has not yet logged in.

Note: Public credentials are required to send mail from a self-registration page.

With private credentials, each user must authenticate to an *individual* mail ID. That is, each application user must specify his own credentials.

h. Click **Finish** to have the external application use private credentials, or click **Next** to set up shared or public credentials.

i. *For Shared Credentials Only*: On the Shared Credentials page, ensure that **Specify Shared Credentials** is selected and then enter the shared user credentials and mail ID.

j. *For Public Credentials Only*: On the Public Credentials page, ensure that **Specify Public Credentials** is selected and then enter the user credentials and mail ID for public use.

k. Click **Finish** to register the external application.

9. Back on the Mail connection wizard, ensure that this newly-created external application connection for mail is selected.

10. You can optionally add LDAP parameters and values for the Active Directory server managing (WebCenter Spaces) group space distribution lists.

    For detailed parameter information, see Table 17–1.

    Note: For WebCenter Spaces and the Mail service to share an identity management system for setting up group space mailing lists, you must use Active Directory. For information about installing and configuring mail servers as the WebCenter administrator, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

Table 17–1 LDAP Directory Server Configuration Parameters

Field	Description
LDAP Host	Enter the host name of the machine where the LDAP directory server is running.
LDAP Port	Enter the port on which the LDAP directory server listens.
LDAP Base DN	Enter the base distinguished name for the LDAP schema. For example, CN=Users,DC=oracle,DC=com.
LDAP Domain	Enter the domain to be appended to distribution list names. In WebCenter Spaces, for example, if the domain value is set to `oracle.com`, then the Finance Project group space will maintain a distribution list named `FinanceProject@oracle.com`.
LDAP Administrator User Name	Enter the user name of the LDAP directory server administrator. A valid user with privileges to make entries into the LDAP schema.
LDAP Administrator Password	Enter the password for the LDAP directory server administrator. The password will be stored in a secured store.
LDAP Default User	Enter a comma-delimited list of user names to whom you want to grant moderator capabilities. These users become members of every group space distribution list that is created. The users specified must exist in the base LDAP schema (specified in the `LDAP Base DN` field).
`ldap.secured`	Indicate whether a secured connection (SSL) is required between the WebCenter application and the LDAP directory server. If LDAP is configured to run in secure mode, then add this property (set to true/false) to use LDAP while creating distribution lists.

11. Click **Finish**.

You can see the new mail connection under **Application Resources - Connections**.

17.2.2 Adding the Mail Service at Design Time

This section explains a basic incorporation of the Mail service. It contains the following subsections:

- Section 17.2.2.1, "Mail Service Task Flows"
- Section 17.2.2.2, "How to Add the Mail Service to your Application"

17.2.2.1 Mail Service Task Flows

The Mail service includes one task flow: **Mail**. This task flow displays a mail inbox.

17.2.2.2 How to Add the Mail Service to your Application

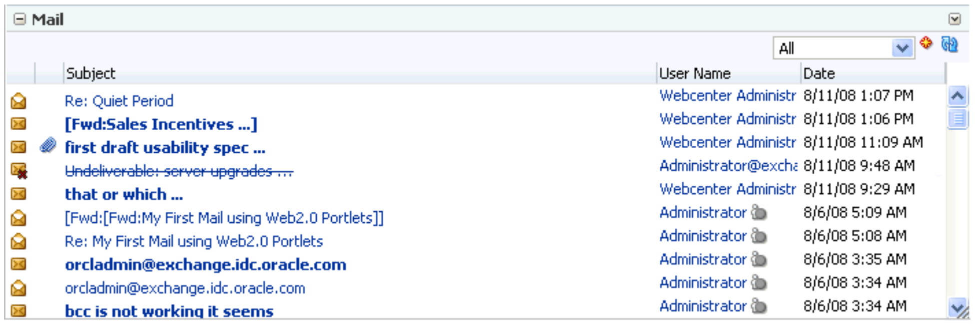
To add the Mail service to your application:

1. Follow the steps described in Chapter 3, "Preparing Your Development Environment" to create a customizable page in your application and implement security.
2. Ensure that you have configured your application to connect to the mail server.
3. If you configured your external application for private or shared credentials, then the WebCenter application must be made secure using the ADF Security wizard, and the pages must be made secure on the page definition.
4. Open the page on which to add the service.
5. In the Resource Palette, expand **My Catalogs**, **WebCenter Services Catalog**, and **Task Flows**.
6. Drag and drop the **Mail** task flow on to the page.
7. In the Create list, select **Region** to add the task flow to your page.
8. The `tabularView` parameter in the Edit Task Flow Binding dialog box is optional.

   If this parameter is set to `true`, then mail messages appear in a table, like an Inbox on any mail client. If this parameter is set to `false`, then mail messages render in a list view.

   Figure 17–7 depicts the Mail task flow where the region parameter `tabularView` is set to true.

Figure 17–7 A Mail Task Flow where the Region Parameter Tabular Is Set to True



Subject	User Name	Date
Re: Quiet Period	Webcenter Administr	8/11/08 1:07 PM
[Fwd:Sales Incentives ...]	Webcenter Administr	8/11/08 1:06 PM
first draft usability spec ...	Webcenter Administr	8/11/08 11:09 AM
~~Undeliverable: server upgrades ...~~	Administrator@exch:	8/11/08 9:48 AM
that or which ...	Webcenter Administr	8/11/08 9:29 AM
[Fwd:[Fwd:My First Mail using Web2.0 Portlets]]	Administrator	8/6/08 5:09 AM
Re: My First Mail using Web2.0 Portlets	Administrator	8/6/08 5:08 AM
orcladmin@exchange.idc.oracle.com	Administrator	8/6/08 3:35 AM
orcladmin@exchange.idc.oracle.com	Administrator	8/6/08 3:34 AM
bcc is not working it seems	Administrator	8/6/08 3:34 AM

Note: If you created a connection in IDE and not in the application, then the connection must be added to the application. For example, in the Resource Palette under IDE Connections, right-click the connection and select **Add to Application**.

9. Save your page and run it to the browser.
10. When you run the page with the Mail task flow for the first time, you will be prompted to enter external application mail credentials from within the Mail task flow.

    Enter the credentials, then click **Submit**.

Notes:

- All instances of the **Mail** task flow in an application run against the same mail server, and it serves no purpose to add more than one **Mail** task flow instance. This is true for all service task flows that require connections to back-end servers, such as task flows from the Discussions or Announcements services.
- After an application has been accessed with saved credentials, those credentials are persisted even after redeployment. To ignore saved credentials from previous deployments, edit `appUID` in `adf-config.xml` before redeploying the application. Edit the following value in bold to specify a modified value; for example, add 1 to the last value to update it as `WCApp1-1235`).

```
 <adf:adf-properties-child
xmlns="http://xmlns.oracle.com/adf/config/properties">
     <adf-property name="adfAppUID" value="WCApp1-1234"/>
 </adf:adf-properties-child>
```

17.2.3 Setting Security for the Mail Service

The Mail service works in a non-secured WebCenter application if, and only if, the external application connection is configured with public credentials. If you do not apply security, and if mail requires a login to access the content, then users will not be able to authenticate and will not see any content at runtime.

With an ADF-secured application, the Mail service fetches mail for each logged-in user. The external application credentials for the user name used to log in to the application is fetched and used to log into the mail server. The recommended approach is to have the mail server and the WebCenter application point to the same identity store.

Note: Even if the WebCenter application and the mail server share the same identity store, the Mail service does not support identity propagation. Single sign-on functionality is enabled through the external application mechanism.

For information about configuring ADF security, see Section 3.5, "Implementing Security in Your Application."

For information about using external applications, see Section 24.2, "Working with External Applications."

17.3 Advanced Information for the Mail Service

This section describes optional features available with the Mail service. It contains the following subsections:

- Section 17.3.1, "Invoking the Mail Compose Page"
- Section 17.3.2, "Configuring the Number of Mails Displayed"
- Section 17.3.3, "Troubleshooting the Mail Service"

17.3.1 Invoking the Mail Compose Page

The **Mail Compose** page enables users to determine how they want to compose individual messages within the application. Invoke the **Mail Compose** page directly with a navigation rule that directs the user to the full page:

Example 17–1 Invoking the Mail Compose Page

```
/oracle/workplace/collab/mail/view/jsf/pages/ComposeView.jspx
```

You can pass optional parameters to seed the compose message. For example, you can pass parameters to pre-populate fields like To, Cc, Bcc, From, Subject, and Content.

Set parameters only for items you want to pre-populate. If you require an empty `ComposeView.jspx`, then no `setActionListener` is necessary. You must set all parameters on `pageFlowScope`.

Example 17–2 shows parameters for the Mail Compose page (`ComposeView.jspx`):

Example 17–2 Parameters of the Mail Compose Page

```
<af:commandLink text="Compose Mail" action="sendMail">
    <af:setActionListener from="#{'john.doe@oracle.com'}"
to="#{pageFlowScope['collab.mail.compose.toList']}"/>
    <af:setActionListener from="#{'Testing Mail......'}"
to="#{pageFlowScope['collab.mail.compose.subject']}"/>
    <af:setActionListener from="#{'Testing Mail Service'}"
to="#{pageFlowScope['collab.mail.compose.content']}"/>
    <af:setActionListener from="#{'ruby@oracle.com'}"
to="#{pageFlowScope['collab.mail.compose.ccList']}" />
    <af:setActionListener from="#{'ruby@oracle.com'}"
to="#{pageFlowScope['collab.mail.compose.bccList']}" />
    <af:setActionListener from="#{'monty@oracle.com'}"
to="#{pageFlowScope['collab.mail.compose.from']}" />
    <af:setActionListener from="#{'text/html'}"
to="#{pageFlowScope['collab.mail.compose.contentType']}" />
</af:commandLink>
```

To hide the recipients fields (like **To**, **Cc** and **Bcc**), set the following action listener in `ComposeView.jspx`:

Example 17–3 Hiding Recipient Fields in the Mail Compose Page

```
    <af:setActionListener from="#{'false'}"
to="#{pageFlowScope['collab.mail.message.showrecipients']}" />sdfs
```

17.3.2 Configuring the Number of Mails Displayed

By default, the Mail service displays the 50 most recent mail messages from your Inbox folder. Providing that your mail server supports the increase in memory cache that fetching additional mail requires, the administrator can change this to a higher number in the `adf-config.xml` file.

Add the `mail.messages.fetch.size` property as shown in Example 17–4:

Example 17–4 Increasing the Number of Mails Displayed

```
<adf-collaboration-config xmlns="http://xmlns.oracle.com/webcenter/collab/config">

<service-config
serviceId="oracle.webcenter.collab.mail">
<property name="mail.messages.fetch.size" value="500"/>
</service-config>

</adf-collaboration-config>
```

Alternatively, your Fusion Middleware administrator can increase this value with the WLST command `setMailServiceProperty`. For more information, see the *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*.

This change applies to all users; that is, following Example 17–4, all users see 500 recent mails in their Inbox in the WebCenter application. Increasing the number of messages shown correspondingly increases the cache size on the WebCenter application. Take care to set this to a reasonable size.

17.3.3 Troubleshooting the Mail Service

This section describes common problems and solutions for the Mail service.

Problem

Users are not able to retrieve their mail messages or send mail messages from within the custom WebCenter application.

Solution

Ensure the following:

- Ensure that a Mail service connection exists in your application.
- Ensure the required Mail service connection is marked as the default connection.
- Ensure the mail server configured in the connection is up and running. You can try connecting from any other e-mail client to ensure that the connection details are correct.

Problem

The Mail service within your custom WebCenter application requires users to log in, but users are not able to authenticate and do not see any content at runtime. When users access the Mail service, it throws the ExtApp Authorization Exception.

Solution

The Mail service works in a non-secured custom WebCenter application if, and only if, the Mail connection is configured to use an external application connection with public credentials. If your application is running in non-secured mode, then ensure that you have configured public credentials for your external application.

If you want your custom WebCenter application to run in secure mode, then you must configure ADF security for your application.

18 Integrating the Recent Activities Service

This chapter describes how to integrate the Recent Activities service into your custom WebCenter application. This service enables you to display the most recent changes to the services in your application.

This chapter includes the following sections:

- Section 18.1, "Introduction to the Recent Activities Service"
- Section 18.2, "Basic Configuration for the Recent Activities Service"
- Section 18.3, "Advanced Information for the Recent Activities Service"

18.1 Introduction to the Recent Activities Service

Oracle WebCenter Services are preconfigured to work with the Recent Activities service. Recent Activities enable users to view the most recent changes to the services in your application. For example, once you add the Documents service, it automatically produces the information that the Recent Activities service will use to display the most recent changes to the document library. This service is useful to your users who want a quick and easy way to view any additions or changes to a particular area of the application.

The Recent Activities service can track information from the following WebCenter Services:

- Documents
- Announcements
- Discussion Forums
- Pages created or modified by the Page service

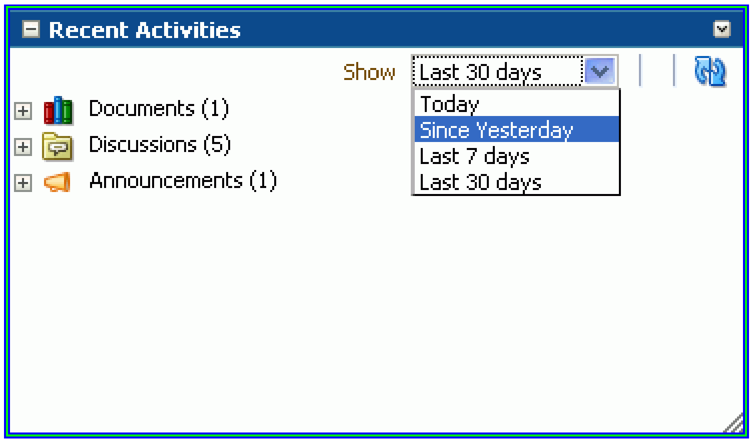
For more information about these services, see Chapter 12, "Integrating the Announcements Service," Chapter 13, "Integrating the Discussions Service,", Chapter 14, "Integrating the Documents Service," and Chapter 7, "Enabling Runtime Creation and Management of Pages."

See Also: *Oracle Fusion Middleware User's Guide for Oracle WebCenter* for more information about the service at runtime.

18.1.1 Understanding the Recent Activities Service

The Recent Activity service provides a summary view of recent changes to a variety of services. You can specify the range of time to consider recent by selecting a time range from a list at the top of the task flow. Recorded activities include additions or revisions of pages, documents, discussion forums, and the like (Figure 18–1).

Figure 18–1 Recent Activities Service at Runtime



By default, the Recent Activities service works with the Documents, Announcements, Discussion Forums, and Page services. You can narrow the services that the Recent Activities service tracks. For more information about this setting, see the next section, Section 18.3, "Advanced Information for the Recent Activities Service." You can also learn more about using Oracle WebCenter Web 2.0 Services in Section 1.1.4, "Introduction to Oracle WebCenter Web 2.0 Services."

Note: You can see the Recent Activities service in action in the sample application, as described in Chapter 2, "Introduction to the WebCenter Sample Application."

18.2 Basic Configuration for the Recent Activities Service

This section describes how to add the Recent Activities service to your custom WebCenter application.

18.2.1 Setting up Connections for the Recent Activities Service

You do not need to set up specific connections for the Recent Activities service. You only need to ensure that the service(s) you are tracking with the Recent Activities service is configured. For more information about setting up connections for other services, see the individual services chapters in this guide.

18.2.2 Adding the Recent Activities Service at Design Time

To use the Recent Activities service, you must add its task flow to your application. Ensure that you have at least one of the services it tracks in your application. There is a single Recent Activities task flow available in the WebCenter Services Catalog, which you can add to your application to display information about services in your application to your users.

The Recent Activities service automatically picks up other services in your application and looks for recent activity within these services. Before running your application containing the Recent Activities service, however, you may want to ensure your application contains at least one of the services you wish to monitor.

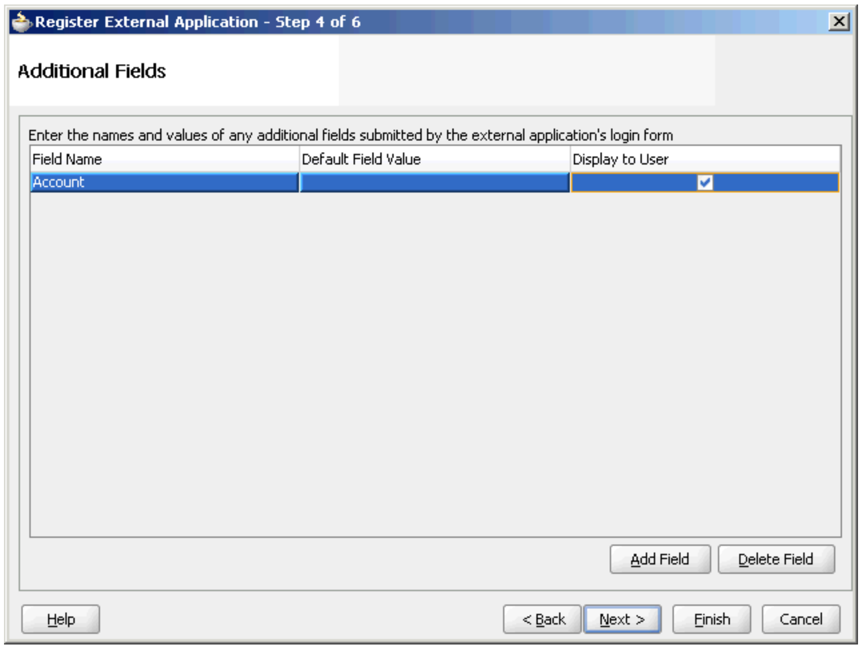
To add the Recent Activities task flow to your application:

1. Follow the steps in Section 11.1.1, "How to Prepare Your Application to Consume Services" to implement security and create a customizable page in your custom WebCenter application.
2. Open the source of the customizable page.
3. Ensure that your application includes at least one of the services that the Recent Activities service tracks.

> **Note:** If you run an application that does not contain one of these services, the Recent Activities service displays an error saying `Warning: Unable to obtain recent activity data`.

4. Create a new external application by right-clicking the **ViewController** project in your application, then choosing New.
5. In the New Gallery, under General, select **External Applications**, then select **External Application** from the Items list.
6. Click **OK**.
7. In the Application Name field, enter a name for your application, such as `MyExternalApplication`, then click **Next**.
8. On the General page, leave the **Login URL** field empty.
9. In the **User Name/ID Field Name** field, enter a user name. The user name should match the login credentials of the servers you are tracking. For example, if you are tracking the Documents service with Oracle Content Server, the login credentials must match those of the user on the Oracle Content Server.
10. In the **Password Field Name** field, enter a password.
11. Under Authentication Details, choose **POST**, then click **Next**.
12. On the Additional Fields page, click the **Add Field** button.
13. For the Field Name, enter `Account`.
14. Select the **Display to User** checkbox.

Figure 18–2 Additional Fields Page

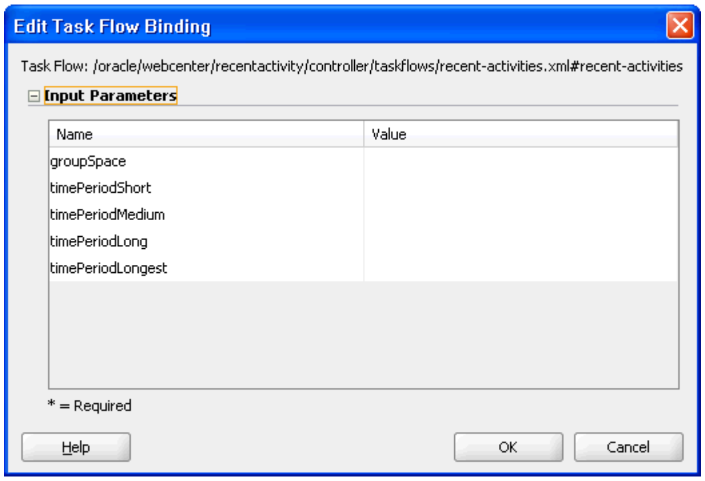


Enter the names and values of any additional fields submitted by the external application's login form

Field Name	Default Field Value	Display to User
Account		✓

15. Click **Finish**.
16. In the Resource Palette, open the **WebCenter Services Catalog**, then open the **Task Flows** folder.
17. Click **Recent Activities** and drag and drop it onto your page after the `<af:form>` tag, and choose **Region**.

Figure 18–3 Edit Task Flow Binding Dialog Box



Task Flow: /oracle/webcenter/recentactivity/controller/taskflows/recent-activities.xml#recent-activities

Name	Value
groupSpace	
timePeriodShort	
timePeriodMedium	
timePeriodLong	
timePeriodLongest	

* = Required

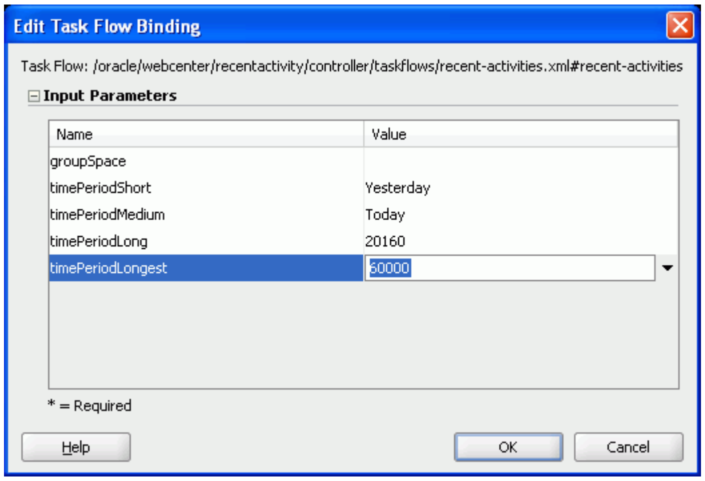
In the Edit Task Flow Binding dialog, you can set the values for the time range that displays at runtime. This time range controls what activities display, as you will see in the next section. If you leave these parameters empty, the Recent Activities service will use the default values of the service.

Table 18–1 describes the possible values for these parameters.

Table 18–1 Recent Activities Time Range Parameter Values

Parameter	Values
`groupSpace`	Leave the value of this parameter as `null`.
`timePeriodShort`	Default value: `Today` Possible values: `Today`, `Yesterday`, or a `Time in Minutes`
`timePeriodMedium`	Default value: `Since Yesterday` Possible values: `Today`, `Yesterday`, or a `Time in Minutes`
`timePeriodLong`	Default value: `10080` (10080 minutes is equivalent to 7 days.) Possible values: `Today`, `Yesterday`, or a `Time in Minutes`
`timePeriodLongest`	Default value: `43200` (43200 minutes is equivalent to 30 days.) Possible values: `Today`, `Yesterday`, or a `Time in Minutes`

Figure 18–4 Example of the Edit Task Flow Binding Dialog for the Recent Activities Service



Edit Task Flow Binding

Task Flow: /oracle/webcenter/recentactivity/controller/taskflows/recent-activities.xml#recent-activities

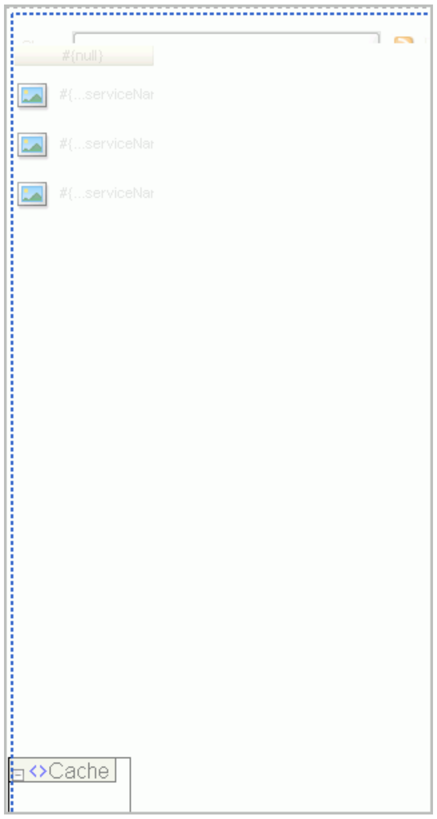
Input Parameters

Name	Value
groupSpace	
timePeriodShort	Yesterday
timePeriodMedium	Today
timePeriodLong	20160
timePeriodLongest	60000

* = Required

18. Click **OK**. The binding displays on your page, as shown in Figure 18–5.

Figure 18–5 The Recent Activities Task Flow on a Page in the Design View



19. Save your project and run your page to the browser.

> **Note:** When you add the Recent Activities task flow to your page, you will see a task flow parameter called **groupSpace**. The default value of this parameter is `null`, which tells the task flow to search the entire application for recent activity.

18.2.3 Setting Security for the Recent Activities Service

The Recent Activities service uses the security applied to the underlying services being analyzed. So, the information returned by the services only contains content for which the current user has at least View privileges. For example, a user who does not have View privileges to view content in a document library will not see recently added documents in the Recent Activities view. Similarly, a public user will only see activities returned on services exposed to the public, and do not require authentication to view.

18.3 Advanced Information for the Recent Activities Service

In addition to the task flow parameters, you can further refine the behavior of the Recent Activities service by editing the settings in the `adf-config.xml` file.

You can change a variety of the Recent Activities service parameters after you have added it to your application. When you add the task flow to your page, you also automatically add the following entry to the `adf-config.xml` file, located in the Application Resources panel, under Descriptors, in the **ADF META-INF** folder:

```
  <recentactivityC:adf-recent-activity-config
 xmlns="http://xmlns.oracle.com/webcenter/recentactivity/config">
     <display-properties numServiceRows="25" shortPeriod="TODAY"
                       mediumPeriod="YESTERDAY" longPeriod="10080"
                       longestPeriod="43200"/>
    <services-filter>
      <exclude/>
    </services-filter>
  </recentactivityC:adf-recent-activity-config>
```

You can change the following settings in this entry:

Table 18–2 Recent Activities Settings in the adf-config.xml File

Setting	Description
`numServiceRows`	Default value: `25` The maximum number of rows displayed for a single service. If there are more recent activity results than this, the most recent ones up to this number are returned. For example, if there are 32 recently modified documents and this setting is 25 then only the most recent 25 will show up in the results.
`shortPeriod`	Default value: `Today` See Table 18–1. This setting is the same as the `timePeriodShort` task flow parameter, and is used when the task flow parameter is not specified.
`mediumPeriod`	Default value: `Yesterday` See Table 18–1. This setting is the same as the `timePeriodMedium` task flow parameter and is used when the task flow parameter is not specified.
`longPeriod`	Default value: `10080` (Last 7 Days) See Table 18–1. This setting is the same as the `timePeriodLong` task flow parameter and is used when the task flow parameter is not specified.
`longestPeriod`	Default value: `43200` (Last 30 Days) See Table 18–1. This setting is the same as the `timePeriodLongest` task flow parameter and is used when the task flow parameter is not specified.

`services-filter`	Default value: `null`
	Use this tag to filter out services you do not wish to track with this service. The following example omits any changes to the Page service from displaying in the Recent Activities service:
	`<services-filter>`
	`  <exclude>oracle.webcenter.page</exclude>`
	`</services-filter>`
	By default, no services are omitted.

19 Integrating the RSS Service

This chapter describes how to integrate the RSS service into a custom WebCenter application. This service enables your users to add and view RSS 2.0 formatted feeds within your application.

This chapter includes the following sections:

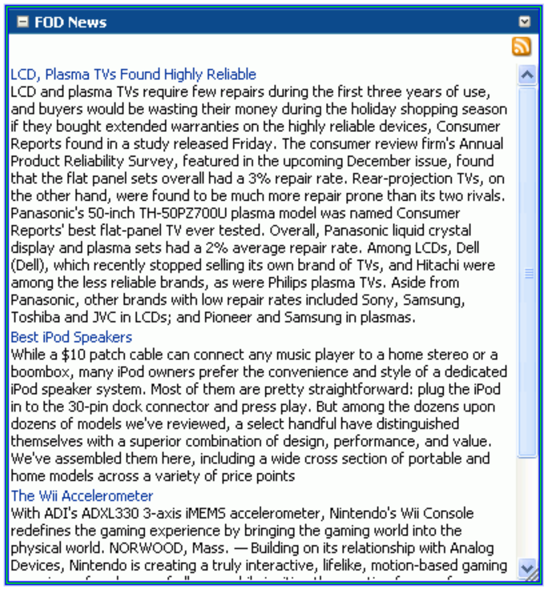
- Section 19.1, "Introduction to the RSS Service"
- Section 19.2, "Basic Configuration for the RSS Service"

19.1 Introduction to the RSS Service

Really Simple Syndication (RSS) provides a means of accessing the content of many different web sites from a single location—a news reader. The Oracle WebCenter RSS Viewer task flow is available for displaying feeds from external sources. For accessing secure application content, the RSS Viewer provides integration with external applications to provide credential mapping services to authenticate with the remote feed. For more information about using external applications, see Section 24.2, "Working with External Applications."

Oracle WebCenter Framework includes the RSS Service, which enables you to add the RSS Viewer task flow to a custom WebCenter application so that you can display content from an RSS (or an external RSS) feed. At design time, you specify a URL for an RSS feed, which your end users can then view at runtime. After deployment, any user who has permissions to modify the page can change the URL of the RSS feed. See *Oracle Fusion Middleware User's Guide for Oracle WebCenter* for more information about the service at runtime. Figure 19–1 shows a sample RSS feed at runtime.

Figure 19–1 RSS Feed at Runtime



LCD, Plasma TVs Found Highly Reliable
LCD and plasma TVs require few repairs during the first three years of use, and buyers would be wasting their money during the holiday shopping season if they bought extended warranties on the highly reliable devices, Consumer Reports found in a study released Friday. The consumer review firm's Annual Product Reliability Survey, featured in the upcoming December issue, found that the flat panel sets overall had a 3% repair rate. Rear-projection TVs, on the other hand, were found to be much more repair prone than its two rivals. Panasonic's 50-inch TH-50PZ700U plasma model was named Consumer Reports' best flat-panel TV ever tested. Overall, Panasonic liquid crystal display and plasma sets had a 2% average repair rate. Among LCDs, Dell (Dell), which recently stopped selling its own brand of TVs, and Hitachi were among the less reliable brands, as were Philips plasma TVs. Aside from Panasonic, other brands with low repair rates included Sony, Samsung, Toshiba and JVC in LCDs; and Pioneer and Samsung in plasmas.
Best iPod Speakers
While a $10 patch cable can connect any music player to a home stereo or a boombox, many iPod owners prefer the convenience and style of a dedicated iPod speaker system. Most of them are pretty straightforward: plug the iPod in to the 30-pin dock connector and press play. But among the dozens upon dozens of models we've reviewed, a select handful have distinguished themselves with a superior combination of design, performance, and value. We've assembled them here, including a wide cross section of portable and home models across a variety of price points
The Wii Accelerometer
With ADI's ADXL330 3-axis iMEMS accelerometer, Nintendo's Wii Console redefines the gaming experience by bringing the gaming world into the physical world. NORWOOD, Mass. — Building on its relationship with Analog Devices, Nintendo is creating a truly interactive, lifelike, motion-based gaming

---

Note: You can see the RSS Viewer service in action in the sample application, as described in Chapter 2, "Introduction to the WebCenter Sample Application."

19.2 Basic Configuration for the RSS Service

This section describes how to add the RSS Viewer task flow to your custom WebCenter application. You do not need to set up a connection to use this service. However, you can set up a proxy for this service, if necessary.

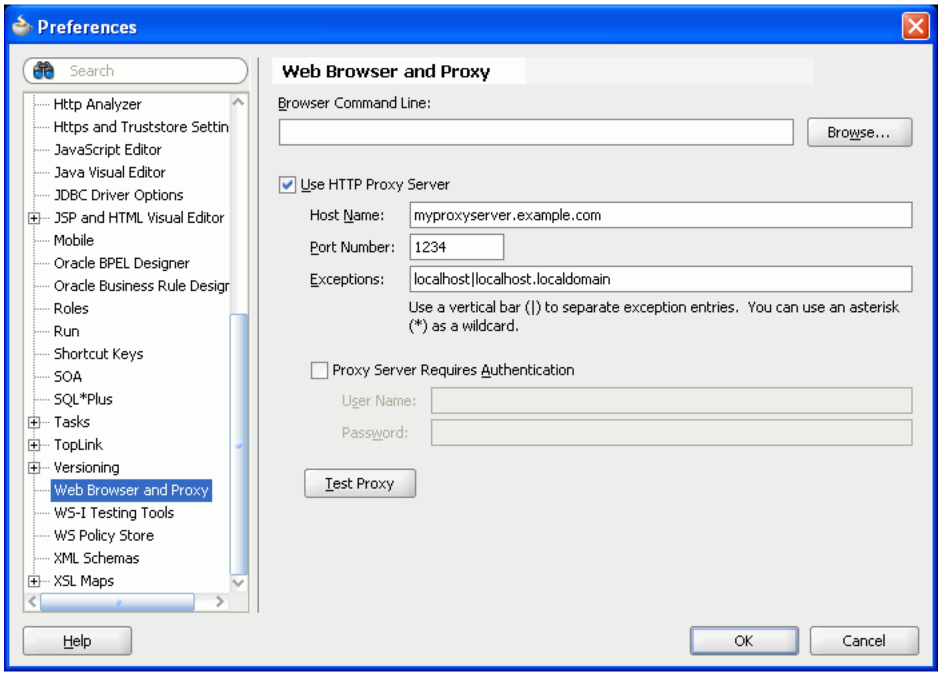
19.2.1 Setting up a Proxy Server for the RSS Service

The RSS Viewer service does not require any connections. You can simply point to the URL of the RSS feed. However, if you want to point to an RSS feed that is external to your intranet and application, you may need to set up a proxy server for your application.

To set up a proxy server for the RSS service:

1. In Oracle JDeveloper, while your application is open and selected in the Application Navigator, choose the **Tools** menu option, then select **Preferences**.
2. In the Preferences dialog, scroll down the list on the left side and select **Web Browser and Proxy.**
3. In the right pane, under Web Browser and Proxy, select **Use HTTP Proxy Server** and enter the host name and port number for your proxy server, and note any exceptions (Figure 19–2).

Figure 19–2 Setting a Proxy Server



Web Browser and Proxy

Browser Command Line:

[✓] Use HTTP Proxy Server

Host Name: myproxyserver.example.com

Port Number: 1234

Exceptions: localhost|localhost.localdomain

Use a vertical bar (|) to separate exception entries. You can use an asterisk (*) as a wildcard.

[ ] Proxy Server Requires Authentication

User Name:

Password:

19.2.2 Adding the RSS Service at Design Time

You can add the RSS Viewer task flow to your custom WebCenter application at design time to enable your users to view an RSS feed at runtime. This section describes the task flow and how to add it to your application. You will also learn how to set up an external application for RSS feeds that require authentication.

19.2.2.1 RSS Service Task Flow

The RSS service has a single task flow called RSS Viewer, which you can add to your application to enable your users to access an RSS feed. You can add multiple instances of the task flow to your application and use the Edit Task Flow Binding dialog to point to multiple RSS feed locations.

19.2.2.2 How to Add the RSS Service to Your Application

To add the RSS Viewer task flow to your application:

1. Follow the steps in Section 11.1.1, "How to Prepare Your Application to Consume Services" to implement security, if necessary, and create a customizable page in your application.
2. Open the customizable page.
3. If the RSS feed you want to use requires authentication, create an external application. If it does not require authentication, proceed to Step 4.

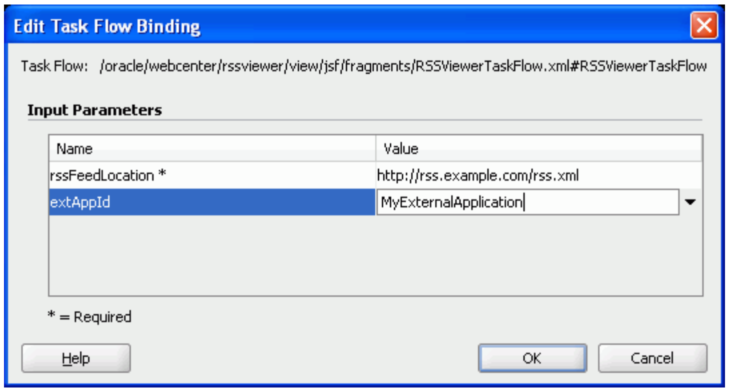
   Note: For more information about external applications, refer to Section 24.2, "Working with External Applications."

4. In the Resource Palette, open the **WebCenter Services Catalog**, then expand the Task Flows folder.

5. Click **RSS Viewer** and drag it to your page after the `<af:form>` tag, and choose **Region**. The Edit Task Flow Binding dialog displays.

6. In the Edit Task Flow Binding dialog, you can set the location of the RSS feed and enter the name (the application name, not the application display name) of the external application that the custom WebCenter application can use to authenticate RSS feed.

Figure 19–3 Example of the Edit Task Flow Binding Dialog for the RSS Viewer Service


Edit Task Flow Binding

Task Flow: /oracle/webcenter/rssviewer/view/jsf/fragments/RSSViewerTaskFlow.xml#RSSViewerTaskFlow

Input Parameters

Name	Value
rssFeedLocation *	http://rss.example.com/rss.xml
extAppId	MyExternalApplication

* = Required

Table 19–1 describes the possible values for the RSS Viewer task flow binding parameters.

Table 19–1 RSS Viewer Task Flow Binding Parameters

Parameter	Value
`rssFeedLocation`	Enter the location of the RSS feed. For example, to use the Oracle Press Releases RSS feed, enter: `${'http://www.oracle.com/rss/rss_ocom_pr.xml'}`
`extAppId`	Enter the name of the external application you wish to use to authenticate the custom WebCenter application with the RSS feed. If the RSS feed does not require authentication, you do not need to set up and identify an external application for this service.

7. Click **OK** and save your page. The binding displays on your page.

   If you look at the Source tab of your page, you can see the RSS Viewer task flow in the page source (Figure 19–4).

Figure 19–4 RSS Viewer Task Flow in the Page Source

```
<af:form>
  <af:region value="#{bindings.RSSViewerTaskFlow1.regionModel}"
           id="RSSVi1"/>
</af:form>
```

19.2.3 Setting Security for the RSS Viewer Service

To use the RSS service with a public RSS feed, you do not need to set security. To use the RSS service with an RSS feed that requires authentication, you can set up an external application for your custom WebCenter application that sets up either user credentials or public credentials for accessing the RSS feed.

Note: For secure application content, your news reader must support BASIC authentication.

For more information on using external applications, refer to Section 24.2, "Working with External Applications."

Only authenticated users can view secure RSS feeds. If a user is not authenticated and the RSS feed is secured, the user will not see any content in the RSS Viewer unless the external application specifies PUBLIC credentials.

Note: When you add the RSS Viewer task flow to your application, the View grant is automatically added to the `authenticated-role`.

20 Integrating the Search Service

This chapter explains how to integrate the Search service in a WebCenter application. It contains the following sections:

- Section 20.1, "Introduction to Search"
- Section 20.2, "Basic Configuration for the Search Service"
- Section 20.3, "Advanced Information for the Search Service"

20.1 Introduction to Search

WebCenter search provides a unified, extensible framework that enables the discovery of information and people through an intuitive user interface. It honors application security settings by returning only the results a user is authorized to view. Because a search spans all accessible services—and can be extended to span your enterprise—users are not required to switch between applications to perform multiple searches. Additionally, users can save their searches for re-use. This includes searches with complex search refinement conditions.

Any WebCenter Web 2.0 service that owns resources supports Search. This includes: Announcements, Discussions, Documents, Tags, and the Page service. You can use the adapter to Oracle Secure Enterprise Search instances to search resources outside of WebCenter.

If you have the Documents service in your application, then Search looks for terms in document names and in the documents themselves. The results include tag hits, so any matching tags on your documents and pages are also returned, with relevance based on the quality and frequency of tags.

Note: Tag results are unaffected when you refine a search. The dates and user names used to refine a search are not provided to the search engine during tagging. You can discover more information about tags, such as who added them, in the Tag Center.

Use the **Search Preferences** task flow to control which WebCenter Web 2.0 services to search. For more information, see Section 20.3.3, "Adding the Search Preferences Task Flow."

The way that search results open differs according to the service that provides them. For example, results from the Documents service open in a new browser tab or window. Results from services that provide resource viewers, such as Discussions or Announcements, open in their resource viewers. For more information about resource viewers and the Resource Action Handling framework, see Section 11.2, "Extending Your Application with Custom Components."

For more information about the Search service and task flows at runtime, see *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

Note: You can see the Search service in action in the sample application, as described in Chapter 2, "Introduction to the WebCenter Sample Application."

20.2 Basic Configuration for the Search Service

This section provides the steps for adding the Search service to your application. It contains the following subsections:

- Section 20.2.1, "Setting up Connections for the Search Service"
- Section 20.2.2, "Adding the Search Service at Design Time"
- Section 20.2.3, "Setting Security for the Search Service"

20.2.1 Setting up Connections for the Search Service

The Search service does not require a connection. However, to incorporate Oracle Secure Enterprise Search (SES) results in your result set, you must add a connection to Oracle SES. For more information, see Section 20.3.4, "Including Oracle SES Results in WebCenter Search Results."

20.2.2 Adding the Search Service at Design Time

This section describes the Search service task flows and explains how to integrate the Search service into your application. It contains the following subsections:

- Section 20.2.2.1, "Search Service Task Flows"
- Section 20.2.2.2, "How to Add the Search Service to Your Application"

20.2.2.1 Search Service Task Flows

Table 20–1 lists the Search service task flows.

Table 20–1 Search Service Task Flows

Task Flow	Definition
Search	This task flow provides a rich search experience with several options for searching and features for refining and saving search results.
Search - Saved Searches	This task flow enables you to create a simple launch pad for running saved searches within the application.
Search Preferences	This task flow enables users to select which WebCenter Web 2.0 services to search. It also provides users with a means of selecting and ordering the columns available in the search results of the **Search** task flow.
Search Toolbar	This task flow enables users to enter simple search criteria and run the search from the application. Search results are rendered as links.

20.2.2.2 How to Add the Search Service to Your Application

The **Search Toolbar** task flow offers the easiest way to add the Search service to your application To add the **Search Toolbar** task flow to your WebCenter application:

1. Follow the steps described in Chapter 3, "Preparing Your Development Environment" to implement security and create a customizable page in your application.
2. Open the page on which to add the Search service.
3. In the Resource Palette, expand **My Catalogs**, **WebCenter Services Catalog**, and **Task Flows**.
4. Drag and drop the **Search Toolbar** task flow onto the page.
5. When prompted, select **Region** as the way to create the task flow.

   If a prompt to confirm the ADF library appears, click **Add Library**.
6. Run your page with the search toolbar on it.

   In the Application Navigator, right-click this application and select **Run**. It may take a few moments for the operation to complete.

   Note: To return results, the Search service requires that other WebCenter Web 2.0 services are included in the application.

On the **Source** tab of the page where you added the toolbar, you can see the Search Toolbar task flow in the page source, similar to Example 20–1.

Example 20–1 Search Toolbar Task Flow Source

```
<af:form id="f1">
  <af:region value="#{bindings.searchtoolbar1.regionModel}" id="r1"/>
</af:form>
```

In the Source view of the *pagename*`pageDef.xml` file, you should see something similar to Example 20–2.

Example 20–2 pageDef.xml File with the Search Toolbar

```
<taskFlow id="searchtoolbar1"
     taskFlowId="/oracle/webcenter/search/controller/taskflows/
     localToolbarSearch.xml#search-toolbar"
     xmlns="http://xmlns.oracle.com/adf/controller/binding"/>
```

20.2.3 Setting Security for the Search Service

The query execution of individual services ensures that the user performing the search is able to view results. Search results are returned based on user privileges.

The Search service does not require ADF security. When unauthenticated, search queries provided by all services return only public content. However, only authenticated users of an ADF-secured WebCenter application can save searches. Unauthenticated users will see the **Save** button grayed out (See Figure 20–2).

For information about configuring ADF security, see Section 3.5, "Implementing Security in Your Application."

20.3 Advanced Information for the Search Service

This section describes optional features available with this service. It contains the following subsections:

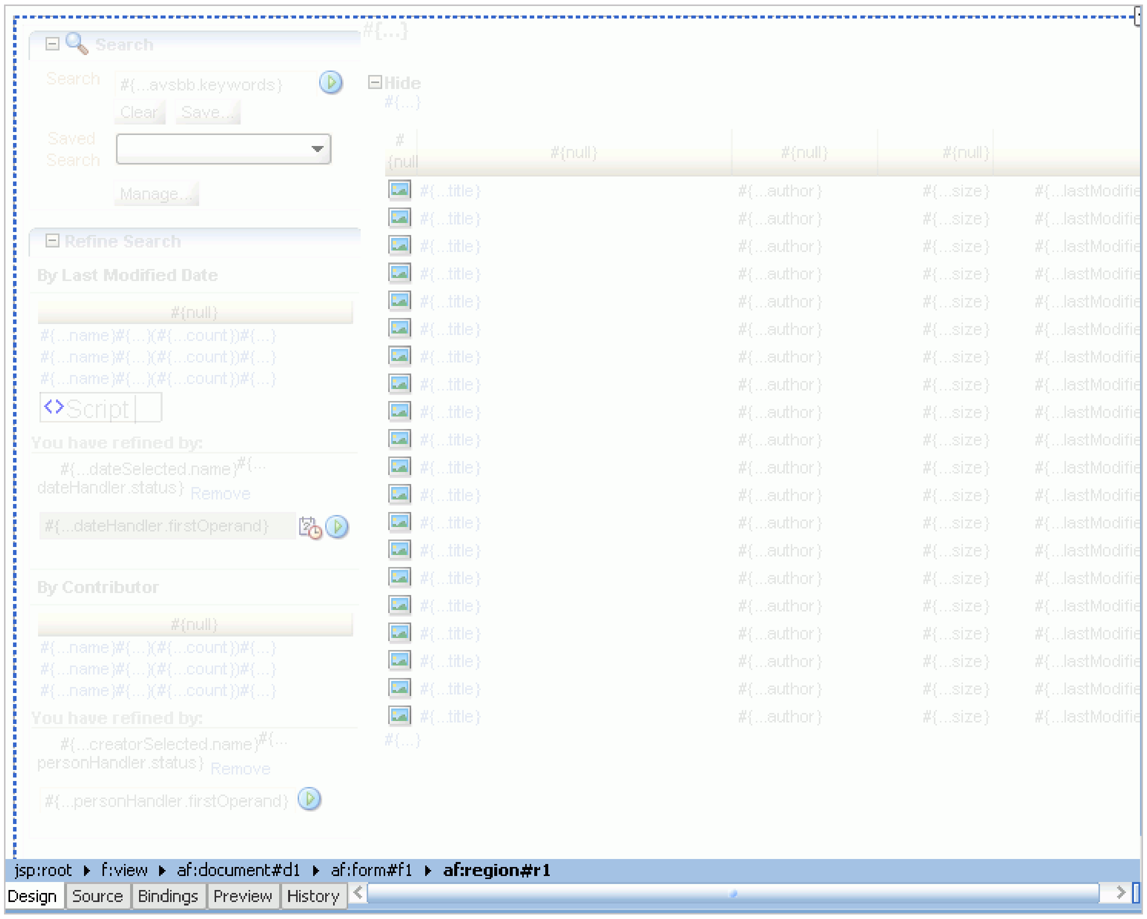
- Section 20.3.1, "Adding the Search Task Flow"
- Section 20.3.2, "Adding the Search - Saved Searches Task Flow"
- Section 20.3.3, "Adding the Search Preferences Task Flow"
- Section 20.3.4, "Including Oracle SES Results in WebCenter Search Results"
- Section 20.3.5, "Using Search Service APIs"
- Section 20.3.6, "Building Adapters for the Search Service"
- Section 20.3.7, "Troubleshooting the Search Service"

20.3.1 Adding the Search Task Flow

To add the **Search** task flow to your WebCenter application, follow the instructions provided for the **Search Toolbar** task flow in Section 20.2.2, "Adding the Search Service at Design Time," but instead, drag and drop the **Search** task flow onto the page.

In the Design view of the page, you should see something similar to Figure 20–1.

Figure 20–1 Search in Design View


Search
Search
#{...avsbb.keywords}
Clear
Save...
Saved Search
Manage...
Refine Search
By Last Modified Date
#{null}
#{...name}#{...}(#{...count})#{...}
Script
You have refined by:
#{...dateSelected.name}#{...dateHandler.status}
Remove
#{...dateHandler.firstOperand}
By Contributor
#{...creatorSelected.name}#{...personHandler.status}
#{...personHandler.firstOperand}
#{...}
Hide
#{null}
#{...title}
#{...author}
#{...size}
#{...lastModifie
jsp:root ▸ f:view ▸ af:document#d1 ▸ af:form#f1 ▸ af:region#r1
Design
Source
Bindings
Preview
History

To run the page with the Search view:

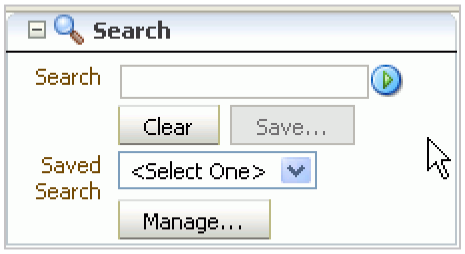
1. In the Application Navigator, right-click the page and select **Run**.

   As long as the application has been ADF-secured, you should see a login page. It may take a few moments for the operation to complete.

2. If you configured ADF security, then enter the user name and password that you provided and click **Submit**.

3. You should see your page with the Search main view (Figure 20–2).

Figure 20–2 Search Main View



4. Enter some search criteria in the field and click the **Search** icon.

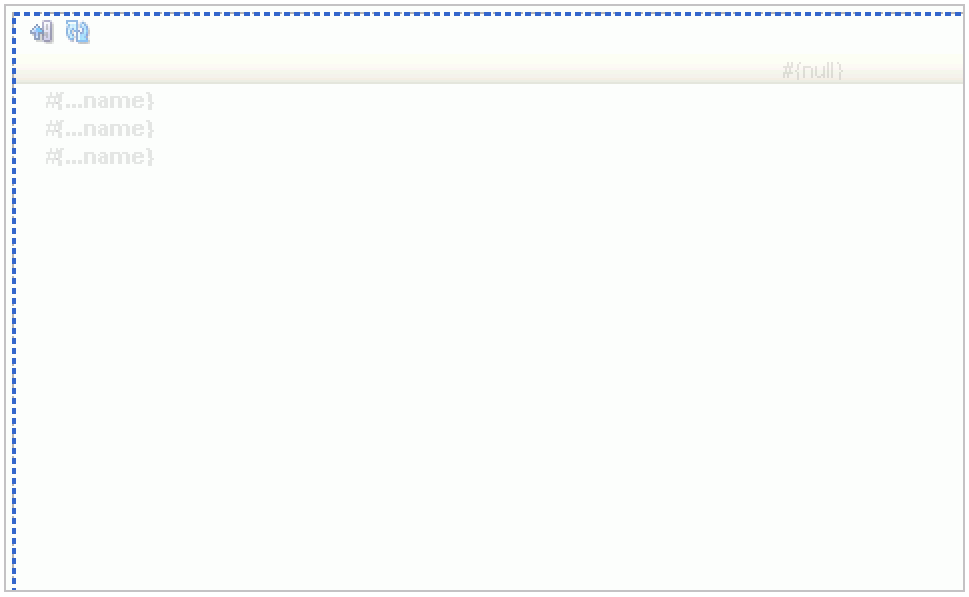
   You should see the results displayed on the page. At this time, you will get few, if any, results because the application is not populated with much besides the Search service itself.

20.3.2 Adding the Search - Saved Searches Task Flow

You can add the **Search - Saved Searches** task flow to provide a simple launch pad for running saved searches within the application. To add this task flow to your WebCenter application, follow the instructions provided for the **Search Toolbar** task flow in Section 20.2.2, "Adding the Search Service at Design Time," but instead, drag and drop the **Search - Saved Searches** task flow onto the page.

In the Design view of the page, you should see something similar to Figure 20–3.

Figure 20–3 Search - Saved Searches Task Flow



The task flow renders links that, when clicked, run a saved search.

20.3.3 Adding the Search Preferences Task Flow

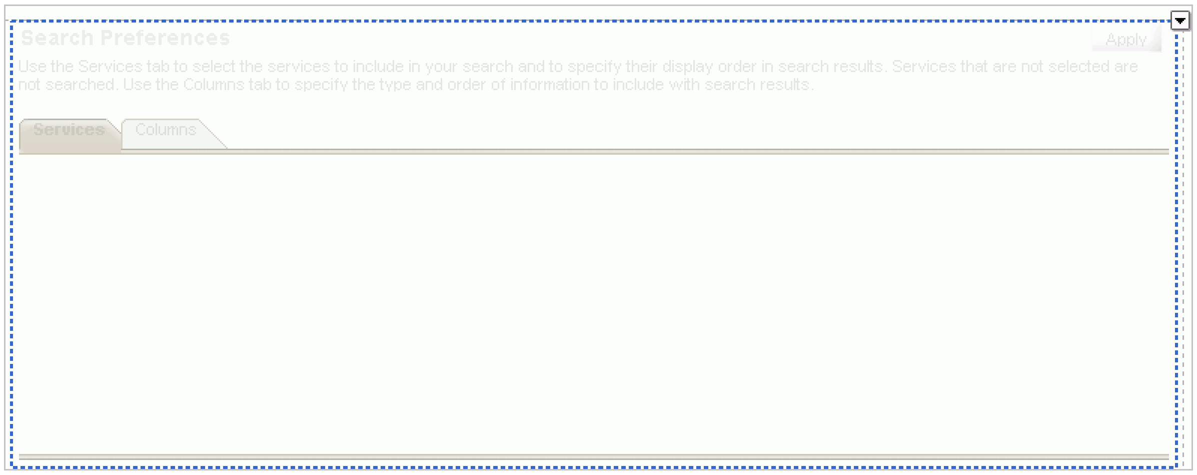
By default, WebCenter searches all WebCenter Web 2.0 service in your application. However, the **Search Preferences** task flow enables users to control that selection. Services that are not selected in preferences are not searched.

The **Search Preferences** task flow also provides a means of selecting and ordering the columns available in the results returned in the **Search** task flow.

To add the **Search Preferences** task flow to your WebCenter application, follow the instructions provided for the **Search Toolbar** task flow in Section 20.2.2, "Adding the Search Service at Design Time," but instead, drag and drop the **Search Preferences** task flow onto the page.

In the Design view of the page, you should see something similar to Figure 20–4.

Figure 20–4 Search Preferences Task Flow



Use the Services tab to select the services to include in your search and to specify their display order in search results. Services that are not selected are not searched. Use the Columns tab to specify the type and order of information to include with search results.

20.3.4 Including Oracle SES Results in WebCenter Search Results

If you have an Oracle Secure Enterprise Search (SES) instance for searching other repositories outside of Oracle WebCenter, then you can integrate it with your application. SES results appear in the same result set as WebCenter search results.

For secure searches, your WebCenter application and your Oracle SES instance must use the same identity management system.

This section describes how to include Oracle SES results in WebCenter searches and how to validate this configuration. It contains the following subsections:

- Section 20.3.4.1, "Including Oracle SES Results in WebCenter Searches"
- Section 20.3.4.2, "Validating the Oracle SES Connection"

20.3.4.1 Including Oracle SES Results in WebCenter Searches

To include Oracle SES results:

1. Add the search toolbar to a page, following Section 20.2.2, "Adding the Search Service at Design Time."
2. In the Application Navigator, expand **Application Resources**.
3. Right-click the **Connections** node, and select **Oracle Secure Enterprise Search**.
4. Enter values for the fields in the dialog.

   For example:

   a. In Oracle WebCenter, select to create the connection in either the Application Resources or IDE Connections.

      A connection in Application Resources is available only for that application, while a connection in IDE Connections is available for all applications you create. If you plan to use the connection in other applications, then select IDE Connections to avoid the need to re-create it.

      Note: If you create the connection in IDE, then the connection must be added to the application later. For example, in the Resource Palette under IDE Connections, right-click the connection and select **Add to Application**.

   b. For **Connection Name**, enter a meaningful name, for example, `SES_instance`.

   c. For **SOAP URL**, enter the URL for the Oracle SES Web Services endpoint, for example:

      ```
      http://host:port/search/query/OracleSearch
      ```

   d. Under **Proxy Login**, for **App User Name**, enter the trusted entity defined for this application and the Oracle SES instance; for example, `wpadmin`.

      You can find (or create) the trusted entity on the **Global Settings - Federation Trusted Entities** page of Oracle SES.

   e. For **App Password**, enter the password of the Federation Trusted Entities user.

   f. Select the box for **Set as default connection for the Search Service**.

This ensures that your application uses that connection to access Oracle SES. The Search service uses only one Oracle SES connection.

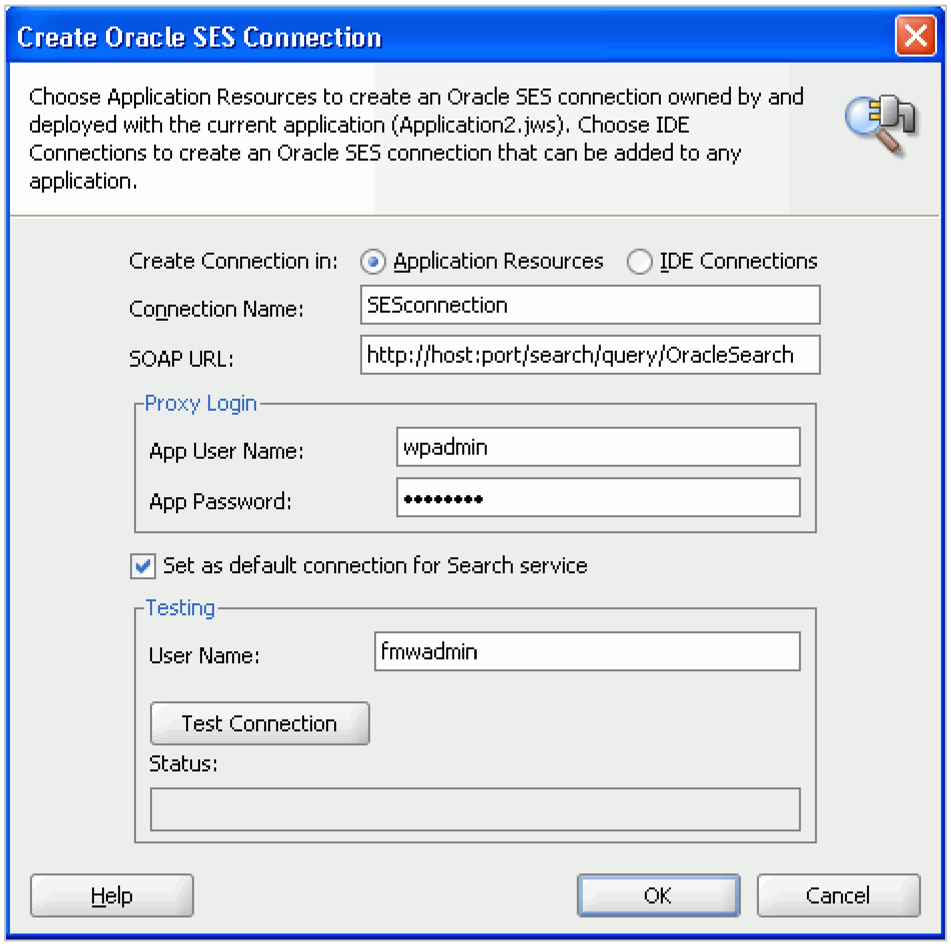
Note: After you create a connection as the default connection, you cannot edit it so that it is *not* the default. To use a different default connection, you must create a new connection and mark that as the default connection.

---

g. Under **Testing**, for **User name**, enter the name of a user present in both the Oracle Identity Management server configured for your WebCenter application and the Oracle Identity Management server configured for Oracle SES.

   You should see something similar to Figure 20–5.

Figure 20–5 Create Oracle SES Connection Dialog



Create Oracle SES Connection

Choose Application Resources to create an Oracle SES connection owned by and deployed with the current application (Application2.jws). Choose IDE Connections to create an Oracle SES connection that can be added to any application.

Create Connection in: [✓ Application Resources] IDE Connections

Connection Name: SESconnection

SOAP URL: http://host:port/search/query/OracleSearch

Proxy Login

App User Name: wpadmin

App Password: •••••••••

[✓] Set as default connection for Search service

Testing

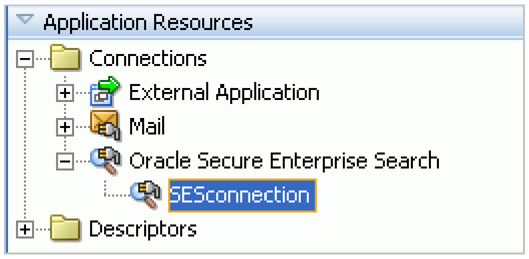
User Name: fmwadmin

Status:

5. Click **Test Connection**.
6. If the connection is successful, then click **OK**.

After you have included the Oracle SES connection in your application, you should see it in your Application Resources panel, as shown in Figure 20–6.

Figure 20–6 Oracle SES Connection in Application Resources



The Oracle SES connection is now included in the `connections.xml` file and is the default connection in the `adf-config.xml` file, as shown in Example 20–3.

Example 20–3 Oracle SES Connection in adf-config.xml

```
<ses-properties>
  <connection>GenericSesConnection</connection>
  <data-group>oracle-sites</data-group>
</ses-properties>
```

The Oracle Secure Enterprise Search connection wizard does not provide a means of setting the `data-group` value in `adf-config.xml`. After the connection is established, you must set it manually. Check the `data-group` on your Oracle SES instance to set to the correct value.

You can also set the `data-group` value to be any one of the data source groups created in your Oracle SES instance. Data source groups present a good way to segment your searchable data on Oracle SES. Given an Oracle SES instance that keeps a search index on all corporate data, you can use data source groups to make only a portion of your corporate data available to a WebCenter search. For more information on creating data groups, see the *Oracle SES Administrator's Guide*.

Note: While you can set up the connections to back-end servers at design time in Oracle JDeveloper, you can later add, delete, or modify connections in your deployed environment using Enterprise Manager Fusion Middleware Control. For more information, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

20.3.4.2 Validating the Oracle SES Connection

To validate the inclusion of Oracle SES results in WebCenter search:

1. In the Application Navigator, right-click the page you created with the Search Toolbar, and click **Run**.

   As long as the application has been ADF-secured, you should see a login page. It may take a few moments for the operation to complete.

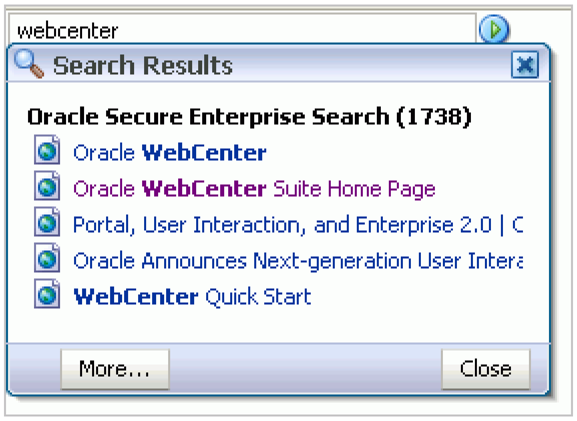
2. Enter the user name and credentials that you created when you defined security.

3. Click **Submit**.

   It may take a few moments for the page to load. When it completes, you should see your search toolbar.

   Figure 20–7 shows example results for the search term *webcenter*.

Figure 20–7 Search Results Dialog with Oracle SES Included


webcenter

Search Results

Oracle Secure Enterprise Search (1738)

Oracle **WebCenter**

Oracle **WebCenter** Suite Home Page

Portal, User Interaction, and Enterprise 2.0 | C

Oracle Announces Next-generation User Interа

WebCenter Quick Start

Note: You can use a web service data control to call the searching function of Oracle SES. For more information, see Appendix D, "Calling Oracle SES to Search Data."

For more information on Oracle SES, see the Oracle Secure Enterprise Search documentation that comes with the product.

20.3.5 Using Search Service APIs

Custom components can implement Search service APIs to expose their content to a WebCenter search. For the Javadoc regarding available APIs, see *Oracle Fusion Middleware Web 2.0 Services Java API Reference for Oracle WebCenter*.

20.3.6 Building Adapters for the Search Service

By default, the Search service searches any WebCenter Web 2.0 service in your application. In some cases, you may want to add other sources to your search.

You can extend WebCenter search through search adapters. WebCenter framework automatically discovers these search adapters and consolidates them in formulating search results in your custom WebCenter application. Subsequently, when you expose your custom component to the Search service in your application, it is included in the federated search.

This section describes how to extend WebCenter search through search adapters. It contains the following subsections:

- Section 20.3.6.1, "How to Add a Search Source"
- Section 20.3.6.2, "How to Register a Custom Adapter"
- Section 20.3.6.3, "Search Adapter Attributes"
- Section 20.3.6.4, "What Happens at Runtime"

20.3.6.1 How to Add a Search Source

To add a new search source, you must create Java classes for the new source to manage the incoming queries and return search results. After you add the necessary Java classes to your application, you must register them with the Search service.

When performing a search, the search framework calls `QueryManager.createRowQueryHandler` with a query object and a `List<QName>` that describes the columns sought. In return, the framework receives a `QueryHandler<Row>`.

A `QueryHandler` can be a `QueryFederator` or a `QueryExecutor`. A `QueryFederator` is just a collection of other `QueryHandlers`. With the `QueryFederator`, the framework can get a list of the child `QueryHandlers` and keep recursing until all `QueryExecutors` are found.

The framework calls the `execute` method on the `QueryExecutor` to get a `QueryResult<Row>`, which is an extension of `java.util.Iterator<Row>` that iterates and retrieves rows of results.

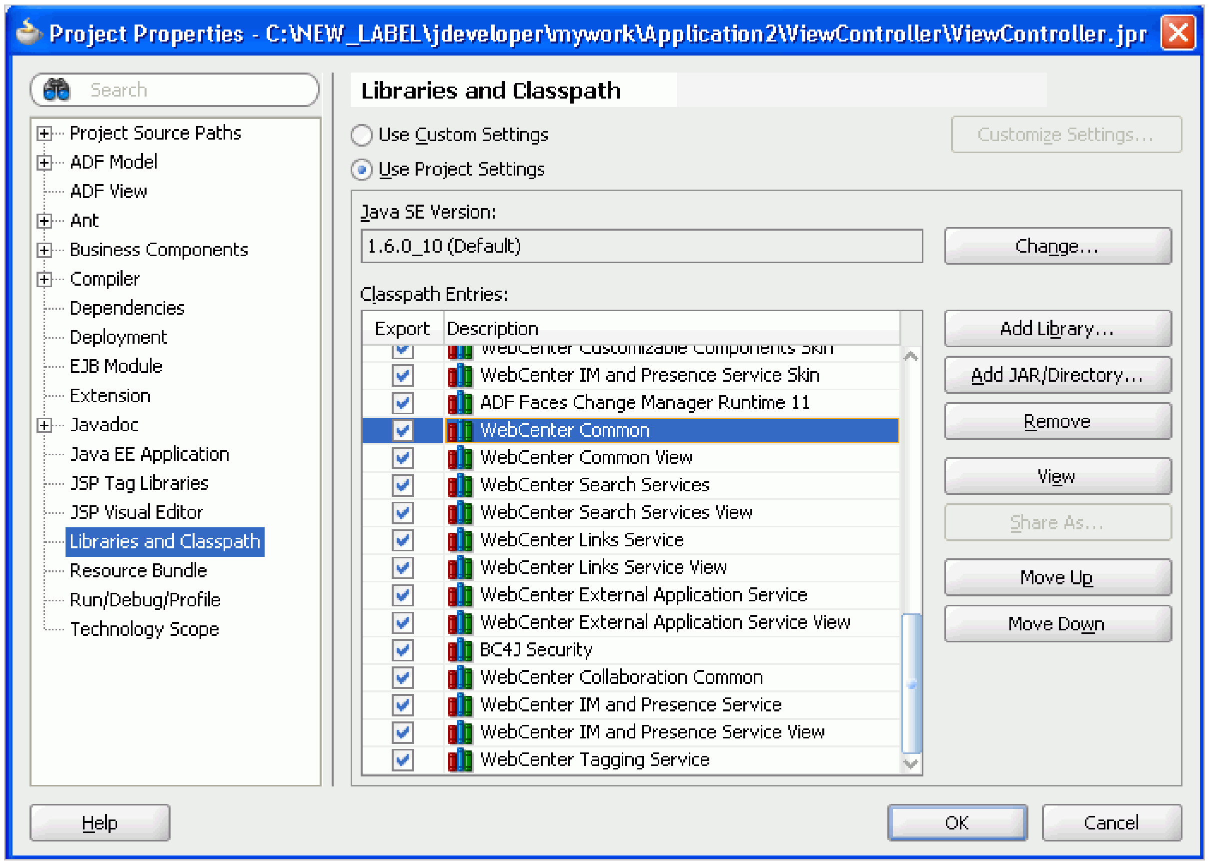
One possible implementation would be to create these classes:

- `SampleQueryManager` ...
- `SampleRowQueryExecutor` ...
- `SampleRowQueryResult` ...
- `SampleRow` ...

To add a new search source with a similar query manager:

1. In Oracle JDeveloper, open the application in which to add the search source.
2. Confirm that the WebCenter Common library is included in your project:
   a. Right-click your **ViewController** project and select **Project Properties** and then **Libraries and Classpath**.
   b. If the **WebCenter Common** library is not included, as shown in Figure 20–8, then click **Add Library** and select it.

Figure 20–8 WebCenter Common Library



Libraries and Classpath

( ) Use Custom Settings
(●) Use Project Settings

Java SE Version: 1.6.0_10 (Default)

Classpath Entries:

Export	Description
✓	WebCenter Customizable Components Skin
✓	WebCenter IM and Presence Service Skin
✓	ADF Faces Change Manager Runtime 11
✓	WebCenter Common
✓	WebCenter Common View
✓	WebCenter Search Services
✓	WebCenter Search Services View
✓	WebCenter Links Service
✓	WebCenter Links Service View
✓	WebCenter External Application Service
✓	WebCenter External Application Service View
✓	BC4J Security
✓	WebCenter Collaboration Common
✓	WebCenter IM and Presence Service
✓	WebCenter IM and Presence Service View
✓	WebCenter Tagging Service

c. Click **OK**.

3. Create a class called `SampleQueryManager.java` (Example 20–4).

Example 20–4 SampleQueryManager.java

```
package mycompany.myproduct.myapp.mycomponent.query;
import java.util.List;
import oracle.webcenter.search.QName;
import oracle.webcenter.search.Query;
import oracle.webcenter.search.QueryExecutor;
import oracle.webcenter.search.QueryManager;
import oracle.webcenter.search.Row;
public class SampleQueryManager
  implements QueryManager
{
  public SampleQueryManager()
  {
  }
  public QueryExecutor<Row> createRowQueryHandler(Query query,
                                                  List<QName> columns)
  {
    return new SampleRowQueryExecutor(query, columns);
  }
}
```

4. Create a class called `SampleRowQueryExecutor.java` (Example 20–5).

Example 20–5 SampleRowQueryExecutor.java

```
package mycompany.myproduct.myapp.mycomponent.query;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import static oracle.webcenter.search.AttributeConstants.DCMI_EXTENT;
import static oracle.webcenter.search.AttributeConstants.DCMI_MODIFIED;
import static oracle.webcenter.search.AttributeConstants.DCMI_CREATOR;
import static oracle.webcenter.search.AttributeConstants.DCMI_TITLE;
import static oracle.webcenter.search.AttributeConstants.DCMI_URI;
import static oracle.webcenter.search.AttributeConstants.DCMI_IDENTIFIER;
import static oracle.webcenter.search.AttributeConstants.WPSAPI_ICON_PATH;
import oracle.webcenter.search.QName;
import oracle.webcenter.search.Query;
import oracle.webcenter.search.QueryExecutor;
import oracle.webcenter.search.QueryHandler;
import oracle.webcenter.search.QueryResult;
import oracle.webcenter.search.Row;
public class SampleRowQueryExecutor
  implements QueryExecutor<Row>
{
  private static final int PRESET_ESTIMATED_RESULT_COUNT = 10;
  private static final int PRESET_VALUES_BATCH_SIZE = 5;
  private static final int MAX_RESULT_LIMIT = 100;
  private int m_resultStart;
  private int m_resultLimit;
  private Query m_query;
  private List<QName> m_columns;
  private Map<String, String> m_properties;
  public SampleRowQueryExecutor(Query query, List<QName> columns)
  {
    m_query = query;
    m_columns = columns;
    m_properties = new HashMap<String, String>();
    // In actuality you may want to make sure this string is translatable
    m_properties.put(QueryHandler.TITLE, "Sample");
  }
  /**
   * Create a number of rows for "batchRepeat" times.
   */
  private static void createRows(List<Row> rows, int batchRepeat)
  {
    while (batchRepeat-- > 0)
    {
      rows.add(new SampleRow("23847", "Initial Sorting of Tables",
                             "Bar", "2006-06-08", 5120));
      rows.add(new SampleRow("4827445", "Support for facelets?",
                             "Foo", "2006-04-17", 1400000));
      rows.add(new SampleRow("952787", "For Thomas Kincade's validation demo",
                             "Tiger", "2006-01-09", 1300000));
      rows.add(new SampleRow("4394588", "Page not showing in WYSIWYG fashion",
                             "Ken Swartz", "2006-04-27", 3000));
      rows.add(new SampleRow("3920", "Tree table",
                             "Scott", "2006-03-24", 1200));
    }
  }
  /**
   * Instead of doing a query execution here,
   * we'll just create a bunch of rows.
   */
  public QueryResult<Row> execute()
  {
    List<Row> rows = new ArrayList<Row>();
    // We want to hit a certain count here but not go over the max
    int realLimit = Math.min(MAX_RESULT_LIMIT, m_resultLimit);
    createRows(rows, (realLimit / PRESET_VALUES_BATCH_SIZE));
    // Create the QueryResult
    QueryResult<Row> ret = new SampleRowQueryResult(rows.iterator());
    ret.getProperties().put(QueryResult.ESTIMATED_RESULT_COUNT,
                            Integer.toString(PRESET_ESTIMATED_RESULT_COUNT));
    return ret;
  }
  /**
   * Remember the result start.
   */
  public void setResultStart(int resultStart)
  {
    m_resultStart = resultStart;
  }
  /**
   * Store away the result limit to use in the execute
   * so that we only get that number of results back.
   */
  public void setResultLimit(int resultLimit)
  {
    m_resultLimit = resultLimit;
  }
  /**
   * Expose our properties map containing our title.
   */
  public Map<String, String> getProperties()
  {
    return m_properties;
  }
}
```

5. Create a class called `SampleRowQueryResult.java` (Example 20–6).

Example 20–6 SampleRowQueryResult.java

```
package mycompany.myproduct.myapp.mycomponent.query;
import java.util.Iterator;
import oracle.webcenter.search.util.WrapperQueryResult;
import oracle.webcenter.search.Row;
public class SampleRowQueryResult extends WrapperQueryResult<Row>
{
  public SampleRowQueryResult(Iterator<Row> rows)
  {
    super(rows);
  }
}
```

6. Create a class called `SampleRow.java` (Example 20–7).

Example 20–7 SampleRow.java

```
package mycompany.myproduct.myapp.mycomponent.query;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import oracle.webcenter.search.QName;
import oracle.webcenter.search.Row;
import static oracle.webcenter.search.AttributeConstants.DCMI_EXTENT;
import static oracle.webcenter.search.AttributeConstants.DCMI_MODIFIED;
import static oracle.webcenter.search.AttributeConstants.DCMI_CREATOR;
import static oracle.webcenter.search.AttributeConstants.DCMI_TITLE;
import static oracle.webcenter.search.AttributeConstants.DCMI_URI;
import static oracle.webcenter.search.AttributeConstants.DCMI_IDENTIFIER;
import static oracle.webcenter.search.AttributeConstants.WPSAPI_ICON_PATH;
public class SampleRow implements Row
{
  private Map<QName, Object> m_storage = new HashMap<QName, Object>();
  private DateFormat m_dateFormat = new SimpleDateFormat("yyyy-MM-dd");
  SampleRow(String id, String title,
            String creator, String lastModified,
            int size)
  {
    m_storage.put(DCMI_IDENTIFIER, id);
    // Only if you have an external URL to go to
    // m_storage.put(DCMI_URI, "http://my.external.url");
    m_storage.put(DCMI_TITLE, title);
    m_storage.put(DCMI_CREATOR, creator);
    // The below path points to a path accessible in the classpath to an icon
    m_storage.put(WPSAPI_ICON_PATH, "/adf/webcenter/search_qualifier.png");
    try
    {
      Date date = m_dateFormat.parse(lastModified);
      Calendar cal = Calendar.getInstance();
      cal.setTime(date);
      m_storage.put(DCMI_MODIFIED, cal);
    }
    catch (ParseException e)
    {
      e.printStackTrace();
    }
    m_storage.put(DCMI_EXTENT, new Long(size));
  }
  public Object getObject(QName columnName)
  {
    return m_storage.get(columnName);
  }
  public Iterator<QName> getColumns()
  {
    return m_storage.keySet().iterator();
  }
}
```

20.3.6.2 How to Register a Custom Adapter

To register your query manager with the Search service, you must edit the `service-definition.xml` file. The `SampleQueryManager` line exposes the custom adapter's query manager implementation to the Search service so that the custom adapter is included with any search user interface.

The entry should be similar to Example 20–8.

Example 20–8 Sample Query Manager Entry in service-definition.xml

```
<service-definition id="mycompany.myproduct.myapp.mycomponent"
   version="11.1.1.0.0">
  <resource-view taskFlowId="/somepath/somefile.xml#someId"
                  authorizerClass="some.service.id.Authorizer1"/>
  <name>Custom Component 1</name>
  <description>Description for Custom Component 1</description>
  <search-definition xmlns="http://xmlns.oracle.com/webcenter/search"
      id="mycompany.myproduct.myapp.mycomponent.query" version="11.1.1.0.0">
  <query-manager-class>
mycompany.myproduct.myapp.mycomponent.query.SampleQueryManager
  </query-manager-class>
  </search-definition>
</service-definition>
```

For more information about the `resource-view` tag (and Oracle WebCenter's Resource Action Handling framework), see Section 11.2, "Extending Your Application with Custom Components."

20.3.6.3 Search Adapter Attributes

This section describes the attributes available for searching a service. It contains the following subsections:

- Section 20.3.6.3.1, "Searchable Attributes"
- Section 20.3.6.3.2, "Keywords"
- Section 20.3.6.3.3, "Date Condition"
- Section 20.3.6.3.4, "Complex Date Condition"
- Section 20.3.6.3.5, "Person Condition"
- Section 20.3.6.3.6, "Complex Person Condition"
- Section 20.3.6.3.7, "Access"
- Section 20.3.6.3.8, "Selectable Attributes"

20.3.6.3.1 Searchable Attributes The WebCenter search user interface exposes searches based on three fields: keywords, date, and person. Each service, in its own search implementation, honors these fields to the best of its capabilities. At minimum, it supports the keywords field. When you define your own search adapter, you must honor these fields to the best of your adapter's capabilities, as described in Section 20.3.6.3.8, "Selectable Attributes."

20.3.6.3.2 Keywords By default, the keyword field is in `oracle.webcenter.search.TextPredicate`. The field can comprise multiple words, with the service determining how to interpret multiple words. The WebCenter Search service does not tokenize before the call comes to the service, but here are some guidelines:

- If two or more tokens are enclosed within quotes, then they should be interpreted as one word.
- Unless the `OR` operator is explicitly set, it is assumed that the default conjunction among the words in the keywords field is `AND`.

20.3.6.3.3 Date Condition For the Search service to support filtering of query results by last modified date, the service implementation of the WPSAPI must support the use of `oracle.webcenter.search.AttributePredicate`, accessible from the query object, that makes use of `oracle.webcenter.search.AttributeConstants.DCMI_MODIFIED`. The supported comparators must include the following:

- `oracle.webcenter.search.ComparatorConstants.GREATER_THAN`
- `oracle.webcenter.search.ComparatorConstants.GREATER_THAN_OR_EQUALS`

They may also include the following:

- `oracle.webcenter.search.ComparatorConstants.EQUALS`
- `oracle.webcenter.search.ComparatorConstants.NOT_EQUALS`
- `oracle.webcenter.search.ComparatorConstants.LESS_THAN`
- `oracle.webcenter.search.ComparatorConstants.LESS_THAN_OR_EQUALS`

The literals for comparison should be of type `java.util.Calendar`.

20.3.6.3.4 Complex Date Condition You can use the `BETWEEN` clause for dates. The complex date condition, instead of being a simple `AttributePredicate`, is a `ComplexPredicate` that contains one `GREATER_THAN AttributePredicate` and one `LESS_THAN AttributePredicate` with an `AND` conjunction operator. This `ComplexPredicate` must apply the `AND` operator with the rest of the fields (that is, keywords and person).

To differentiate between this case and the default case, service authors must check on the type of the predicate that is meant for the date condition. If it is an `AttributePredicate`, then it is the default case. If it is a `ComplexPredicate`, then it is a special case.

20.3.6.3.5 Person Condition For the Search service to support filtering of query results by person, the service implementation of the WPSAPI must support the use of `oracle.webcenter.search.AttributePredicate`, accessible from the query object, that makes use of the `oracle.webcenter.search.AttributeConstants.WPSAPI_MODIFIER`.

The comparators supported must include the following:

- `oracle.webcenter.search.ComparatorConstants.EQUALS`
- `oracle.webcenter.search.ComparatorConstants.CONTAINS`
- `oracle.webcenter.search.ComparatorConstants.STARTS_WITH`
- `oracle.webcenter.search.ComparatorConstants.ENDS_WITH`

The literals for comparison should be of type `java.lang.String`.

20.3.6.3.6 Complex Person Condition Service authors must check on the type of predicate that is meant for the person condition. An `AttributePredicate` is the default case.

20.3.6.3.7 Access The Search service (as a caller) expects that the query's `getPredicate()` method will likely return an `oracle.webcenter.search.ComplexPredicate` that joins an `oracle.webcenter.search.TextPredicate` holding the keyword, and an `oracle.webcenter.search.ComplexPredicate` holding the date condition, and an `oracle.webcenter.search.AttributePredicate` holding the person condition.

20.3.6.3.8 Selectable Attributes The following selectable attributes/columns should be supported by your search adapter in its search implementation:

- **Title** (Required)

  For each resource returned, the Search service user interface can display the title column, if provided. It is strongly recommended that the title column comes with the QName `AttributeConstants.DCMI_TITLE`.

- **Resource ID** (Required)

  For resources to communicate a unique identifier that allows the Search service to render a link to navigate to the resource view (or any declared views) of the service, the column with the name `oracle.webcenter.search.AttributeConstants.DCMI_IDENTIFIER` must be returned as one of the columns accessible from a row within the `QueryResult<Row>`.

  The Search service extracts the `DCMI_IDENTIFIER` column and feeds it into the resource viewer as the value of the `resourceId` parameter when the link is clicked.

  If your resource view (or any declared views) requires more than one column in the primary key, then try to bundle all of them into a composite string value for the `oracle.webcenter.search.AttributeConstants.DCMI_IDENTIFIER` column. The Search service treats that opaquely and passes it to your resource view, within which it may deconstruct the string into the various parts of the primary key.

- **Resource Type**

  For each resource returned, the Search service user interface can display the type column, if provided. The type column comes with the QName `AttributeConstants.DCMI_TYPE`. If the `DCMI_TYPE` column is not found, then the value of the `DCMI_FORMAT` column is used in its place. In your resource viewer, you can make use of the resource type to aid in the rendering of a resource.

- **Resource URL** (in place of Resource ID)

  In the absence of an `oracle.webcenter.search.AttributeConstants.DCMI_IDENTIFIER` column, the Search service constructs the link from the value of the column specified by `oracle.webcenter.search.AttributeConstants.DCMI_URI`. With absolute URLs, the `DCMI_URI` can be used to launch a link from the search results to the URL.

  This allows for services, such as the Documents service and the Page service, to invoke the browser plug-in to render various results link targets, instead of invoking a resource view.

- **Last Modified Date**

For each resource returned, the Search service user interface can display the last modified date column, if provided. To mirror the searchable attribute of the last modified date, it is important that users can see the last modified date as a returned column in the search result.

For most users, the last modified date is more relevant than the creation date. To have better consistency between the searchable attributes and the selectable attributes (so that users know what date to search with after seeing some values in the columns), it is strongly recommended that service authors return `AttributeConstants.DCMI_MODIFIED` rather than `AttributeConstants.DCMI_CREATED`. The last modified date is of type `java.util.Calendar`.

- **Creator**

  For each resource returned, the Search service user interface can display the creator and last modified by column, if provided. The search user interface lumps them together in the same column as a list of contributors for the found resource. The QNames to use are `AttributeConstants.DCMI_CREATOR` and `AttributeConstants.WPSAPI_MODIFIED`, respectively. The creator is of type `java.lang.String`.

- **Icon**

  For each resource returned, the Search service user interface can display the icon column, if provided. The icon column comes with the QName `AttributeConstants.WPSAPI_ICON_PATH` and corresponds to a valid path to an icon file that is accessible in the class path. The icon path is of type `java.lang.String`.

- **Size**

  For each resource returned, the Search service user interface can display the size column, if provided. The size column comes with the QName `AttributeConstants.DCMI_EXTENT` and corresponds to the size of the resource found. The size is of type `java.lang.Number`.

20.3.6.4 What Happens at Runtime

When you perform a search from the main view or toolbar, it should seamlessly include your new search source. For testing purposes, be sure to enter criteria that will yield a hit in your newly-added search source.

The Search service invokes a resource viewer using the Resource Action Handling framework. For information about registering a resource viewer to allow custom components to be found using Search, see Section 11.2, "Extending Your Application with Custom Components."

20.3.7 Troubleshooting the Search Service

This section describes common problems and solutions for the Search service.

Problem

There is a set timeout for querying each service. If a particular service does not return search results in the allotted time, then it will time out. The Announcements and Discussions services are susceptible to timeouts. When a WebCenter Spaces instance is heavily used, the Page service and Group Spaces could have timeouts from search execution.

Solution

To improve performance in design-time, you can increase the values (in milliseconds) of `timeoutMs` and prepare `TimeoutMs` in `adf-config.xml`. The following is the relevant fragment of `adf-config.xml`, found in the Application Resources panel in the `ADF META-INF` folder:

```
<execution-properties timeoutMs="3000"prepare TimeoutMs="1000"/>
```

At runtime, you can configure timeouts using WLST or Enterprise Manager.

21 Integrating the Tags Service

This chapter explains how to integrate the Tags service in a WebCenter application.

This chapter includes the following sections:

- Section 21.1, "Introduction to the Tags Service"
- Section 21.2, "Basic Configuration for the Tags Service"
- Section 21.3, "Advanced Information for the Tags Service"

21.1 Introduction to the Tags Service

Tags enable users to apply their own meaningful terms to items, making those items more easily discoverable in search results and the Tag Center. The Tag Center is a page that displays the interactions between all the tags, tagged items, and their taggers in a WebCenter application.

Having multiple users tag objects contributes to the collective knowledge, which makes searching much more relevant. WebCenter search takes advantage of the knowledge captured by tagging by indicating the relevance of results based on the quality and frequency of their applied tags.

Tags let the *users* of data (instead of the *publishers* of data) classify information. In this way, tags act as a personal productivity tool as well as a method to increase searchability for everyone.

For example, suppose your application includes a component that provides a view of departmental human resources contacts. If you enable tags for this component, a user who comes to it can bookmark it for themselves and assign tags like HR or contacts or department that will make it easier for others to find it as well.

The Tags service is available for pages, documents, and custom objects.

21.1.1 Understanding the Tags Service

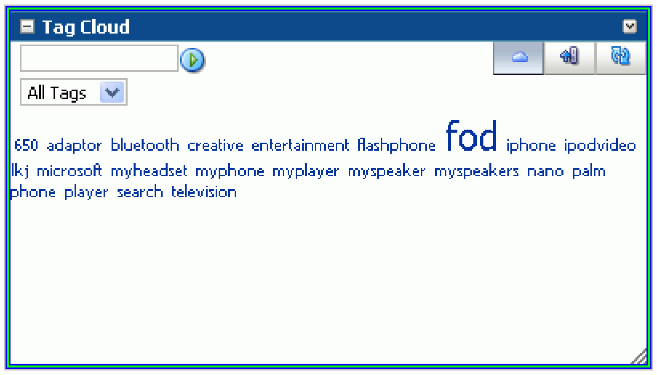
You can apply one or more meaningful terms, called *tags*, to remind yourself and alert other users of the content they might expect to find at the tagged location. Anywhere you see the **Tags** icon (Figure 21–1), you can apply a tag.

Figure 21–1 The Tags Icon

Tags let you apply your own classifications to items. For example, you could apply the tag `phone` to a product page that provides useful information about new phones. When you or other users search for phone, the tagged page displays in the search results.

When you access the Tag Center, you see all users who applied the same tag anywhere else in the application. Within the Tag Center, a tag cloud (Figure 21–2) displays all currently applied tags.

Figure 21–2 A Tag Cloud


Tag Cloud
All Tags
650 adaptor bluetooth creative entertainment flashphone fod iphone ipodvideo
lkj microsoft myheadset myphone myplayer myspeaker myspeakers nano palm
phone player search television

A tag cloud is a visual depiction of all tags. Tags are presented according to the frequency of their use—the larger the font, the more the tag has been applied to items. Click a tag in the tag cloud to run a search that returns a list of all items that use the tag.

For more information about the service at runtime, see *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

Note: You can see the Tags service in action in the sample application, as described in Chapter 2, "Introduction to the WebCenter Sample Application."

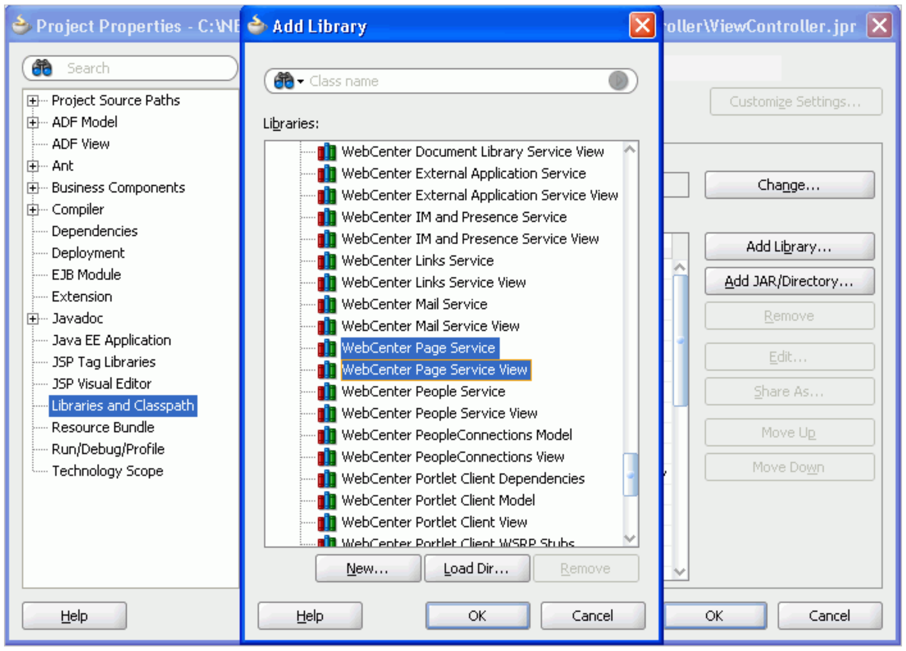
21.1.2 Requirements for the Tags Service

The Tags service requires that you set up the WebCenter schema and create a database connection to the schema. Tag information is stored in the database. For details, see Section 21.2.1, "Setting up Connections for the Tags Service."

Note: Typically, when you add tags, you will also add a search toolbar to enable searching for the objects with the tags. To add the search toolbar, follow the instructions in Section 20.2.2, "Adding the Search Service at Design Time."

There are no additional requirements to use the Tags service. However, to enable the ability to tag *pages*, you must manually add the Page service libraries to your project. Right-click your **ViewController** project and select **Project Properties** and then **Libraries and Classpath**. Click **Add Library** and select the **WebCenter Page Service** and **WebCenter Page Service View** libraries, as shown in Figure 21–3. Click **OK**.

Figure 21–3 Page Service Libraries

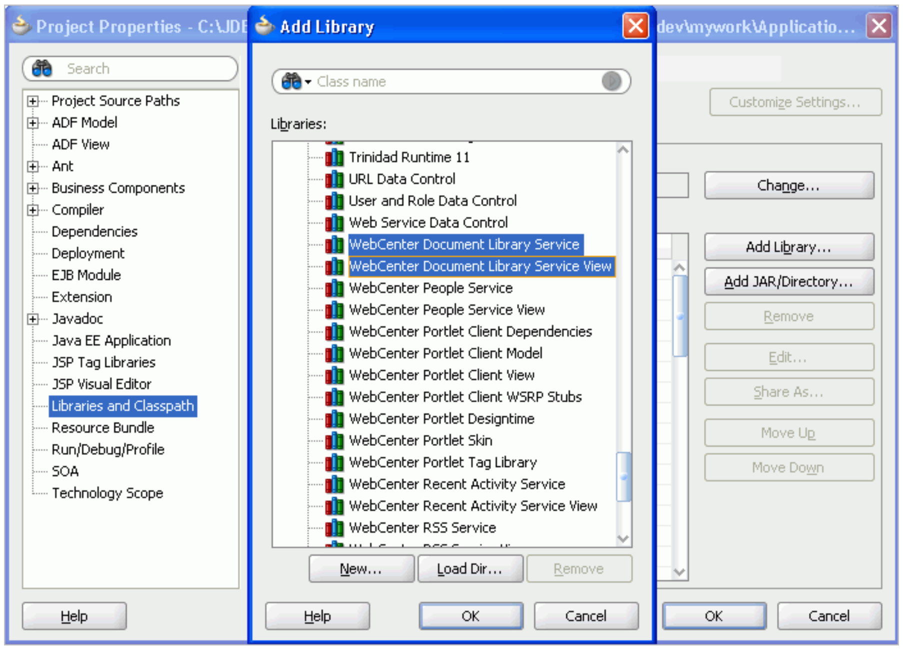


Libraries:

- WebCenter Document Library Service View
- WebCenter External Application Service
- WebCenter External Application Service View
- WebCenter IM and Presence Service
- WebCenter IM and Presence Service View
- WebCenter Links Service
- WebCenter Links Service View
- WebCenter Mail Service
- WebCenter Mail Service View
- WebCenter Page Service
- WebCenter Page Service View
- WebCenter People Service
- WebCenter People Service View
- WebCenter PeopleConnections Model
- WebCenter PeopleConnections View
- WebCenter Portlet Client Dependencies
- WebCenter Portlet Client Model
- WebCenter Portlet Client View
- WebCenter Portlet Client WSRP Stubs

Similarly, to enable the ability to tag *documents*, you must manually add the Document Library service libraries to your project, as shown in Figure 21–4.

Figure 21–4 Documents Service Libraries



Libraries:

- Trinidad Runtime 11
- URL Data Control
- User and Role Data Control
- Web Service Data Control
- WebCenter Document Library Service
- WebCenter Document Library Service View
- WebCenter People Service
- WebCenter People Service View
- WebCenter Portlet Client Dependencies
- WebCenter Portlet Client Model
- WebCenter Portlet Client View
- WebCenter Portlet Client WSRP Stubs
- WebCenter Portlet Designtime
- WebCenter Portlet Skin
- WebCenter Portlet Tag Library
- WebCenter Recent Activity Service
- WebCenter Recent Activity Service View
- WebCenter RSS Service

21.2 Basic Configuration for the Tags Service

This section describes required steps for adding this service to your application.

21.2.1 Setting up Connections for the Tags Service

The Tags service requires a connection to the database with the WebCenter schema installed. The script to create the WebCenter schema is included with the product.

For details about installing the database and the WebCenter schema, see *Oracle Fusion Middleware Installation Guide for Oracle WebCenter*.

21.2.1.1 How to Set Up Connections for the Tags Service

To create the database connection:

1. In the Application Navigator, expand the **Application Resources** pane.
2. Right-click **Connections**, then click **New Database.**
3. Enter the following information for your database connection:
   - **Connection Name**: `WebCenter`
   - **Connection Type**: `Oracle (JDBC)`
   - **Username:** `username`
   - **Password**: `password`
   - **Host**: `<host where you will install the WebCenter schema>` (for example, `localhost`)
   - **JDBC Port:** `<port>` (for example, `1521`)
   - **SID**: `<system identifier for the database with the same JDBC port>` (for example, `ORCL`)

   You must enter the **Connection Name** exactly as "WebCenter". There are cases when you may want to leverage an existing database connection for WebCenter services, and it may not be possible to change the database connection name to "WebCenter".

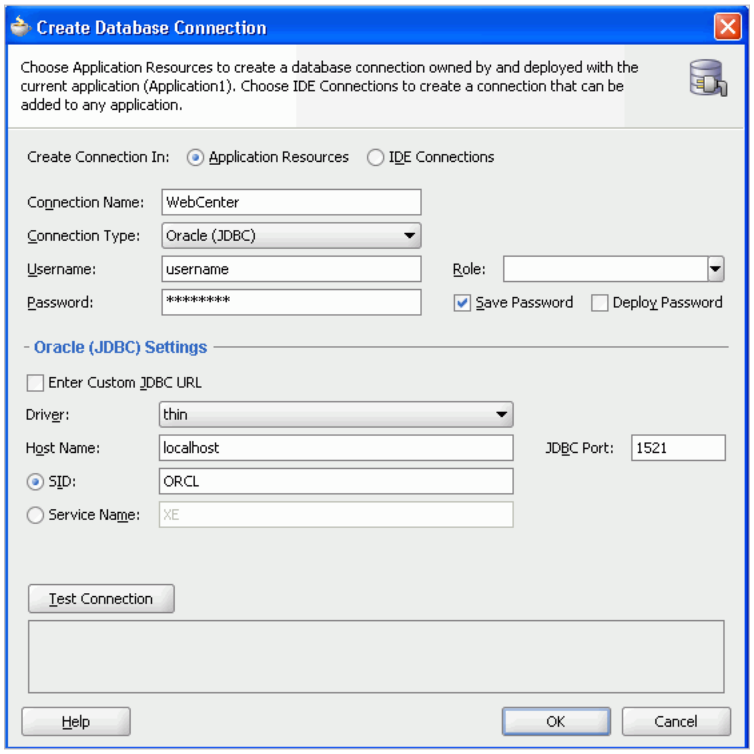
   To allow WebCenter services to use another database connection by a different name, you must add the following `<data-source>` tag as a child of the `<wpsC:adf-service-config>` element in the `adf-config.xml` file. (`adf-service-config` is a child of `adf-config`, and `data-source` is a child of `adf-service-config` or sibling of `extension-registry-config`.)

   For example:

```
<adf-config xmlns="http://xmlns.oracle.com/adf/config"
            xmlns:mdsC="http://xmlns.oracle.com/adf/mds/config"
            xmlns:wpsC="http://xmlns.oracle.com/webcenter/framework/service"
            xmlns:jndiC="http://xmlns.oracle.com/adf/jndi/config">

    <wpsC:adf-service-config
xmlns="http://xmlns.oracle.com/webcenter/framework/service">
        <data-source jndi-name="java:/comp/env/jdbc/NewDatabaseConnDS"/>
   </wpsC:adf-service-config>
</adf-config>
```

> **Note:** The syntax of the example name is `"java:/comp/env/jdbc/NewDatabaseConnDS"`. This is derived from the example name `NewDatabaseConn` you would have used to create the database connection in the creation wizard.

Figure 21–5 Database Connection



Create Database Connection

Choose Application Resources to create a database connection owned by and deployed with the current application (Application1). Choose IDE Connections to create a connection that can be added to any application.

Create Connection In: (● Application Resources) (IDE Connections)

Connection Name: WebCenter

Connection Type: Oracle (JDBC)

Username: username

Role:

Password: *********

[✓ Save Password] Deploy Password

Oracle (JDBC) Settings

Enter Custom JDBC URL

Driver: thin

Host Name: localhost

JDBC Port: 1521

(● SID:) ORCL

Service Name: XE

4. Click **OK**.

Note: While you can set up the connections to back-end servers at design time in Oracle JDeveloper, you can later add, delete, or modify connections in your deployed environment using Enterprise Manager Fusion Middleware Control. For more information, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

21.2.2 Adding the Tags Service at Design Time

To enable basic tagging, you must add either the **Tagging Button** component or the **Tagging Menu Item** component as well as the **Tagging - Dialog** task flow. In addition, if you want to tag *custom* objects, you need to register a resource viewer to visualize the tagged component when it is discovered by users.

This section describes how to enable tagging by adding the **Tagging Button** and the **Tagging - Dialog** task flow.

Note: To enable the ability to tag pages, you must manually add the Page service libraries to your project. For more information, see Section 21.1.2, "Requirements for the Tags Service."

21.2.2.1 Tags Service Components

The following JSF components are included in the service:

- **Tagging Button**: When you add a Tagging Button, a Tags link on the page enables you to invoke the Tag This Item (or Tag This Page) dialog at runtime.

- **Tagging Menu Item**: When you add a Tagging Menu Item, a menu option on the page enables you to invoke the Tag This Item (or Tag This Page) dialog at runtime.

21.2.2.2 Tags Service Task Flows

Table 21–1 lists the Tags service task flows.

Table 21–1 Tags Service Task Flows

Task Flow	Definition
Tagging - Dialog	This task flow displays a dialog that shows up to tag a particular object. This does not have any visible user interface rendering when dropped onto a page. It only becomes visible when the tagging button is clicked.
Tagging - Personal View	This task flow shows the tags created by the current user and objects tagged by those tags. Because this is a non-public task flow, it requires authentication.
Tagging - Related Links	For an object identified by a unique ID, this task flow lists all the other objects that are tagged with similar tags. The input parameters are `serviceId` and `resourceId`. The other objects listed include those that are similarly tagged; that is, their tag "sets" have an intersection of at least one. The list is ordered by the number of tags in the intersection.
Tagging - Tag Cloud	This task flow displays a tag cloud, which is a visual depiction of all the tags used. Tags are presented according to the frequency of their use. More frequently used tags display in bold fonts and varying font sizes—the larger the font, the more the tag has been applied. You can click a tag in the tag cloud to run a search that returns a list of all items that use the tag. Clicking a tag in the **Tagging - Tag Cloud** task flow sends the tag word as an event to all interested event consumers, such as the **Tagging - Tagged Items** task flow.
Tagging - Tagged Items	This task flow displays items or pages that have been tagged and listens for events sent by the **Tagging - Tag Cloud** task flow. This task flow is capable of refining itself to show only the items tagged with the tag passed to it.

21.2.2.3 How to Add the Tags Service to your Application

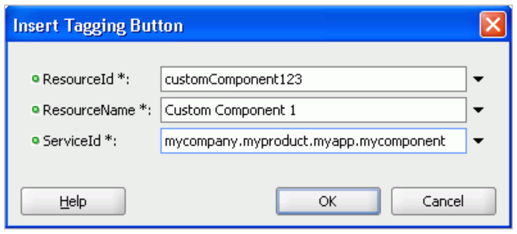
To add the Tags service to your WebCenter application:

1. Follow the steps described in Chapter 3, "Preparing Your Development Environment" to implement security and create a new customizable page in your application.
2. Ensure that you have set up a database connection to a database with the WebCenter schema installed.
3. Open the page where you included the search toolbar.
4. On the Component Palette (All Pages), select the **Tagging Button** and drop it on the page, inside a `PanelGroupLayout`. Make it the first child of the `PanelGroupLayout` by placing it to the left of the search toolbar. The Tagging Button enables users to start tagging.
5. Fill out the Insert Tagging Button dialog. Table 21–2 provides descriptions and example values for the fields in the dialog.

Table 21–2 Fields in the Insert Tagging Button Dialog

Field	Description
`ResourceId`	The ID that uniquely identifies the object to which you are binding the Tagging Button. The value in this field need not be static. For example, you could use EL to insert a unique value for each row in a table. For example: ■ To tag pages, enter `#{facesContext.viewRoot.viewId}` ■ To tag custom components, enter `customComponent123`
`ResourceName`	The name of the object to which you are binding the Tagging Button. The value in this field need not be static. For example, you could use EL to insert a unique value for each row in a table. ■ To tag pages, enter `#{facesContext.viewRoot.viewId}` ■ To tag custom components, enter `Custom Component 1`
`ServiceId`	An application-wide ID. (With custom components, you will add this to `service-definition.xml` when adding the resource viewer.) ■ To tag pages, enter `oracle.webcenter.page` ■ To tag custom components, enter `mycompany.myproduct.myapp.mycomponent`

Figure 21–6 Insert Tagging Button Dialog for a Custom Component



ResourceId *: customComponent123
ResourceName *: Custom Component 1
ServiceId *: mycompany.myproduct.myapp.mycomponent

6. Click **OK**.
7. From the Resource Palette, expand **WebCenter Services Catalog** and **Task Flows**.
8. Drag the **Tagging Dialog** from the Resource Palette and drop it on the page right after (that is, as a sibling to) the tagging button.
9. When prompted, select **Region** as the way to create the task flow.
10. For custom resources, follow the steps described in Section 11.2, "Extending Your Application with Custom Components" to register a resource viewer. For services exposing resources to be tagged (and therefore searched and viewed) WebCenter provides a Resource Action Handling framework. The Tags service uses this framework to allow acting on a search result.

In the Source view where you added the Tags service, you should see something similar to Example 21–1.

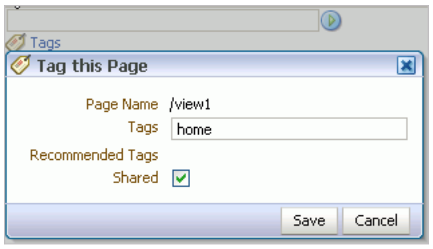
Example 21–1 Tagging Button and Tagging - Dialog Task Flow Source

```
<af:form id="f1">
        <af:region value="#{bindings.searchtoolbar1.regionModel}" id="r1"/>
        <tag:taggingButton resourceId="#{facesContext.viewRoot.viewId}"
                           serviceId="oracle.webcenter.page"
                           resourceName="#{facesContext.veiwRoot.viewId}"
                           id="tb1"/>
        <af:region value="#{bindings.tagginglaunchdialog1.regionModel}"
                   id="r2"/>
      </af:form>
```

Note: Even if there are multiple Tagging Buttons on the page, only one Tagging Dialog `af:region` is needed.

Run the page with the **Tagging Button** and **Tagging - Dialog** task flow. When the page renders in the browser, click the **Tags** link (Figure 21–7).

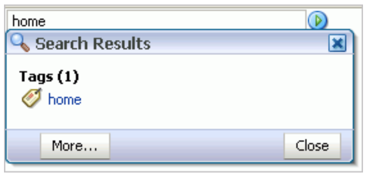
Figure 21–7 Tag This Page Dialog



Page Name /view1
Tags home
Recommended Tags
Shared ✓

11. When the Tag This Page dialog appears, enter the tag `home` and click **Save**. Note that every user of the system can perform this same operation and assign the same tag, which means that you get a weighted collection of tags.
12. In the search toolbar, type the search criteria `home` and click the **Search** icon.
13. You should `home` under Tags. (Figure 21–8)

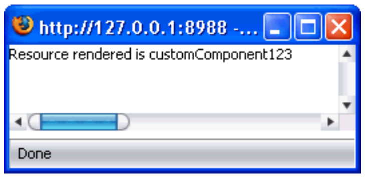
Figure 21–8 Search Results with Tagged Items and Tags



Tags (1)
home

14. For custom resources, you also would see `Custom Component 1` under Tagged Items. Click `Custom Component 1` and a dialog appears with the resource viewer you just created (Figure 21–9).

Figure 21–9 Resource Viewer



Resource rendered is customComponent123

21.2.3 Setting Security for the Tags Service

The Tags service does not require ADF security.

However, the **Tagging - Personal View** and **Tagging - Dialog** task flows do require authentication. Also, if the application is not secured, then pages are not returned in search results for tagged items.

When users are not authenticated, the user name associated with tags in the user's bookmarks will be "anonymous." This means that multiple users using the same unprotected custom WebCenter applications risk overwriting each other's tags.

For information about configuring ADF security, see Section 3.5, "Implementing Security in Your Application."

Note: You must register a resource viewer to allow custom objects to be found using Search or Tags. For information about how the resource viewer's `authorizerClass` determines whether users can view the resource, see Section 11.2, "Extending Your Application with Custom Components."

21.3 Advanced Information for the Tags Service

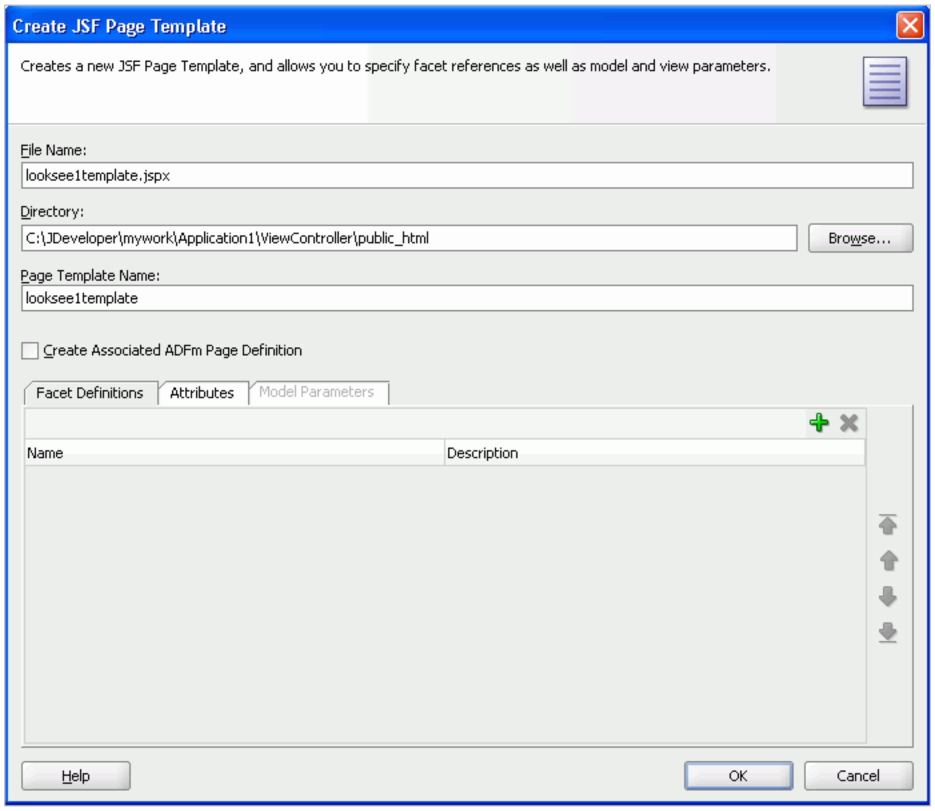
This section describes optional features available with this service.

21.3.1 Optional Way to Show Tags on Pages

While it is possible to add the Tagging Button and task flows on a page-by-page basis, typically you will want to add the search toolbar and tags as part of a page template that you can apply whenever building a page that requires it. To add tags through a page template:

1. Open the project where you plan to create pages with the Tags service.
2. Right-click the project name and select **New** from the context menu.
3. Under **Categories**, select **JSF** and, under **Items**, select **JSF Page Template**.
4. For **File Name**, enter `looksee1template`.

Figure 21–10 Create JSF Page Template Dialog


Create JSF Page Template

Creates a new JSF Page Template, and allows you to specify facet references as well as model and view parameters.

File Name: looksee1template.jspx

Directory: C:\JDeveloper\mywork\Application1\ViewController\public_html

Page Template Name: looksee1template

☐ Create Associated ADFm Page Definition

5. Click **OK**.
6. Go to the Source view and modify the code to be similar to that in Example 21–2.

Example 21–2 Sample JSF Page Template

```
<af:pageTemplateDef var="attrs">
  <af:xmlContent>
    <component xmlns="http://xmlns.oracle.com/adf/faces/rich/component">
      <display-name>looksee1template1</display-name>
      <facet>
        <facet-name>content</facet-name>
      </facet>
    </component>
  </af:xmlContent>
  <af:panelGroupLayout layout="vertical">
    <!-- Our toolbar -->
    <af:panelGroupLayout id="toolbar" layout="horizontal" halign="end"
                         inlineStyle="background-color:cyan;width:100%">
    </af:panelGroupLayout>
    <af:panelStretchLayout startWidth="300"
                           inlineStyle="height:600px; width:100%;">
      <f:facet name="start">
        <af:panelSplitter id="sidebar" orientation="vertical">
          <f:facet name="first">
            </f:facet>
          <f:facet name="second">
          </f:facet>
        </af:panelSplitter>
      </f:facet>
```

```
        <f:facet name="center">
          <af:facetRef facetName="content"/>
        </f:facet>
      </af:panelStretchLayout>
    </af:panelGroupLayout>
</af:pageTemplateDef>
```

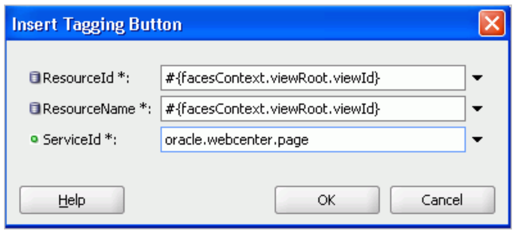
7. On the Component Palette, select the **Tagging Button** and drop it on the page, inside the `PanelGroupLayout`. Make it the first child of the `PanelGroupLayout`.
8. Fill out the Insert Tagging Button dialog as shown in Figure 21–11.

   Table 21–3 provides descriptions and example values for the fields in the dialog.

Table 21–3 Fields in the Insert Tagging Button Dialog for a Page

Field	Description
`ResourceId`	Is the unique identifier of your page. In this case, because the resource ID is meant for a page, enter `#{facesContext.viewRoot.viewId}`.
`ResourceName`	Is the name of the page. In this case, enter `#{facesContext.viewRoot.viewId}`. (You can enter a more user-friendly name.)
`ServiceId`	Is the service identifier that you will eventually add to `service-definition.xml` when adding the resource viewer. In this case, enter `oracle.webcenter.page`. Note: Because `oracle.webcenter.page` is already available in the Page service library, nothing will need to be added to `service-definition.xml`.

Figure 21–11 Insert Tagging Button Dialog for a Page



ResourceId *: #{facesContext.viewRoot.viewId}
ResourceName *: #{facesContext.viewRoot.viewId}
ServiceId *: oracle.webcenter.page

9. Click **OK**.
10. From the Resource Palette, expand the **WebCenter Services Catalog** and the **Task Flows**.
11. Drag **Tagging Dialog** from the Resource Palette and drop it on the page inside of the `panelGroupLayout`.
12. When prompted, select **Region** as the way to create the task flow.
13. Drag **Search Toolbar** and drop it after the **Tagging Dialog** you just added to the page template in the `panelGroupLayout`.
14. When prompted, select **Region** as the way to create the task flow.
15. Drag **Document Library - Recent Documents** and drop it in the second facet of the `panelSplitter` with identifier `sidebar`.

16. When prompted, select **Region** as the way to create the task flow.
17. In the Edit Task Flow Binding dialog, for `connectionName`, enter the name of the connection to your document library's content repository; for example, `${'MyFileSystem'}`. For information on how to set up a connection to your content repository, see Section 8.2, "Configuring Content Repository Connections."
18. The source for your page template should now look similar to Example 21–3.

Example 21–3 JSF Page Template with Tagging and Search

```
<af:pageTemplateDef var="attrs">
  <af:xmlContent>
    <component xmlns="http://xmlns.oracle.com/adf/faces/rich/component">
      <display-name>lookseeltemplate1</display-name>
      <facet>
       <facet-name>content</facet-name>
      </facet>
    </component>
  </af:xmlContent>
  <af:panelGroupLayout layout="vertical">
   <!-- Our toolbar -->
   <af:panelGroupLayout id="top" layout="horizontal" halign="end"
               inlineStyle="background-color:cyan;width:100%">
    <tag:taggingButton resourceId="#{facesContext.viewRoot.viewId}"
                     resourceName="#{facesContext.viewRoot.viewId}"
                        serviceId="oracle.webcenter.page"/>
     <af:region value="#{bindings.tagginglaunchdialog1.regionModel}"
                id="tagginglaunchdialog1"/>
     <af:region value="#{bindings.searchtoolbar1.regionModel}"
                id="searchtoolbar1"/>
   </af:panelGroupLayout>
    <af:panelStretchLayout startWidth="300"
                          inlineStyle="height:600px; width:100%;">
     <f:facet name="start">
     <af:panelSplitter orientation="vertical">
      <f:facet name="first">
       <af:region value="#{bindings.taskservicetaskflowdefinition1.regionModel}"
                  id="taskservicetaskflowdefinition1"
                  inlineStyle="position:absolute; left:0px; width:100%;
                   height:100%"/>
      </f:facet>
      <f:facet name="second">
       <af:region value="#{bindings.doclibrecentdocuments1.regionModel}"
                  id="doclibrecentdocuments1"/>
      </f:facet>
       </af:panelSplitter>
      </f:facet>
      <f:facet name="center">
       <af:facetRef facetName="content"/>
      </f:facet>
    </af:panelStretchLayout>
   </af:panelGroupLayout>
  </af:pageTemplateDef>
```

19. Create two new pages based upon the page template you just created. In the Create JSF Page dialog, select `lookseeltemplate1` from the **Use Page Template** list.
    - Create a page called `Search.jspx` and add the **Search** task flow in the center facet.

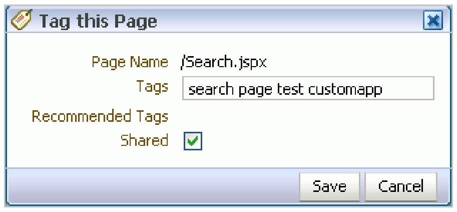
- Create a page called `Documents.jspx` and add the **Document Library - List View** task flow as the center facet. In the Edit Task Flow Binding dialog, for `connectionName`, enter the name of the connection to your document library's content repository; for example, `${'MyFileSystem'}`. For information on how to set up a connection to your content repository, see Section 8.2, "Configuring Content Repository Connections."

20. Save your pages.

To run your pages with tags:

1. In the Applications Navigator, run the `Search.jspx` page.
2. After you log in and the page comes up, click the **Tags** link.
3. In the Tag this Page dialog, enter `search page test customapp` for **Tags** (Figure 21–12).

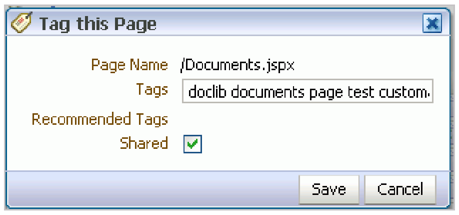
Figure 21–12 Tag this Page Dialog for Search Page



Page Name /Search.jspx
Tags search page test customapp
Recommended Tags
Shared ✓

4. Click **Save**.
5. Navigate to `Documents.jspx` (or run it separately).
6. Click the **Tags** link.
7. In the Tag this Page dialog, enter `doclib documents page test customapp` for **Tags** (Figure 21–13).

Figure 21–13 Tag this Page Dialog for Documents Page



Page Name /Documents.jspx
Tags doclib documents page test custom
Recommended Tags
Shared ✓

8. Click **Save**.
9. In the search toolbar for the `Documents.jspx` page, enter `search` and click the **Search** icon.
10. Under **Tags**, click `search`.

21.3.2 Using the Resource Action Handling Framework to Tag Custom Objects

For WebCenter Web 2.0 services exposing resources to be tagged (and therefore searched and viewed) WebCenter provides a Resource Action Handling framework. For example, rows in a table can be tagged.

To register a resource viewer, see Section 11.2, "Extending Your Application with Custom Components."

21.3.3 Using Tagged Cloud and Tagged Items Together

Using the events mechanism, you can wire together the **Tagging - Tag Cloud** and **Tagging - Tagged Items** task flows to show tagged items only for the selected tags in the tag clouds. The *producer* for the event is the tag cloud, and the *consumer* of the event is the tagged items.

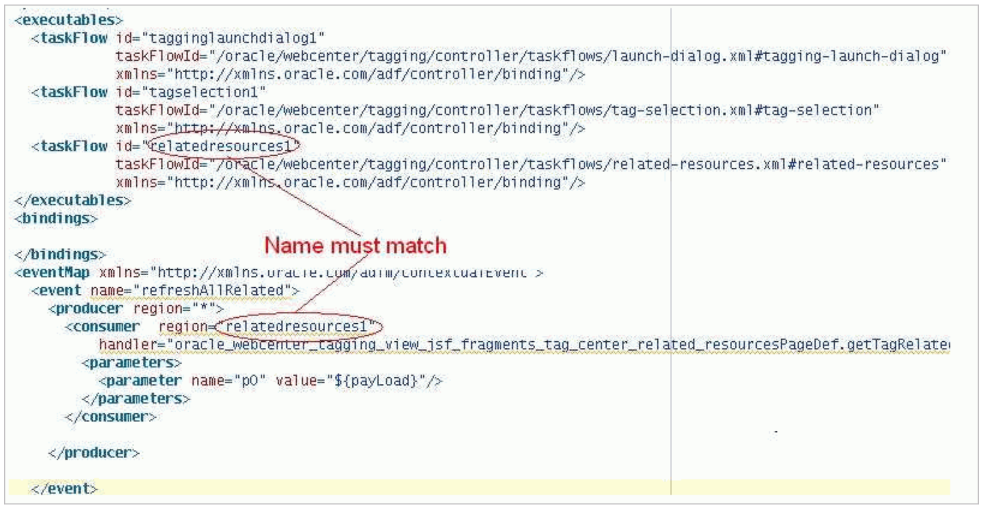
After adding the **Tagging - Tag Cloud** and **Tagging - Tagged Items** task flows to a page, there are two ways to wire them together.

The first, and easiest, way is to cut and paste the following snippet to the page definition, and then replace the region name `relatedresources1` with the task flow name used in your page definition.

```
<eventMap xmlns="http://xmlns.oracle.com/adfm/contextualEvent">
    <event name="refreshAllRelated">
      <producer region="*">
        <consumer region="relatedresources1"
                  handler="oracle_webcenter_tagging_view_jsf_fragments_tag_center_
related_resourcesPageDef.getTagRelatedResources">
          <parameters>
            <parameter name="p0" value="${payLoad}"/>
          </parameters>
        </consumer>
      </producer>
    </event>
    <event name="setShowSystemResources">
      <producer region="*">
        <consumer region="relatedresources1"
                  handler="oracle_webcenter_tagging_view_jsf_fragments_tag_center_
related_resourcesPageDef.setShowSystemResources"/>
      </producer>
    </event>
     <event name="setShowPersonalResources">
      <producer region="*">
        <consumer region="relatedresources1"
                  handler="oracle_webcenter_tagging_view_jsf_fragments_tag_center_
related_resourcesPageDef.setShowPersonalResources"/>
      </producer>
    </event>
  </eventMap>
```

Figure 21–14 shows an example of this.

Figure 21–14 Page Definition Replacement

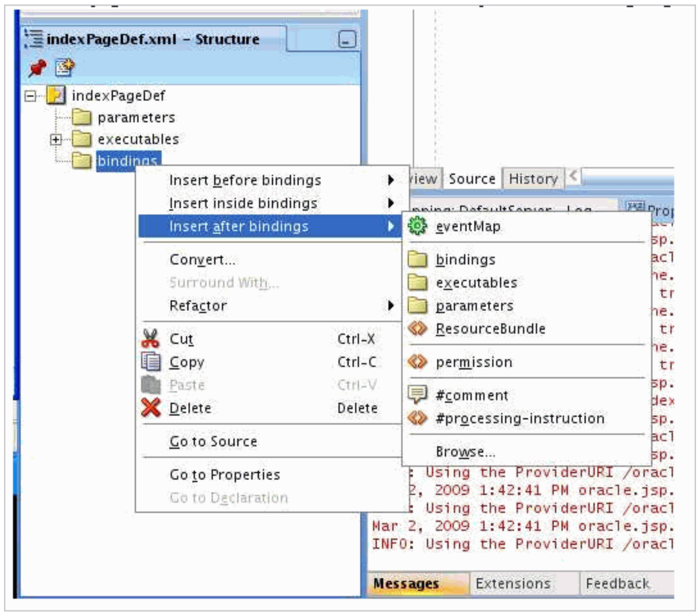


```
<executables>
  <taskFlow id="tagginglaunchdialog1"
            taskFlowId="/oracle/webcenter/tagging/controller/taskflows/launch-dialog.xml#tagging-launch-dialog"
            xmlns="http://xmlns.oracle.com/adf/controller/binding"/>
  <taskFlow id="tagselection1"
            taskFlowId="/oracle/webcenter/tagging/controller/taskflows/tag-selection.xml#tag-selection"
            xmlns="http://xmlns.oracle.com/adf/controller/binding"/>
  <taskFlow id="relatedresources1"
            taskFlowId="/oracle/webcenter/tagging/controller/taskflows/related-resources.xml#related-resources"
            xmlns="http://xmlns.oracle.com/adf/controller/binding"/>
</executables>
<bindings>

</bindings>
<eventMap xmlns="http://xmlns.oracle.com/adfm/contextualEvent">
  <event name="refreshAllRelated">
    <producer region="*">
      <consumer  region="relatedresources1"
          handler="oracle_webcenter_tagging_view_jsf_fragments_tag_center_related_resourcesPageDef.getTagRelate
        <parameters>
          <parameter name="p0" value="${payLoad}"/>
        </parameters>
      </consumer>

    </producer>

  </event>
```

The second way to wire the together the **Tagging - Tag Cloud** and **Tagging - Tagged Items** task flows is to follow these steps. The following example shows steps for adding the `refreshAllRelated` event.

1. Right-click your page, then choose **Go to Page Definition** from the context menu.
2. On the Structure pane, on lower left hand side, right-click **bindings**, and select **Insert after bindings** and then **eventMap**.

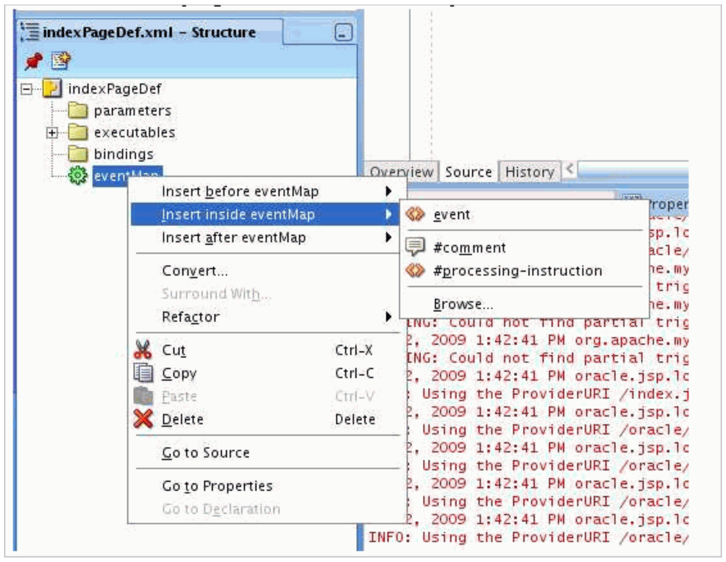
Figure 21–15 Page Definition - Structure Pane



```
: Using the ProviderURI /oracl
2, 2009 1:42:41 PM oracle.jsp.
: Using the ProviderURI /oracl
Mar 2, 2009 1:42:41 PM oracle.jsp.
INFO: Using the ProviderURI /oracl
```

3. Right-click the **eventMap**, and select **Insert inside eventMap** and then **event**.

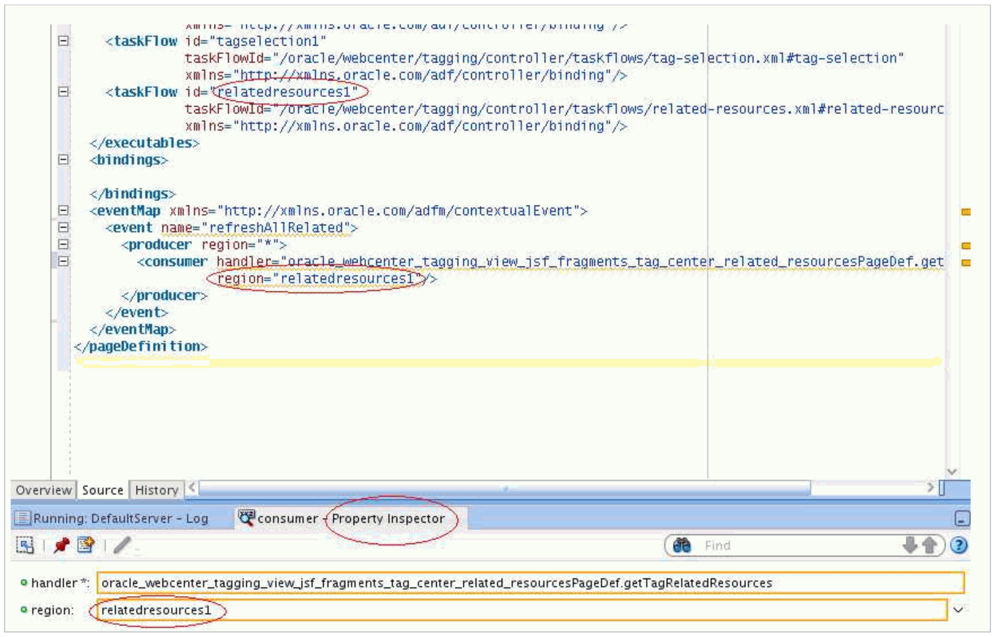
Figure 21–16 Page Definition - Event Map



```
ING: Could not find partial trig
2, 2009 1:42:41 PM org.apache.my
ING: Could not find partial trig
2, 2009 1:42:41 PM oracle.jsp.lc
: Using the ProviderURI /index.j
2, 2009 1:42:41 PM oracle.jsp.lc
: Using the ProviderURI /oracle/
2, 2009 1:42:41 PM oracle.jsp.lc
: Using the ProviderURI /oracle/
2, 2009 1:42:41 PM oracle.jsp.lc
: Using the ProviderURI /oracle/
2, 2009 1:42:41 PM oracle.jsp.lc
INFO: Using the ProviderURI /oracle/
```

4. In the dialog, enter the event name from the snippet above. For this example, enter `refreshAllRelated`.
5. Right-click **refreshAllRelated**, select **Insert inside refreshAllRelated** and then **Producer**.
6. When the dialog asks for the producer region, enter `*`.
7. Right-click the producer, select **Insert inside product**, and select **Consumer**.
8. When the dialog asks for the handler name, enter `oracle_webcenter_tagging_view_jsf_fragments_tag_center_related_resourcesPageDef.getTagRelatedResources`.
9. Because that dialog asks only for the consumer handler name and not the consumer region, you must add it manually. From the main window, select the consumer node. From the Property Inspector, you see a region field. Enter the related resources task flow name in your pageDef. By default, when you drag and drop the task flow to the page, Oracle JDeveloper will name it `relatedresources1`. If you later change the name, then make sure to use that new name instead of `relatedresources1`. (Figure 21–17)

Figure 21–17 Related Resources in the Page Definition



```
      <taskFlow id="tagselection1"
                taskFlowId="/oracle/webcenter/tagging/controller/taskflows/tag-selection.xml#tag-selection"
                xmlns="http://xmlns.oracle.com/adf/controller/binding"/>
      <taskFlow id="relatedresources1"
                taskFlowId="/oracle/webcenter/tagging/controller/taskflows/related-resources.xml#related-resourc
                xmlns="http://xmlns.oracle.com/adf/controller/binding"/>
    </executables>
    <bindings>

    </bindings>
    <eventMap xmlns="http://xmlns.oracle.com/adfm/contextualEvent">
      <event name="refreshAllRelated">
        <producer region="*">
          <consumer handler="oracle_webcenter_tagging_view_jsf_fragments_tag_center_related_resourcesPageDef.get
                    region="relatedresources1"/>
        </producer>
      </event>
    </eventMap>
  </pageDefinition>
```

handler *: oracle_webcenter_tagging_view_jsf_fragments_tag_center_related_resourcesPageDef.getTagRelatedResources

region: relatedresources1

10. Right-click the **consumer**, **Insert inside consumer**, and then **parameters**.

11. Right-click **parameters**, **Insert inside parameters**, and then **parameter**.

12. When the dialog asks for the parameter name, enter anything; for example, p0. For the value, enter `${payLoad}`.

Simply repeat the steps for the other events, substituting the corresponding event name and the consumer handler.

Note: The `setShowSystemResources` and `setShowPersonalResources` events do not take parameters, so step 9 to 11 are not needed.

Table 21–4 shows the information for the `refreshAllRelated` event.

Table 21–4 refreshAllRelated Event

Field	**Description**
Producer region	*
Consumer region	The related task flow name in your pageDef. By default, this is `relatedresources1`.
Consumer handler	`oracle_webcenter_tagging_view_jsf_fragments_tag_center_related_resourcesPageDef.getTagRelatedResources`
Parameter	This should be some meaningful name (p0 will work); value: `${payLoad}`

You can then click a tag in the tag cloud to see associated tagged items.

These are the only two task flows that talk to each other using events, but you can design your own task flows that listen to WebCenter events.

22 Integrating Oracle WebCenter Wiki and Blog Server

Oracle WebCenter Wiki and Blog Server includes features that enable you to incorporate wikis and blogs into an application or portal. This chapter explains how to integrate wiki and blog functionality into your applications at design time.

This chapter includes the following sections:

- Section 22.1, "Adding Wikis or Blogs to Your Application or Portal"
- Section 22.2, "Oracle WebCenter Wiki and Blog Server URL Endpoints and Query String Parameters"
- Section 22.3, "Oracle WebCenter Wiki and Blog Server Web Services Interface Interface"

22.1 Adding Wikis or Blogs to Your Application or Portal

You can add wikis or blogs to your application by using the following methods:

- Through the Web Clipping portlet or any portlet capable of consuming a URL
- Through an iFrame, which you include in a page in your custom WebCenter application
- Through the use of a custom-built user interface that you can create using the provided web services

Sample portlets that demonstrate some of these integration methods are located on the Oracle Technology Network at the following URL:

`http://webcenter.oracle.com`

From here, you can learn more about the sample portlets, as well as how to download and install them.

Note: You can use Oracle Access Manager-based single sign-on, Oracle Single Sign-On (OSSO), and SAML-based security to secure Oracle WebCenter Wiki and Blog Server. For information, see Section 12.7 "Setting Up the Wiki Service" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

The users you set up for your applications must match the user credentials on the Oracle WebCenter Wiki and Blog Server. Once a user is authenticated, if the user does not exist within the Oracle WebCenter Wiki and Blog Server, then the user is created and a default role is assigned to the user.

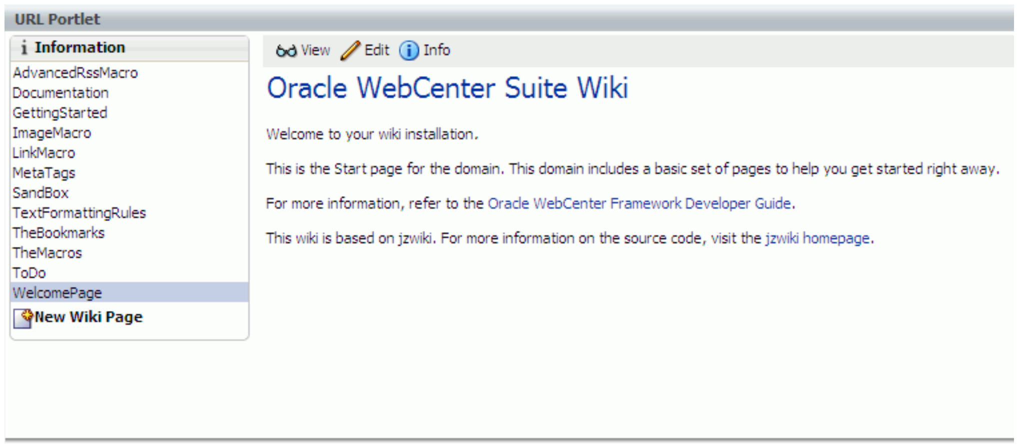
22.1.1 Adding Wikis or Blogs by Using a Portlet

You can bring a wiki into your application through the Web Clipping portlet or any portlet capable of consuming a URL. For examples of portlets capable of consuming a URL, see the Oracle WebCenter Suite page on OTN (`http://webcenter.oracle.com`).

Oracle WebCenter Framework includes a preconfigured Web Clipping portlet that you can register with your custom WebCenter application. Ensure that your application is secured and that the user credentials match on the Oracle WebCenter Wiki and Blog Server. You can learn more about this portlet in Chapter 32, "Creating Content-Based Portlets with Web Clipping."

Once you run your application to your browser, you can use the Web Clipping Studio to consume the desired URLs into your application. For information about the URL formats to use, see Section 22.2, "Oracle WebCenter Wiki and Blog Server URL Endpoints and Query String Parameters."

Figure 22–1 Sample Wiki Portlet in an Application



Oracle WebCenter Suite Wiki

Welcome to your wiki installation.

This is the Start page for the domain. This domain includes a basic set of pages to help you get started right away.

For more information, refer to the Oracle WebCenter Framework Developer Guide.

This wiki is based on jzwiki. For more information on the source code, visit the jzwiki homepage.

22.1.2 Adding Wikis or Blogs by Using an iFrame

You can add wikis and blogs to a page in your application with an iFrame. Simply reference the desired URL from within an iFrame, as shown in Example 22–1 and Figure 22–2. For information about URL formats to use, see Section 22.2, "Oracle WebCenter Wiki and Blog Server URL Endpoints and Query String Parameters."

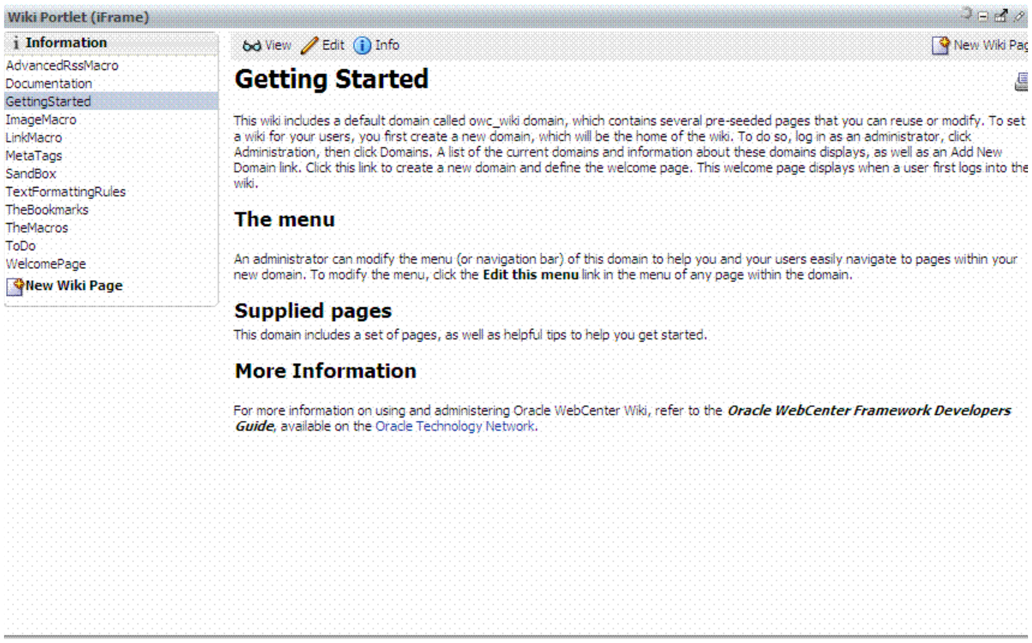
Example 22–1 Referencing a Wiki URL from an iFrame

```
<iframe src="http://server:port/owc_wiki/page/show.jz?inline=1&scope=domain"
width="100%"></iframe>
```

In this code, replace the server and port number with those of the Oracle WebCenter Wiki and Blog Server.

Figure 22–2 shows an example of how you can use an iFrame to integrate wiki. This example shows this being done using a portlet.

Figure 22–2 Sample Wiki Portlet in an iFrame at Runtime



Wiki Portlet (iFrame)

Information

- AdvancedRssMacro
- Documentation
- GettingStarted
- ImageMacro
- LinkMacro
- MetaTags
- SandBox
- TextFormattingRules
- TheBookmarks
- TheMacros
- ToDo
- WelcomePage
- **New Wiki Page**

View Edit Info New Wiki Pag

Getting Started

This wiki includes a default domain called owc_wiki domain, which contains several pre-seeded pages that you can reuse or modify. To set a wiki for your users, you first create a new domain, which will be the home of the wiki. To do so, log in as an administrator, click Administration, then click Domains. A list of the current domains and information about these domains displays, as well as an Add New Domain link. Click this link to create a new domain and define the welcome page. This welcome page displays when a user first logs into the wiki.

The menu

An administrator can modify the menu (or navigation bar) of this domain to help you and your users easily navigate to pages within your new domain. To modify the menu, click the **Edit this menu** link in the menu of any page within the domain.

Supplied pages

This domain includes a set of pages, as well as helpful tips to help you get started.

More Information

For more information on using and administering Oracle WebCenter Wiki, refer to the ***Oracle WebCenter Framework Developers Guide***, available on the Oracle Technology Network.

22.1.3 Adding Wikis or Blogs by Using Web Services

Oracle WebCenter Wiki and Blog Server provides web services that enable interaction between your application and the wiki. You can add wikis and blogs to your applications by calling web services along with a custom-built user interface. This requires you to call the web services directly and create the user interface for interactions with the WebCenter Wiki and Blog Server. Alternatively, you can just redirect to the prebuilt wiki and blog pages.

For information about how to use web services, see Section 22.3, "Oracle WebCenter Wiki and Blog Server Web Services Interface Interface."

22.2 Oracle WebCenter Wiki and Blog Server URL Endpoints and Query String Parameters

When you reach the point where you enter the wiki or blog source in the URL-consuming portlet or in the iFrame, use the formats provided in Table 22–1.

If you want to add a hyperlink that references a specific wiki page or a blog, on one of your application pages, use the URL formats provided in Table 22–2.

You can use wiki and blog query string parameters to define context (within the application context or external to it), and look and feel (page background colors and fonts). Query string parameters are bits of information you add to a URL to refine the behavior of the URL target. Table 22–3 lists and describes query string parameters you can use in custom WebCenter application wiki and blog URLs.

Note: By default, Oracle WebCenter and Oracle WebCenter Wiki and Blog Server do not share cookies. To enable this feature, you can update the settings in the `weblogic.xml` file of your custom WebCenter application. For information, see Section 12.7 "Setting Up the Wiki Service" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

For more information about using these URLs and query string parameters in your custom WebCenter application, see Section 22.1, "Adding Wikis or Blogs to Your Application or Portal."

Table 22–1 URL Formats for Exposing Wikis and Blogs in Applications

Type	URL Format
Wiki	`http://`*`server:port`*`/owc_wiki/page/show.jz?inline=1&scope=`*`domain`*
General Blog	`http://`*`server:port`*`/owc_wiki/blog/list.jz?inline=1&name=`*`domain`*
Personal Blog	`http://`*`server:port`*`/owc_wiki/blog/list.jz?inline=1&name=`*`user`*

Table 22–2 Formats for Hyperlinks to Wikis and Blogs

Target	URL Format
Wiki	`http://`*`server:port`*`/owc_wiki/page/show.jz?inline=1&page=`*`domain`*`:`*`wikiPageName`*
General Blog	`http://`*`server:port`*`/owc_wiki/blog/list.jz?inline=1&name=`*`domain`*
Personal Blog	`http://`*`server:port`*`/owc_wiki/blog/list.jz?inline=1&name=`*`user`*

Table 22–3 Query String Parameters to Use in Application Wiki and Blog URLs

Parameter	Description
`inline`	■ Value of `0` sets the view to normal, displaying the default user interface and features of the wiki and blog server. This is the recommended mode for wiki administrators. ■ Value of `1` strips away nonessential wiki and blog chrome. It also renders left-side navigation that lists all wiki pages within the current domain. This is the recommended mode when integrating with an application or portal. ■ Value of `2` is similar to `inline=1`, except it turns off left-side navigation. For information on how a wiki page looks on Oracle WebCenter Wiki and Blog Server when rendered in different `inline` modes, see Chapter 28, Working with Wikis and Blogs in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.
`name`	Facilitates navigation to a specific blog, attributed either to a particular domain or user.
`page`	Facilitates navigation to a specific page in a specified domain. This variable follows the syntax: `page=`*`domain`*`:`*`wikiPageName`*. In general, you should include both the domain and the page name to reduce the potential for any ambiguity.

Table 22–3 (Cont.) Query String Parameters to Use in Application Wiki and Blog URLs

Parameter	Description
`scope`	Navigates to the start page within the specified domain. If the specified domain does not exist, then it automatically creates a new domain with the name specified for the scope variable. If the scope variable creates the domain on the fly, it also creates the home page (Welcome Page) and redirects the user to that page.
`theme`	Dynamically applies the specified wiki theme to the requested page (for usage other than with iFrames, the theme must already exist on both the Oracle WebCenter Wiki and Blog Server and the application server). ■ `default` applies the default theme specified on the server. ■ `theme_name` applies the specified wiki theme/CSS to the wiki page and all its children. Oracle WebCenter Wiki and Blog Server provides seeded wiki themes, which you can include in your wiki and blog URLs. Use any one of the following (use the value in parenthesis in the session variable): ■ Deep Sea (`deepsea`) ■ Sand (`sand`) ■ WebCenter (`webcenter`) ■ Wiki Default (`default`) ■ Red (`red`) ■ Blue (`blue`) ■ Tech Gray (`tech_gray`) ■ Bighorn (`bighorn`) ■ Midnight (`onyx`) ■ Flatirons (`flatirons`) ■ Blue Sky (`bluesky`) ■ Dew (`olive`) ■ Dusk (`monochrome`) ■ Mist (`white`) ■ Storm (`storm`) If you create your own themes, then for your application chrome to render consistently, you must ensure that the CSS files you use reside on both the server containing your Oracle WebCenter installation and the Oracle WebCenter Wiki and Blog Server.
`wcURL`	Allows you to specify an encoded URL to which a domain and page are appended when the wiki and blog server renders links to wiki pages. This applies only when `inline=1`. This parameter is useful for integrating wiki pages into a portal when you want wiki links to navigate to a portal location with an embedded wiki page rather than directly to the page on the wiki and blog server.

All of these URL formats can take query string parameters to focus the result provided on the target page. Parameters, such as `inline` and `theme`, are session-level variables. Once the URL passes session-level variables to the target, the variable values continue to apply, even if you leave and return to the original target page. Also, session-level variables are applied even when the parameter is not included in subsequent URLs.

22.3 Oracle WebCenter Wiki and Blog Server Web Services Interface Interface

Once you have installed the WebCenter Wiki and Blog Server, you can access the Oracle WebCenter Wiki and Blog Server Web Services end point with the following URL:

```
http://host:port/owc_wiki/services/WikiRemoteService
```

Note: In this URL, the host and port information refers to the computer where you installed your Oracle WebCenter Wiki and Blog server.

Oracle Wiki and Blog Server Web Services provide access to obtain information and content from wiki pages and domains. It also enables the creation, modification, and removal of wiki pages and domains. You can use Oracle JDeveloper (or other tools) to create a proxy for the Web Services from the WSDL definition, located here:

```
http://host:port/owc_wiki/services/WikiRemoteService?WSDL
```

This section describes the Oracle Wiki and Blog Server Web Services interface. For information about the Web services security, refer to Section 22.3.2, "Oracle WebCenter Wiki and Blog Server Web Services Security."

22.3.1 Definition of the Interface

Some of the methods return information in JavaBeans. Table 22–4 shows the attributes of the `DomainInfo` and `PageInfo` beans. You can also use the getter methods of the described attributes, for example, `long getCreated()`.

Table 22–4 Oracle WebCenter Wiki and Blog Server Web Services Data Structures

`DomainInfo` Bean	`PageInfo` Bean
`String domain;`	`String domain;`
`String description;`	`String name;`
`String author;`	`int revision;`
`long created;`	`int views;`
`String startPage;`	`String author;`
	`long created;`
	`String editor;`
	`long modified;`
	`String viewURL;`
	`String editURL;`

The Web Services methods include methods for accessing and performing actions on the domains and pages in the wiki, the blogs, and blog entries, as described in the following tables.

Table 22–5 Domain-Related Methods

Return Type	Method	Description
`DomainInfo[]`	`getAllDomainInfo(int maxResult, int offset, String key)`	Returns a list of all domains.
`DomainInfo`	`getDomainInfo(String domainName, String key)`	Returns information about the specified domain.
`void`	`createDomain (String domainName, String description, String startPage, String key)`	Creates a domain with the specified attributes.
`void`	`deleteDomain(String domainName, String key)`	Deletes the specified domain.
`PageInfo[]`	`getAllPageInfo (String domainName, int maxResult, int offset, String key)`	Returns a list of all pages within a domain.
`editDomainInfo`	`editDomainInfo (String domainName, String description, String startPage, String key)`	Modifies the specified domain with the specified attributes.

Table 22–6 Page-Related Methods

Return Type	Method	Description
`PageInfo`	`getPageInfo (String domainName, String pageName, String key)`	Returns information about the specified wiki page.
`void`	`createPage (String domainName, String pageName, PageEditMode mode, PageType type, String key)`	Creates a wiki page with the specified attributes. This method supports different editing modes. These are `editable by everyone`, `restricted to logged in users`, `restricted to members of the domain`[1], and `only admins are allowed to edit.`[2] The method also supports two page types: `HTML` and `wiki markup`.
`void`	`deletePage (String domainName, String pageName, String key)`	Deletes the specified wiki page.
`String`	`getPlainPage (String domainName, String pageName, String key)`	Returns the page content in the wiki markup format.
`String`	`getRenderedPage (String domainName, String pageName, String key)`	Returns the content rendered to HTML.
`void`	`savePage (String domainName, String pageName, String content, String key)`	The String content parameter takes text in the format that you specified in the `type` parameter while creating the wiki page.

[1] If the wiki administrator has specified a list of members in **Domains > Manage Members**, then "restricted to members of the domain" displays. The domain membership is otherwise open to all wiki users.

[2] This option only displays if the currently authenticated user is an administrator of the wiki.

Table 22–7 Administrative Methods

Return Type	Method	Description
`String key`	`login (String username, String passcode)`	Logs in a user and obtains the key that is used on all subsequent web service methods. The web service login expires after 30 minutes.
`null`	`logout (String key)`	Logs out the current user as defined by the passed key.

Table 22–8 Blog-Related Methods

Return Type	Method	Description
`boolean`	`enablePersonalBlog (String description, boolean enable, String key)`	Creates a personal blog for the user associated with the passed key (if it does not exist already). If the personal blog already exists, then this method does nothing.
`String[]`	`getListofBlogs (BlogType type, String keyword, int maxResults, int offset, String key)`	Returns a list of all blogs on the server (if `BlogType` is set to `all`), all personal blogs (if `BlogType` is set to `user`), or all blogs associated with domains (if `BlogType` is set to `domain`).

Table 22–9 Blog Entry-Related Methods

Return Type	Method	Description
`String[]`	`getAllBlogEntries (String blogName, String key)`	Returns a list of blog entries within the specified blog.
`null`	`createBlogEntry (String blogName, String title, String content, String key)`	Creates a blog entry in the specified blog with the specified title and content. Can only be performed by the blog author, those identified as additional blog authors (see *Oracle Fusion Middleware User's Guide for Oracle WebCenter* for more information), and the wiki administrator. If the blog is a domain blog, then only the wiki administrator can perform this task.
`null`	`editBlogEntry (String blogEntryID, String title, String content, String key)`	Modifies a specified blog entry with the specified title and content. Can only be performed by the blog author, those identified as additional blog authors (see *Oracle Fusion Middleware User's Guide for Oracle WebCenter* for more information), and the wiki administrator. If the blog is a domain blog, then only the wiki administrator can perform this task.
`null`	`deleteBlogEntry (String blogEntryID, String key)`	Deletes the specified blog entry. Can only be performed by the blog author, those identified as additional blog authors (see *Oracle Fusion Middleware User's Guide for Oracle WebCenter* for more information), and the wiki administrator. If the blog is a domain blog, then only the wiki administrator can perform this task.

Table 22–9 (Cont.) Blog Entry-Related Methods

Return Type	Method	Description
`String`	`getBlogEntry (String blogEntryID, String key)`	Returns the content for the specified blog entry.

Table 22–10 Blog Comment-Related Methods

Return Type	Method	Description
`String[]`	`getAllBlogEntryComments (String blogEntryID, String key)`	Returns all comments on a blog entry.
`void`	`createBlogEntryComment (String blogEntryID, String text, String key)`	Creates a comment on the blog entry.

Table 22–11 Search-Related Methods

Return Type	Method	Description
`SearchResult`	`SearchResult[] search (String searchText, SearchType type, int maxResults, String key)`	Returns URLs to wikis and blog objects that contain the specified keywords (in `searchText`). You can constrain the size of the results, if desired. Valid values for `SearchType` are `all`, `wiki`, or `blog`, which specifies the type of content to search.

The APIs: `getAllDomainInfo()`,`getAllPageInfo()`,`search()`, and `getListOfBlogs()` support pagination. These APIs support the notion of "block fetch," where the clients can specify the maximum number and block of results that should be returned.

The`getAllDomainInfo()`,`getAllPageInfo()`, and `getListOfBlogs()` APIs support two parameters, `maxResults` and `offset`from which the desired (maximum) number of results are returned. However, the calling client must maintain the state of `offset` or the cursor. Clients should choose and provide a fixed value for `maxResults` in repeated calls to the API to get the correct result.

For example, setting `maxResults = 10`, and `offset` set to `1`, the first call returns the first 10 rows (if present). Subsequent calls increment the offset parameter (being maintained at the client side) with `maxResults` fixed at `10`. The subsequent calls return rows from `11-20`, `21-30` and so on until it returns all rows.

A value of `<= 0` for `maxResults` returns the full result set in one call, without pagination. Also, `offset <= 0` returns the full result set in one call (without pagination), starting from the first row.

The `search()` method supports the `maxResults` parameter, which you can use to restrict the number of rows returned.

22.3.2 Oracle WebCenter Wiki and Blog Server Web Services Security

All Oracle WebCenter Wiki and Blog Server Web Services methods are protected to prevent unauthorized access. Every method contains a String key parameter to ensure authorized access. This key is generated as a function of a user's name and a preconfigured passcode. The passcode is an arbitrary string that the administrator sets up in the Oracle WebCenter Wiki and Blog Server application after installation. As the Wiki and Blog Server developer, you need to obtain this passcode to use the Web Services interface to access the wiki. For information about the passcode, see Section 12.7 "Setting Up the Wiki Service" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*

To create the key, use the following method:

```
String key = client.login(username, passcode);
```

Note: In this method, the `username` refers to the name of the user on whose behalf the Web Services is making the call (for example, the user logged into your custom WebCenter application who is accessing the wiki by means of a portlet) and `passcode` refers to the parameter key that you configured, as described in Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter.

22.3.3 Example Java program

The following code is an example of a Java program that accesses the Web Services interface. This program lists all the page names in the Training domain.

```
package oracle.webcenter.wiki.ws.test;
import  oracle.webcenter.wiki.ws.*;
import  oracle.webcenter.wiki.security.*;

public class ListPages
{
```

In this example, because we have deployed the wiki to a server, using port `6688`, we code the Web Services end point to this URL. You can, however, parameterize the end point.

```
  private static final String endpoint =
    "http://localhost:6688/owc_wiki/services/WikiRemoteService";
```

Each Web Services method must authenticate the caller to the Web Services. Authentication consists of a user name and a preconfigured passcode. In this example, we code these values.

```
  private static final String username = "jsmith";
  private static final String passcode = "passcode";
```

We code the domain name in this example.

```
  private static final String domain = "Training";

  public static void main(String[] args) throws Exception
    {
      try
      {
```

Next, we create a client-side proxy to access the Web Services.

```
      WikiRemoteServiceClient client =
          new WikiRemoteServiceClient();
```

We then set the end point of the proxy to the actual location where we deployed the Web Services.

```
      client.setEndpoint(endpoint);
```

Each Web Services method must pass a security key to authenticate the user and call the method. We can generate this key by calling the login web service using the user's name and the Web Services-configured passcode.

```
       String key = client.login(username, passcode);
```

Using the Web Services proxy, we fetch into an array the information about all the pages in the selected domain: If there is no such domain, then the program throws an exception. If the domain does not contain any pages, then the program returns an empty array.

```
      PageInfo[] pages = client.getAllPageInfo(domain, key);
      System.out.println("Pages in " + domain + " domain:");
```

Throughout the pages array, print the name of each page using the getter method `getName()`.

```
      for (int i = 0; i < pages.length; i++)
      {
        System.out.println("  " + pages[i].getName());
      }
    }
  }
```

If there is an exception, then the program captures the error and prints it.

```
    catch (Exception e)
    {
      System.out.println("Exception: " + e);
    }
  }
}
```

For more examples of portlets capable of consuming a URL, see the Oracle WebCenter Suite page on the Oracle Technology Network (`http://webcenter.oracle.com`).

23 Integrating the Worklist Service

This chapter describes how to integrate the Worklist service into your application. Worklists enable users to view and take action on all tasks and notifications from a BPEL (Business Process Execution Language) server all in one place.

This chapter includes the following sections:

- Section 23.1, "Introduction to the Worklist Service"
- Section 23.2, "Basic Configuration for the Worklist Service"
- Section 23.3, "Advanced Information for the Worklist Service"

23.1 Introduction to the Worklist Service

The Worklist service enables you to show Business Process Execution Language (BPEL) Worklist items assigned to the currently authenticated user. The BPEL Worklist items are open BPEL tasks from one or more BPEL Worklist Repositories to which your application is connected.The Worklist displays BPEL Worklist items that are a result of a task invoked as part of a BPEL Workflow process, or are a result of a message being sent to the Worklist channel on the Oracle User Messaging Service.

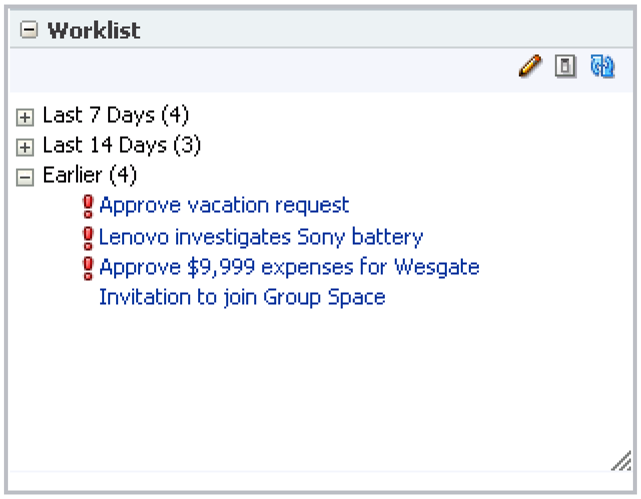
23.1.1 Understanding Worklists

The Worklist service provides a personal, at-a-glance view of business processes that require your users' attention. These can include a request for document review or other types of business processes that come directly from your enterprise applications.

Worklist items come from a variety of sources. Some Worklist items are kicked off by events that are associated with an externally defined workflow. A workflow maps the route an item follows once an event kicks off. This type of workflow is defined in a Worklist BPEL server, such as Oracle BPM Worklist.

Messages, alerts, and notifications might also come from the SPD. The Worklist task flow includes a control for accessing messaging preferences on this server. Use these controls to specify the channels over which to receive Oracle User Messaging Service messages and to define messaging filters.

Figure 23–1 Worklist Service at Runtime


Worklist
Last 7 Days (4)
Last 14 Days (3)
Earlier (4)
Approve vacation request
Lenovo investigates Sony battery
Approve $9,999 expenses for Wesgate
Invitation to join Group Space

See Also: *Oracle Fusion Middleware User's Guide for Oracle WebCenter* for more information about the service at runtime.

23.1.2 Requirements for the Worklist Service

To use the Worklist service, you must have a BPEL server installed. For information on installing BPEL for the Worklist service, see Section 4.4 "Back-End Requirements for the Worklist Service and WebCenter Spaces Workflows" in the *Oracle Fusion Middleware Installation Guide for Oracle WebCenter*.

For the Worklist to properly function, it must reside on a secured page, as the service uses the currently authenticated user to access the BPEL server. If the page or application that contains the Worklist service is not secured, you will encounter exception errors.

23.2 Basic Configuration for the Worklist Service

This section describes how to set up a connection for the Worklist service and add the task flow to your custom WebCenter application.

23.2.1 Setting up Connections for the Worklist Service

To use the Worklist service, you must establish a connection to one or more BPEL servers. This section describes these connections and how to create them in a custom WebCenter application.

23.2.1.1 Worklist Service Connections

The Worklist service relies on a BPEL server. To use this server, you must create a connection to it. Once you create a connection using the Create Worklist Connection dialog box, the connection is registered in your application's `connections.xml` and referenced in your application's `adf-config.xml`. You can see the connections to your BPEL Worklist Repositories in the Application Resources pane.

Note: While you can set up the connections to back-end servers at design time in Oracle JDeveloper, you can later add, delete, or modify connections in your deployed environment using Enterprise Manager Fusion Middleware Control. For more information, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

23.2.1.2 How to Set Up Connections for the Worklist Service

The Worklist service provides a task flow which you can add to your application at design time or at runtime. Before you can run the task flow, you create a connection from your application to a BPEL server.

Note: The application you use must be prepared for service consumption. For information on doing so, see Chapter 11, "Preparing Your Application for Oracle WebCenter Web 2.0 Services."

To set up the connection for the Worklist service:

1. In the Application Resources panel for your application, right-click **Connections**, then choose **New Connection> Worklist**.
2. In the Create Worklist Connection dialog box, choose whether to create the connection in the Application Resources (only the current application can use this connection) or IDE Connections (other applications you create can use the same connection).
3. In the Connection Name field, enter a name for your connection. This name will be used as the display name of the Worklist server.

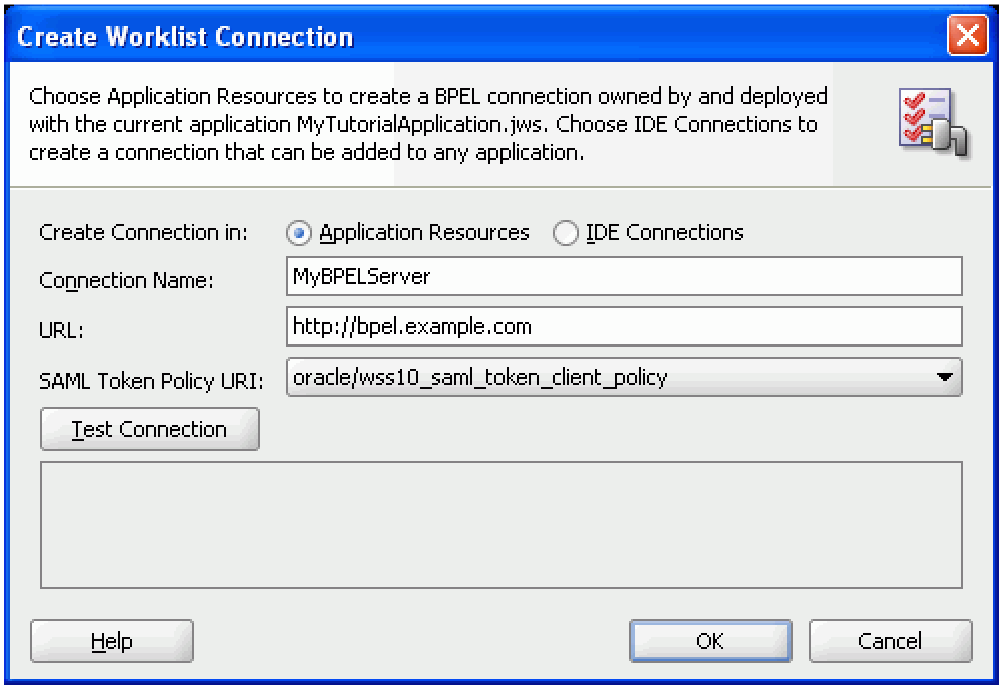
   Note: At runtime, when users view their worklist items, they can choose to group the items by "Worklist Server." The Worklist Server name used is the name you enter in the Connection Name field.

4. In the URL field, enter the location of your BPEL/SOA server, such as `http://bpel.example.com`. This is the URL for the managed server running the SOA processes.
5. Select the SAML Token Policy URI to use for this server.

   SAML (Security Assertion Markup Language) is an XML-based standard for passing security tokens defining authentication and authorization rights. An attesting entity (that already has trust relationship with the receiver) vouches for the verification of the subject by method called sender-vouches. Options available are:

   - SAML Token Client Policy (**oracle/wss10_saml_token_client_policy**) - Select to verify your basic configuration without any additional security. This is the default setting.
   - SAML Token With Message Client Policy (**oracle/wss10_saml_token_with_message_protection_client_policy**) - Select to increase the security using SAML-based BPEL Web Services. If selected, you must configure keys stores both in your custom WebCenter application and in the BPEL application. To learn more about this configuration, refer to Chapter 9 "Configuring Policies" in *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

Figure 23–2 Create Worklist Connection Dialog Box


Create Worklist Connection
Choose Application Resources to create a BPEL connection owned by and deployed with the current application MyTutorialApplication.jws. Choose IDE Connections to create a connection that can be added to any application.
Create Connection in:
Application Resources
IDE Connections
Connection Name:
URL:
SAML Token Policy URI:
Test Connection
Help
OK
Cancel

Create Connection in: [✓ Application Resources] IDE Connections
Connection Name: MyBPELServer
URL: http://bpel.example.com
SAML Token Policy URI: oracle/wss10_saml_token_client_policy

6. Click **Test Connection** to validate the URL
7. Click **OK**.

23.2.2 Adding the Worklist Service at Design Time

This section describes the Worklist task flow and how to add it to your application.

23.2.2.1 Worklist Service Task Flows

There is one task flow associated with the Worklist service, called "Worklist." You can find this task flow in the WebCenter Services Catalog in the Resource Palette.

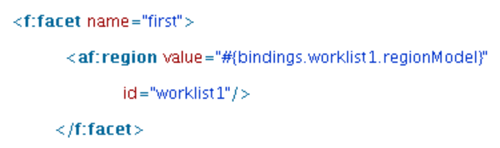
23.2.2.2 How to Add the Worklist Service to your Application

You can add the Worklist service to your application before or after you've created your connection. However, you must set up your connection before you run the service.

To add the Worklist task flow to your application:

1. Follow the steps in Section 11.1.1, "How to Prepare Your Application to Consume Services" to implement security and create a customizable page in your application.
2. Open the page where you wish to add the Worklist task flow.
3. In the Resource Palette, in the My Catalogs pane, open the **WebCenter Services Catalog**, then open the **Task Flows** folder.
4. Drag the worklist task flow onto the desired region on the page, then choose **Region** from the pop-up list. The task flow displays on the page. For example, in the Source view, you will see Figure 23–3.

Figure 23–3 Worklist Task Flow in the Page Source



```
<f:facet name="first">
    <af:region value="#{bindings.worklist1.regionModel}"
        id="worklist1"/>
  </f:facet>
```

5. Save your project, then run your page to a browser. The Worklist service displays in your browser.

23.2.3 Setting Security for the Worklist Service

The Worklist service displays the tasks for the currently authenticated user. So, for users to store and retrieve Worklist tasks on the BPEL server from your application, you must set up security on the application (as described in Section 11.1.1.1, "Implementing Security for Services"). The usernames need to either exist in a shared user directory (LDAP), or set up similarly (same username) on both the custom WebCenter application and the BPEL Server. For example, if the user `rsmith` needs to use the Worklist service in your custom WebCenter application to store and retrieve tasks from the BPEL server, you must ensure that the user `rsmith` exists on both the BPEL server and within your application.

To enable users to perform actions on assigned Worklist tasks, you can configure SSO If you do not configure SSO, the users will be prompted to log into the Worklist Task Details page on the BPEL server. For more information, refer to the Oracle Fusion Middleware Security Guide.

Only authenticated users can view Worklist tasks on the BPEL server to which they have permissions to view. If the application end user does not supply appropriate credentials, or if the page containing the Worklist service is not secured, she will not see any Worklist items stored on the BPEL server and exception errors may occur.

23.3 Advanced Information for the Worklist Service

This section describes additional configuration information and other optional servers you can use with the Worklist service.

23.3.1 Adding or Modifying Connections

The Worklist service collates all the Worklist items for the authenticated user from all connections in your application defined for the Worklist service. You can use the create connection method described in Section 23.2.1.2, "How to Set Up Connections for the Worklist Service" to create additional connections, or right-click the connection name in the Application Resources list and choose **Delete** to remove a connection.

Part IV

Completing Your WebCenter Application

Part IV contains the following chapters:

24 Securing Your WebCenter Application

This chapter describes how to use Oracle ADF Security in your WebCenter application to handle authentication and authorization.

This chapter includes the following sections:

24.1 Introduction to WebCenter Application Security

WebCenter applications are dynamic and often involve input from users in the form of customization and personalization, and require a flexible security model. Consequently, the WebCenter security model is based on the ADF security model rather than the more traditional J2EE security model. ADF Security is a framework built upon the Java Authentication and Authorization Service (JAAS) standard. See *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework* for a complete introduction to ADF Security.

24.2 Working with External Applications

Oracle WebCenter Framework defines an external application as any application that implements its own authentication process. That is, an application that does not take part in the WebCenter Framework application's single sign-on process. In some cases, the identity management solution may be the same, but the authentication process can be different.

When WebCenter Framework Service interacts with an application that handles its own authentication, you can associate that service with an external application to allow for credential provisioning. Therefore, the use of an external application definition provides a means of accessing content from these independently authenticated applications.

In order to replicate a single sign-on experience from the end user's perspective, the external application service captures the username and password, and any other credentials for the external application, and supplies it to the WebCenter service requiring it. The WebCenter service then uses this and logs in on behalf of the end user. This username and password combination is securely stored in a credential store configured for the WebLogic domain where the application is deployed.

The user provides login credentials when prompted, and these credentials are mapped to the WebCenter application user and stored in the credential store configured for the domain. The credential store subsequently supplies that information during authentication to the external application. Unless the external application's credentials change, the user supplies the credentials only once as the mapped information is read from the credential store for future requests.

The external applications that are to be used by the custom WebCenter application can be specified prior to deployment through a wizard in Oracle JDeveloper, or post-deployment through the application server management interfaces (WebLogic Scripting Tool and Oracle Enterprise Manager). See *Oracle Fusion Middleware Administrator's Guide* for more information.

This section contains the following:

- Section 24.2.1, "Using External Applications"
- Section 24.2.2, "Supplying User Credentials"
- Section 24.2.3, "Managing External Applications"

24.2.1 Using External Applications

The WebCenter External Application Service provides a way for mapped user identities to be passed to a web application that requires its own authentication. The support for external applications and credential mapping provided by the WebCenter External Application Service can be used to set up a secured service connection and to provide a seamless automated single sign-on experience for the user.

This is described in the following sections:

- Section 24.2.1.1, "Secured Service Connections"
- Section 24.2.1.2, "Automated Single Sign-On"

24.2.1.1 Secured Service Connections

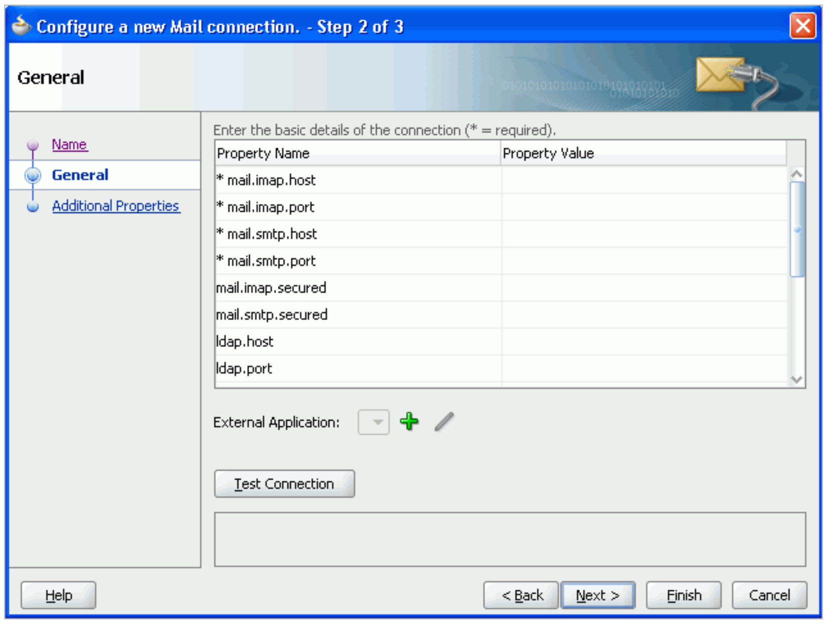
To use an external application definition with a secured service (such as a mail server or portlet producer) you associate the named external application with the connection configuration to the required service. For example, the connection to a mail server requires the user to supply a valid username and password in order to see their mail. Therefore, by associating an external application to the IMAP server connection definition, the user's credentials are automatically passed as part of the mail request as shown in Figure 24–1.

Note: The following WebCenter Web 2.0 Services must be configured with external applications for credential mapping support to be available:

- Instant Messaging and Presence (IMP)
- Mail
- Documents
- RSS

For more information about the identity propagation mechanisms used by WebCenter Web 2.0 Services, see Section 24.11, "Identity Propagation Mechanisms".

Figure 24–1 Configure a New Mail Connection Page

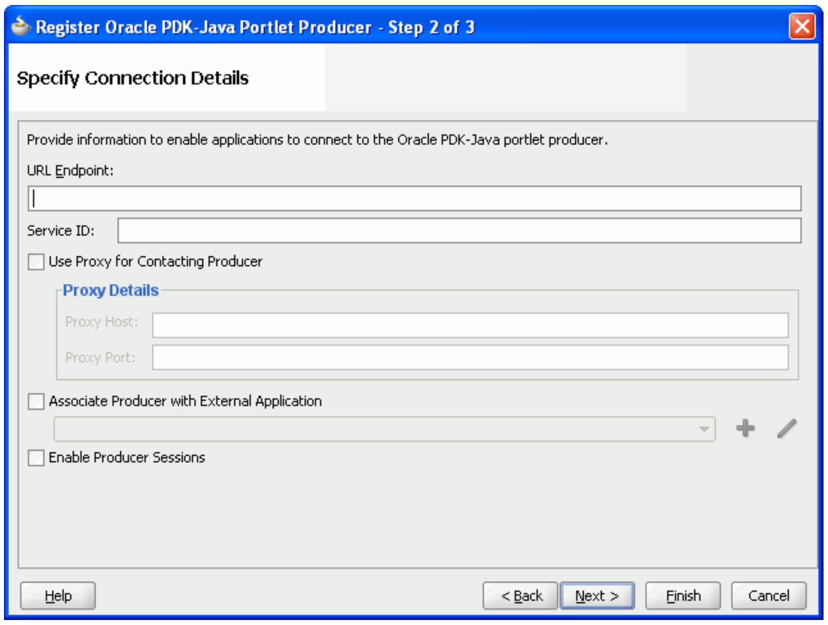


Enter the basic details of the connection (* = required).

Property Name	Property Value
* mail.imap.host	
* mail.imap.port	
* mail.smtp.host	
* mail.smtp.port	
mail.imap.secured	
mail.smtp.secured	
ldap.host	
ldap.port	

When a portlet producer depends on an application that handles its own authentication, you can associate the producer with an external application so that when you register the producer it is a simple task to select the appropriate external definition that maps to the application that is exposed within the portlet, as shown in Figure 24–2.

Figure 24–2 External Application URL



Register Oracle PDK-Java Portlet Producer - Step 2 of 3

Specify Connection Details

Provide information to enable applications to connect to the Oracle PDK-Java portlet producer.

URL Endpoint:

Service ID:

Use Proxy for Contacting Producer

Proxy Details

Proxy Host:

Proxy Port:

Associate Producer with External Application

Enable Producer Sessions

At run time, the producer uses the information associated with the external application to authenticate the user to the application, and consequently consume its portlets. The producer code is responsible for actually performing the authentication interaction with the external application. The external application support provided with WebCenter Framework simply provides the information needed for authentication to the portlet producer. The use of external applications is supported for both Oracle PDK-Java as well as WSRP producers.

For example, a producer provides a stock portfolio portlet from a portlet-producing application that has its own authentication mechanism. In this case the developer:

- Defines the external application. This can be done through the Oracle JDeveloper wizard or through Oracle Enterprise Manager.
- Associates the external application with the portlet producer.

24.2.1.2 Automated Single Sign-On

With automated single sign-on, the user directly links to the application and is automatically authenticated to the secured web application, as their credentials are retrieved from the credential store. This provides the end user with a seamless single sign-on experience.

Note: Automated login is not supported for external applications using BASIC authentication.

Rather than using a URL directly to the web application, links to the application are proxied through the external application's automated login servlet (adfextapplogin). If the user is not authenticated to the external application and they have not previously stored their credentials in the credential store, they will be challenged for their password via the credential provisioning page discussed below. If, however, the user has previously defined credentials, they will be returned from the credential store and the user will automatically be logged on to the application.

The proxy URL references the external application in question and redirects to the URL that is specified in the external application definition.

```
/adfextapplogin?extappid=<extappid>
```

For example, if you had a WebCenter application in which you defined an external application that represented the myoracle.com Web site (external application identifier is "myoracle"), the proxy URL would look like this:

```
/adfextapplogin?extappid=myoracle
```

The link's `target` attribute should also be set appropriately. For example, if you use `<a href=>`, then set the `target` attribute appropriately in addition to specifying the target in the `href`. The target attribute specified for the servlet will determine how the Cancel button functions as described below.

```
/adfextapplogin?extappid=<extappid> [target= _self | _blank]
```

If you specify `target=_blank` the link opens in new window. If you specify `target=_self` the link opens in current window. If the `target` parameter is not specified the link opens in current window.

This parameter also affects how the **Cancel** button on the credential provisioning page works. If `_blank` is specified, the new window is closed when **Cancel** is clicked; if `_self` is specified (or the `target` parameter is not used), the user is returned to the calling page.

Note: Automated login for external sites is not supported for sites that do not support UTF8 encoding.

24.2.2 Supplying User Credentials

How you allow an end user to define their credentials for an external application depends on the use of the external application. For most services, the credential provisioning screen is incorporated into the task flow that the service exposes and for these no further configuration steps are required. You can, however, add the `External Application - Change Password` task flow component to applications using these services thereby allowing the end user to preemptively set the appropriate user name and password for each of the external applications that is registered with your custom WebCenter application.

The `External Application - Change Password` task flow displays all external applications defined in the application that do not specify shared credentials (for more information about shared credentials, see Section 24.2.3, "Managing External Applications"). Note that the user must to be authenticated to view this task flow.

For the Instant Messaging and Presence service, however, you must explicitly add the `External Application - Change Password` task flow component to your application from the Resource Palette or Component Palette. For step-by-step instructions on how to configure security for the Instant Messaging and Presence service using the `External Application - Change Password` task flow component, see Section 15.2.2, "Adding the IMP Service at Design Time".

At run time, the credential provisioning screen displays login data fields composed of the data fields specified through external application registration. Users fill in the data fields with their login information for the specified external application and that login information is passed to the external application or service. Entering the credentials in the provisioning screen also results in the credentials being persisted in the credential store configured for the WebLogic domain.

By default, the login information the user entered is preserved in a credential store, which handles logins for future sessions. The user does not have to enter login information again (unless the user's credentials change). However, the end user can choose to use the information for the current session by deselecting the **Remember my Login Information** checkbox on the credential provisioning page.

24.2.3 Managing External Applications

This section provides information about registering external applications. Additionally, it describes the process of editing and deleting registration details. It contains the following subsections:

- Section 24.2.3.1, "Working with External Applications in Oracle JDeveloper"
- Section 24.2.3.2, "Working with External Applications in Enterprise Manager"
- Section 24.2.3.3, "Working with External Applications Using WLST"

24.2.3.1 Working with External Applications in Oracle JDeveloper

This section provides information about registering external applications and editing and deleting registration details in Oracle JDeveloper. It contains the following subsections:

- Section 24.2.3.1.1, "Registering an External Application in Oracle JDeveloper"
- Section 24.2.3.1.2, "Editing External Application Registration Details in Oracle JDeveloper"
- Section 24.2.3.1.3, "Deleting External Application Registration Details in Oracle JDeveloper"

24.2.3.1.1 Registering an External Application in Oracle JDeveloper

Use the Register External Application Wizard to identify and store information about the type of data required to authenticate to an external application, such as the names of login fields.

To register an external application in Oracle JDeveloper:

1. In the Applications Navigator, right-click a WebCenter application or project and select **New** from the context menu.
2. In the New Gallery, select **External Applications** under the **General** node.
3. In the right pane, select **External Application**, and click **OK**.

   This displays the Register External Application Wizard.
4. On the Name page, use the **Create external application in** option to specify whether the external application can be reused in other WebCenter applications. Select **Application Resources** to make the external application available only in the WebCenter application in which it is registered, or select **Resource Palette** to make the external application available from the Resource Palette to any new WebCenter applications you create in Oracle JDeveloper.

   If you choose **Resource Palette**, then the external application connection will be visible under IDE connections in the Resource Palette. If you want to use it in an application, you can right click the external application from resource palette and click **Add to Application**.
5. In the **Name** field enter a unique name to identify the application.

This name must be unique within the WebCenter application, and among other connections as well. Note that you cannot edit this field afterwards.

6. In the **Display Name** field enter a name for the application that end users will see in the credential provisioning screens.

   You can modify the Display Name at any time as described in Section 24.2.3.1.2, "Editing External Application Registration Details in Oracle JDeveloper". If you leave the field blank the display name is set to the value of the Name field by default.

7. Click **Next**.

8. On the General page, in the **Login URL** field enter the URL to which the HTML login page is submitted.

   View the HTML source of the application's login form to retrieve this URL.

   Note: The fields on the General pane are only required if the external application being defined participates in click-through login as described in Section 24.2.2, "Supplying User Credentials".

   ---

9. In the **User Name/ID FieldName** field, enter the label that the application uses for the user name field, for example, `User Name`.

10. In the **Password FieldName** field, enter the label that the application uses for the password field, for example `Password`.

11. From the **Authentication Method** list, select the application's login method.

    Choose from the following:

    - **GET**

      Presents a page request to a server. Submits the login credentials as part of the login URL.

    - **POST**

      Submits login credentials within the body of a form.

12. Click **Next**.

13. On the Additional Fields page, enter the names and values of any additional fields that are submitted with the external application's login form:

    Click the **Add Field** button to create a new input field:

    - **Field Name**

      Enter a unique name for any additional field that requires user input on the external application HTML login form.

    - **Field Value**

      Enter a default value for the corresponding field name.

    - **Display to User**

      Select to display the field on the external application login screen. If the field is not displayed (unchecked), then a default value must be specified, which will be used to login into the external application for all users. If the value is user-specific, then the field must be displayed to the user provisioning page.

Note: The Delete Field option can be used to delete selected rows.

14. Click **Next**.
15. On the Shared Credentials page, select the **Specify Shared Credentials** check box and specify a **Username** and **Password** if you want all authenticated users to access the external application using this credential. Authenticated users will not be challenged to provide their credentials when they access the external application.
16. Click **Next**.
17. On the Public Credentials page, select the **Specify Public Credentials** check box and specify a **Username** and **Password** to be used for all unauthenticated (public) users accessing the external application. This is required when the external application content is accessed through one of the WebCenter Services (such as Document Library, or Instant Messaging and Presence) and the portlet is placed on a public page.
18. Click **Finish** to register the external application.

After registering your external application, you have to configure the application to allow the end user to define the username and password. You can do this by dropping an `External Application - Change Password` task flow component into your application. This allows the end user to preemptively set the appropriate username and password for each of the external applications that is registered with your custom WebCenter application.

To configure the application to allow the end user to define the username and password, perform the following steps:

1. Open a JSF Page in your ViewController project.
2. From the Resource Palette, under My Catalogs, WebCenter Services Catalog, expand **Task Flows**.
3. Drag an `External Application - Change Password` task flow component onto your page.
4. When prompted, choose **Region** as the way to create the task flow.
5. Secure the page as this taskflow is accessible only to authenticated users

24.2.3.1.2 Editing External Application Registration Details in Oracle JDeveloper Use the Edit External Application Registration Information Wizard to revise the registration details provided for an external application.

To edit external application registration details:

1. In the Application Navigator, from the Application Resources pane under the Connections node, right-click an external application and select **Properties** from the context menu.
2. In the Edit External Application Wizard, click a link to show a page and revise its values.

   Choose from the following:

   - **Name**
   - **General**
   - **Additional Fields**

- **Shared Credentials**
- **Public Credentials**

  For more information, see Section 24.2.3.1.1, "Registering an External Application in Oracle JDeveloper".

3. Click **OK** to save your changes and exit the wizard, or click **Cancel** to exit the wizard without saving.

24.2.3.1.3 Deleting External Application Registration Details in Oracle JDeveloper

To delete external application registration information, perform the following steps:

1. In the Application Navigator, from the Application Resources pane under the Connections node, right-click an external application and select **Delete** from the context menu.

   Alternatively, you can select an external application in the Applications Navigator and from the Edit menu, select **Delete**.
2. In the External Application Delete dialog box, select **Yes**.

Oracle recommends that you remove any references to services, such as a portlet producer, with which the external application is associated. Failing to do so will likely result in runtime errors, because the components will attempt to communicate with the external application.

24.2.3.2 Working with External Applications in Enterprise Manager

Just as you can create, edit, and delete external application registration details in Oracle JDeveloper, you can also do this in Enterprise Manager. See the section "Working with External Applications" in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

24.2.3.3 Working with External Applications Using WLST

As well as Oracle JDeveloper and Enterprise Manager, you can also use WebLogic Scripting Tool (WLST) commands to create, edit, and delete external application registration details. See the section "Working with External Applications" in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

24.3 Setting Up Security for Your Application

Refer to the section on *adding security to an ADF application* in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework* for information about securing your application by using Oracle ADF Security, which is based on JAAS.

24.4 Creating a Login Portlet

Refer to the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework* for more information on creating a login portlet for your application.

24.5 Creating Login Pages and a Login Component

Refer to the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework* for more information on creating login pages and login components.

24.6 Adding Portlets to a Login Page

The *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework* describes how you can create an ADF Faces-based login page for your application. In this section, you will add some portlets to such a login page so that the login page becomes indistinguishable from the other pages in your WebCenter application.

Make sure your portlet producers have been registered before proceeding. See Section 9.2, "Registering Portlet Producers with a Custom WebCenter Application" for details.

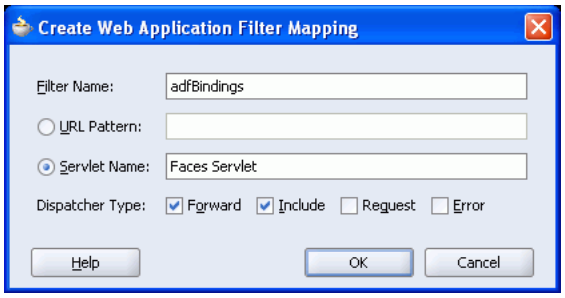
To add portlets to the login page, perform the following steps:

1. Drag a **PanelCustomizable** onto the `h:form` tag that is inside the first `cust:panelCustomizable` tag you added to the login page.
2. From the Component Palette, select **RichTextPortlet Producer**, then select the Rich Text portlet from the list and drag it onto the `PanelCustomizable` component.
3. From the Component Palette, select **ADF Faces Core** and drag an **ObjectSeparator** below the Rich Text portlet on the `PanelCustomizable` component.
4. From the Component Palette, select **OmniPortlet Producer**, then select the OmniPortlet from the list and drag it onto the `PanelCustomizable` component.
5. Save the page.

Because the login page is called from the container as part of the login process, the request must be forwarded to the ADF binding filter to establish the appropriate portlet and security context. To do this, you must configure a mapping for the ADF Binding filter in the `web.xml` file. To do this, perform the following steps:

1. In the Applications Navigator, expand the WEB-INF node, right-click **web.xml** and select **properties** to open the property palette.
2. Select **Filter Mappings** in the left panel and click **add** to define a new mapping for the `adfBindings` Filter. This displays the Create Web Application Filter Mapping dialog box.
3. Specify **adfBindings** for the filter name and click the Servlet Name option and specify **Faces Servlet** as the servlet name. Ensure that the Forward and Include dispatcher types are selected as shown in Figure 24–3.

Figure 24–3 Create Web Application Filter Mapping Dialog Box



Create Web Application Filter Mapping

Filter Name: adfBindings

( ) URL Pattern:

(•) Servlet Name: Faces Servlet

Dispatcher Type: [✓] Forward [✓] Include [ ] Request [ ] Error

4. Click **OK**.

Follow the steps in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework* to complete the creation of the login page.

24.7 Creating a Public Welcome Page for Your Application

WebCenter applications are not secured by default. If you have implemented ADF Security for your WebCenter application and intend to make part of the application public, you must provide a starting point or home page for unauthenticated users. To create this public welcome page, refer to the section on *adding security to an ADF application* in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

24.8 Configuring Basic Authentication for Testing Portlet Personalization

Portlet personalizations are tied to particular, authenticated users. Hence, when running a portlet that has an Edit mode, the Personalize option in the portlet's dropdown menu only appears to authenticated users of the application. Anonymous or public users will not have the option to personalize the portlet. If you are a developer creating portlets and pages, then you may want to quickly test the Edit mode of your portlet without creating a complete security model for your application. To perform this sort of testing, you can easily configure some very basic authentication for your application and then remove it when you are done testing:

Note: This procedure is useful for any portlet that has an Edit mode (Omniportlet, Web Clipping, JPS, and PDK-Java).

1. Create a user `sking` and the role `manager`. See the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework* for information about creating users and roles.
2. Secure your application using the ADF Security Wizard. See the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework* for the steps to be performed. On the Login page of the wizard, select **HTTP Basic Authentication (RFC 2617)**. This specifies that the application will use basic authentication.
3. Run the page in the integrated WLS and log in as a valid user and test your portlet's edit mode.

When you are done testing your portlet's Edit mode, you can quickly remove this test security by do the following:

1. In the Applications Navigator, click the project that contains a page with the portlet you want to test.
2. From the **Tools** menu, choose **ADF Security Wizard**.
3. If the Welcome page appears, then click **Next**.
4. Choose **Remove All ADF Security Settings**.
5. Click **Next** until you come to the Finish page of the wizard. Click **Finish**. The security is removed. If you want to ensure that the security has been removed, then exit your browser and rerun the application. When you access the page, you should not be prompted to login and the personalize option should be gone from the portlet's dropdown menu.

24.9 Registering Custom Certificates with the Keystore

Secure Sockets Layer (SSL) Communication requires the use of trusted certificates issued by a certificate authority, which vouches for the authenticity of the certificates that it issues or signs. Widely accepted certificate authorities are listed in the keystore, the `cacerts` file, available in the `<JDEV_HOME>\jdk\jre\lib\security` directory. If a portlet producer uses a security certificate issued by a non-widely accepted certificate authority and you try to access portlets from this producer, a security alert is displayed informing you that the security certificate was issued from a certificate authority you do not trust. This means the certificate is not available in the keystore. To avoid being prompted each time you access such portlets, you must register this certificate with the keystore.

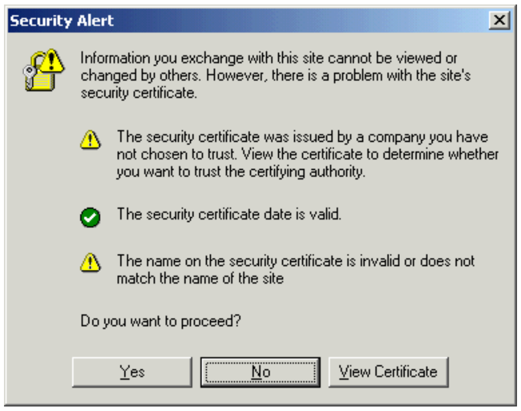
To register a certificate with the keystore, perform the following steps:

1. Navigate to `JDEV_HOME\jdk\jre\lib\security`.
2. Back up the `cacerts` file.
3. Access the producer URL in Internet Explorer to get the certificate.

> **Note:** Recent versions of FireFox do not provide a means to export certificates.

4. In the Security Alert dialog box, shown in Figure 24–4, click **View Certificate**.

Figure 24–4 Security Alert Dialog Box



Information you exchange with this site cannot be viewed or changed by others. However, there is a problem with the site's security certificate.

The security certificate was issued by a company you have not chosen to trust. View the certificate to determine whether you want to trust the certifying authority.

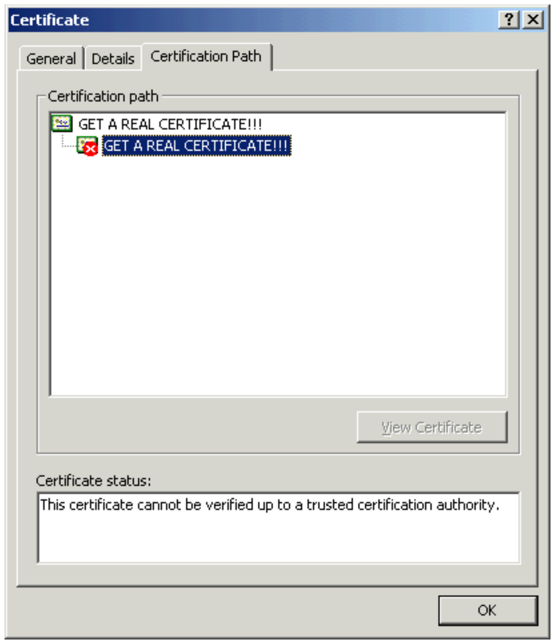
The security certificate date is valid.

The name on the security certificate is invalid or does not match the name of the site

Do you want to proceed?

5. In the Certificate dialog box, click the **Certification Path** tab.
6. The dummy child certificate is selected by default as shown in Figure 24–5. Select the root certificate and click **View Certificate**.

Figure 24–5 Certificate Dialog Box



Certification path

GET A REAL CERTIFICATE!!!

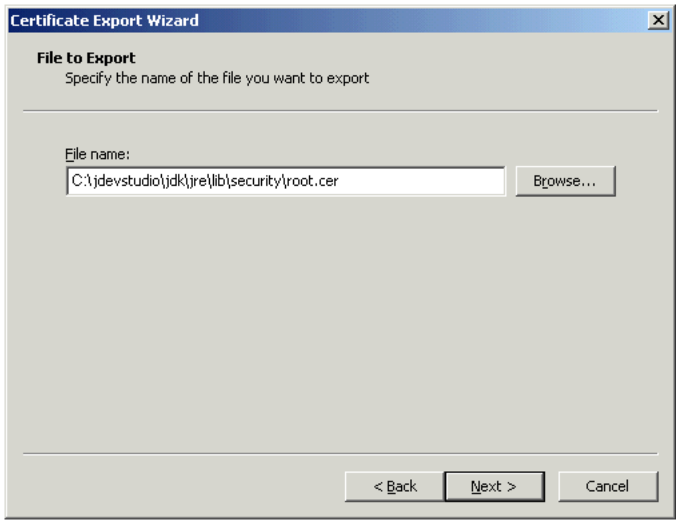
GET A REAL CERTIFICATE!!!

Certificate status:

This certificate cannot be verified up to a trusted certification authority.

7. Click the **Details** tab, and click **Copy to File**.
8. In the Certificate Export Wizard, accept the default settings and click **Next** until you reach the File to Export screen, shown in Figure 24–6.

Figure 24–6 File to Export Screen of the Certificate Export Wizard



File to Export

Specify the name of the file you want to export

File name:

C:\jdevstudio\jdk\jre\lib\security\root.cer

9. In the File Name field, enter <*JDEV_HOME*>`\jdk\jre\lib\security\root.cer` and click **Next**.
10. Click **Finish**.
11. In the command prompt, set your default directory to <*JDEV_HOME*>`\jdk\jre\lib\security` and run the following command:

```
keytool -import -file root.cer -keystore cacerts -storepass changeit
```

By running this command, the `root.cer` certificate is imported into the keystore.

12. Enter `y` at the prompt to confirm that you trust this certificate.
13. Verify that the `cacerts` file is updated with the certificate.

24.10 Overriding Inherited Security on Portlets and Customizable Components

Individual actions on portlets and customizable components are not secured by default. Rather, the ability to customize a portlet or customizable component as a whole is inherited from the page permissions. If you want to grant more granular activities within a portlet or customizable component, then you can override the page-level security inheritance and define security directly on the required actions.

The ability of a user to perform actions on portlets and customizable components is inherited from the page security based on the value of the application-wide switch, `enableSecurity`, in the `adf-config.xml` file. If you selected the WebCenter application template while creating your application, then the `adf-config.xml` file is located in the `<`*`APPLICATION_NAME`*`>/.adf/META-INF` directory. The `enableSecurity` element is not available by default in `adf-config.xml`. To override or extend the page-level security inheritance for portlets and customizable components, you must add the portlets security and customizable components security sections in the `adf-config.xml` file, as shown in Example 24–1 and Example 24–2 and set the `enableSecurity` element in those sections to `true`.

Example 24–1 enableSecurity Element in the Portlet Security Section in adf-config.xml

```
<!--
==============================================================================
PORTLETS ACTIONS SECURITY
==============================================================================
-->
<adfp:adf-config-child xmlns:adfp="http://xmlns.oracle.com/adfp/portlet">
  <adfp:enableSecurity value="true"/>
    <adfp:actionsCategory>
       ........................................
</adfp:adf-config-child>
```

Example 24–2 enableSecurity Element in the Customizable Components Security Section in adf-config.xml

```
<!--
==============================================================================
CUSTOMIZABLE COMPONENTS ACTIONS SECURITY
==============================================================================
-->
<cust:customizableComponentsSecurity
xmlns:cust="http://xmlns.oracle.com/adf/faces/customizable/config">
  <cust:enableSecurity value="true"/>
    <cust:actionsCategory>
       ........................................
</cust:customizableComponentsSecurity>
```

Security for actions on portlets and customizable components can be implemented at the following levels:

- Page level: You can define security for portlets and customizable components such that page-level privileges are inherited by these components. This is the default behavior.

By default, portlets and customizable components inherit allowable actions from the defined page-level permissions such as personalize or customize. That is, a user who has *customize* privileges on the page has permission on the customize action for the components on that page. The `enableSecurity` element enables you to override the security inheritance behavior and can take either of the following values:

– `true`: If set to `true` (the default), then the ability for a user to modify a component will first be determined from the page permissions and then adjusted according to the current set of actions defined for that type of permission. If a user has customize permission, then the actions that constitute the customize category (move, customize, and so on) are available to the user, but they will be overridden by the actions that are defined in the `adf-config.xml` file. For example, a page designer wants to allow the end user to be able to customize portlets, but not customize the page layout. By setting `enableSecurity` to `true`, the page designer enforces that users must first have customize permission on the page. Setting `customizeActionsCategory` to `false` for customizable components will prevent the customization of the page layout, yet still allowing portlet customization. (As the default for `customizeActionsCategory` is `true`, it does not need to be set explicitly for portlets.).

– `false`: If set to `false`, then all the actions are available to users. The user's page permissions and actions configured in `adf-config.xml` are ignored.

- Actions category level: You can define security on all actions for portlets or customizable components that belong to a named category.

  You can add an `actionsCategory` element in the `adf-config.xml` file to define security on multiple actions simultaneously. Depending on the `actionCategory` attributes that you enable, appropriate privileges are provided on the portlets or customizable components.

- Actions level: You can define security on individual actions for portlets or customizable components.

  You can use the `actions` element in the `adf-config.xml` file to enable or disable individual actions. Depending on the `actions` attributes that you enable, appropriate privileges are provided on the portlets or customizable components.

Notes:

- Privileges can be inherited from the parent only. Inheritance from a component in any other position in the hierarchy is not supported
- Although the security override implementation for portlets and customizable components is similar, they are independent from each other. Therefore, if you place a portlet inside a customizable component (for example, in a `PanelCustomizable` component), the portlet will not inherit override settings from the customizable component. Instead it will use the security override settings that are defined for portlets.
- Settings made at the actions category level or actions level are applicable for all component instances in the application. These settings cannot be made for a single instance of a portlet or customizable component.

Section 24.10.1, "Portlets Security" describes how you can implement security on portlets at the actions category level and actions level. For information on how to define security for actions on customizable components at the application level, see Section 5.8.4, "Applying Action-Level Restrictions on Show Detail Component Actions."

24.10.1 Portlets Security

You can define portlet security if actions on portlets are inherited from the page at the application level by setting `enableSecurity` to `true` in the portlets security section of the `adf-config.xml` file. A value of `true` implies that the user's permissions are determined from the page permission and then augmented according to the `actionsCategory` and `actions` elements specified. By defining actions categories and individual actions, you can control the exposure of the individual actions available within the given page permissions.

To implement security for actions on portlets at various levels as described earlier, you must define security settings at the following sections:

- Defining Security at the Actions Category Level
- Defining Security at the Actions Level

24.10.1.1 Defining Security at the Actions Category Level

You can add an `actionsCategory` element in the portlets security section in the `adf-config.xml` file to define the group of actions that are exposed on the portlets within the application. Depending on the `actionsCategory` attributes that you enable, appropriate privileges are provided on the portlets. Table 24–1 describes the different `actionsCategory` attributes and the portlet actions they support by default.

Table 24–1 actionsCategory Attributes and Portlets Actions Mapping

Attribute Value	Actions Supported
viewActionsCategory	Render isHelpModeAvailable isNormalModeAvailable isAboutModeAvailable isPreviewModeAvailable isDetailModeAvailable isLinkModeAvailable isPrintModeAvailable
customizeActionsCategory	showMoveAction showRemoveAction isCustomizeModeAvailable showMinimizeAction showMaximizeAction isConfigModeAvailable
personalizeActionsCategory	isPersonalizeModeAvailable

Example 24–3 shows the `actionsCategory` entry that you can add to the portlets security section in the `adf-config.xml` file. In this example, `customizeActionsCategory` is set to `false` to prevent customization. You can use Expression Language (EL) for the values of these elements.

Example 24–3 actionsCategory Element in the Portlets Security Section

```
<!--
============================================================================
PORTLETS ACTIONS SECURITY
============================================================================
-->
<adfp:adf-config-child xmlns:adfp="http://xmlns.oracle.com/adfp/portlet">
  <adfp:enableSecurity value="true"/>
    <adfp:actionsCategory>
      <adfp:actionCategory name="viewActionsCategory" value="true"/>
      <adfp:actionCategory name="customizeActionsCategory" value="false"/>
      <adfp:actionCategory name="personalizeActionsCategory" value="true"/>
    </adfp:actionsCategory>

    <adfp:actions>
    .........................................
    </adfp:actions>

</adfp:adf-config-child>
```

24.10.1.2 Defining Security at the Actions Level

You can use the `actions` element in the portlets security section of the `adf-config.xml` file to enable or disable individual portlet actions. Depending on the `action` attributes that you enable, appropriate privileges are provided on the portlets.

Example 24–4 shows an example of an `actions` entry that you can add to the portlets security section in the `adf-config.xml` file. You can use EL for the values of these elements. In this case you prevent customization by setting `isCustomizeModeAvailable` to `false`.

Example 24–4 actions Element in the Portlets Security Section

```
<!--
============================================================================
PORTLETS ACTIONS SECURITY
============================================================================
-->
<adfp:adf-config-child xmlns:adfp="http://xmlns.oracle.com/adfp/portlet">
  <adfp:enableSecurity value="true"/>
    <adfp:actionsCategory>
    .........................................
    </adfp:actionsCategory>

    <adfp:actions>
      <adfp:action name="Render" value="true"/>
      <adfp:action name="showMoveAction" value="true"/>
      <adfp:action name="isCustomizeModeAvailable" value="false"/>
      <adfp:action name="isPersonalizeModeAvailable" value="true"/>
    </adfp:actions>

</adfp:adf-config-child>
```

Using EL to Prevent Customization of Portlets Outside of Business Hours

An example to show when you may need to override inherited portlet security is an application that is configured to disable portlet customization outside of standard business hours. For this, you must first create a managed bean (for example, a managed bean called `appBusinessRules`), containing the method shown in Example 24–5.

Example 24–5 InsideBizHours Method Defined in appBusinessRules Managed Bean

```
public boolean isInsideBizHours()
  {
    Calendar rightNow = Calendar.getInstance();
    int      currentHr = rightNow.get(rightNow.HOUR_OF_DAY);

    // Do not allow customize operation outside of standard business hours

    if ((currentHr > 9) && (currentHr < 17))
       return true;
    else
       return false;
  }
```

You can then reference this managed bean from the `actionsCategory` element in the portlet security section of the `adf-config.xml` file, as shown in Example 24–6.

Example 24–6 InsideBizHours Method Referenced in the adf-config.xml File

```
<adfp:adf-config-child xmlns:adfp="http://xmlns.oracle.com/adfp/portlet">
  <adfp:enableSecurity value="true"/>

    <adfp:actionsCategory>
      <adfp:actionCategory name="customizeActionsCategory"
      value="#{appBusinessRules.InsideBizHours?"true":"false"}"/>
    </adfp:actionsCategory>

</adfp:adf-config-child>
```

In this example, the `customizeActionsCategory` will be set to `true` only if the application is run within business hours. Outside of these hours, the portlet cannot be customized even if the user had that permission granted on the page. All other categories that are not explicitly defined, will be inherited from the page.

Using EL to Prevent Personalization and Customization of Portlets Outside the Corporate Network

In this example the managed bean checks the IP address of the request to determine whether the user has accessed the application through the corporate proxy server or from within the corporate network. In this simple example, assume that if the request has the proxy server's IP address, then it is coming from outside the corporate network. In general it is not advised to base security strictly on IP addresses, because these can be compromised. For this, you must add the method shown in Example 24–7 to the managed bean:

Example 24–7 InsideCorpNetwork Method Defined in appBusinessRules Managed Bean

```
public boolean isInsideCorpNetwork()
  {
    // Do not allow personalize and customize operations
    // for requests that go through the corporate proxy
    FacesContext       ctx = FacesContext.getCurrentInstance();
    ExternalContext    ectx = ctx.getExternalContext();
    HttpServletRequest myrequest = (HttpServletRequest) ectx.getRequest();
    String             currentIP = myrequest.getRemoteAddr();

    if (currentIP.equals(getProxyServerIP()))
        return false;
    else
        return true;
  }
```

You can then reference this managed bean from the `actionsCategory` element in the portlet security section of the `adf-config.xml` file, as shown in Example 24–8.

Example 24–8   InsideCorpNetwork Method Referenced in the adf-config.xml File

```
<adfp:adf-config-child xmlns="http://xmlns.oracle.com/adfp/portlet">
  <adfp:enableSecurity value="true"/>

    <adfp:actionsCategory>
      <adfp:actionCategory name="customizeActionsCategory"
                    value="#{appBusinessRules.InsideCorpNetwork?"true":"false"}"/>
      <adfp:actionCategory name="personalizeActionsCategory"
                    value="#{appBusinessRules.InsideCorpNetwork?"true":"false"}"/>
    </adfp:actionsCategory>
</adfp:adf-config-child>
```

In this example, the `customizeActionsCategory` and the `personalizeActionsCategory` will be set to `true` only if the IP address of the request for the application does not match that of the corporate proxy. The assumption is that the internal requests would have a valid client IP address. All other categories that are not explicitly defined, will be inherited from the page.

24.11 Identity Propagation Mechanisms

The following table lists the identity propagation mechanisms employed by WebCenter Services for propagating the end-user's identity to the various information sources from which content is being integrated into the WebCenter application.

Whenever possible, WS-Security is the preferred means of identity propagation. Where WS-Security cannot be used due to legacy restrictions or pre-defined store-specific standards or specifications, the primary mechanism used is the credential mapping capability provided by the External Applications service to obtain the user's credentials for a remote application using a distinct security provider. For more information about WS-Security, see the *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*. For more information about External Applications, see Section 24.2, "Working with External Applications".

Table 24–2   Identity Propagation Mechanisms

Service	Connection Type	Identity Propagation Mechanism
Conference	Webex	External Application
Discussions	Jive	Oracle WS-Security
Announcements	see Discussion Forum	

Table 24–2 (Cont.) Identity Propagation Mechanisms

Service	Connection Type	Identity Propagation Mechanism
Documents	Oracle Content Server	Proprietary ID propagation mechanism through socket connection. Can use SSL with mutual authentication, or clear socket with IP authorization (or use External Application option)
	File System	N/A
	Portal 10g/11g	JSR-170 (External Application)
	Day Adapters	JSR-170 (External Application)
EMail	EMail Connection	External Application
	LDAP Connection	External Application
Events - Personal	Exchange Server Connection	External Application
External Application	HTTP Request from Browser	External Application
IMP	Microsoft Live Communication Server (LCS)	SOAP, Web Services calls. External Application
	Oracle WebLogic Communications Services (OWLCS)	Oracle WS-Security
Portlet Producers	WSRP Producer (Secure)	Oracle WS-Security (Recommended by WSRP Specification)
	WSRP Producer (Non-Secure)	WSRP userContext in SOAP message (WSRP Specification)
	JPDK Producer (External App)	External Application JPDK payload includes the user information and is conveyed to the producer in the ProviderUser. The information also includes a mapped username and password. (Proprietary, Legacy)
	JPDK Producer (Non-Secure)	Username in render request (Proprietary, Legacy)
Search	Secure Enterprise Search (SES)	Web Service Call
Wiki	Browser Connection	SSO mechanisms - OSSO/OAM or other WLS supported SSO mechanism
Worklist Service	BPEL Connection	Web Service call Oracle WS-Security

24.12 Securing Identity Propagation Through WSRP Producers with WS-Security

The Web Services for Remote Portlets (WSRP) specification indicates that Web Services Security (WS-Security) can be leveraged for providing secure identity propagation between the consumer and the portlet producer. However, WSRP in and of itself does not provide secure identity propagation of the end user's identity to the portlet producer. The WSRP specification explicitly defers to other security standards for secure identity propagation and does not go into the specific WS-Security profiles or options that should be employed. In the absence of a secure mechanism, WSRP defines the concept of *user categories*, which can be mapped to security roles like the ones used by the JSR168 portlets. By using a combination of WSRP and WS-Security, you can ensure end-to-end security.

This section covers the following:

- Identity Propagation Without WS-Security
- Identity Propagation with WS-Security
- Configuring Security for WSRP Portlets

24.12.1 Identity Propagation Without WS-Security

When using WSRP without WS-Security, the `userContext` structure within the SOAP message contains user profile information and user category information. This information is not considered secure and should only be used for personalization and customization functionality. It should not be used for authorization of sensitive resources. This information is also exposed in the JSR168 APIs, `isUserInRole(`*`role`*`)` and `getUserPrincipal`. The code in Example 24–9 shows how a sample portlet's markup rendering code may use the `isUserInRole` API to decide what content to display.

Example 24–9 isUserInRole(role) API

```
private void doViewHtml(RenderRequest
request, RenderResponse response)
throws PortletException, IOException
{
// To do: markup the required content.
PrintWriter out = response.getWriter();
out.print("<p>Welcome");
out.println("</p>");
if (request.isUserInRole("moderator"))
   {out.println("<p>MODERATOR</p>" );}
else
   {out.println("<p>not moderator</p>" );}
if (request.isUserInRole("participant"))
   {out.println("<p>PARTICIPANT</p>" );}
else
   {out.println("<p>not participant</p>" );}
if (request.isUserInRole("viewer"))
   {out.println("<p>VIEWER</p>" );}
else
   {out.println("<p>not viewer</p>" );}
    }
```

24.12.2 Identity Propagation with WS-Security

When WS-Security is leveraged with WSRP, the user's identity is propagated outside of the SOAP message body, in the WS-Security header. This is a user assertion, using the Username Token format, and is digitally signed to authenticate the consumer and to ensure the integrity of the assertion.

When this mechanism is used, the JSR 168 APIs `isUserInRole` and `getUserPrincipal` are established based on the security context resulting from the

WS-Security authentication, rather than the information in the SOAP message's `userContext`.

The use of WS-Security adds some complexity to the configuration and management of the WebCenter application and the set of producers it consumes. However, when the situation warrants its use, it becomes an important ingredient of the SOA architecture that ensures the security of the information being published by the WebCenter application.

Oracle WebCenter Framework supports the following token profiles (to digitally sign the security token and message body to ensure authenticity and integrity):

- Username token without password
- Username token with password
- SAML token (uses the *sender vouches* method that the producer uses to confirm the subject assertion)

Digitally signing the security token and the SOAP message body accomplishes the following objectives:

- Consumer Authentication
- Assertion and Message Integrity
- Supported Producers
- Security Domain Implication

Consumer Authentication

When a portlet producer is generating sensitive information, for example paystub information, it is imperative that it only responds to requests to show the information from a legitimate consumer.

By using WS-Security, and having the consumer digitally sign the security token and the message body, the producer can verify the signature using the public key of the legitimate consumer. If the signature cannot be verified, then it means that the request may have come from a fraudulent consumer. By requiring the verification of the digital signature, the sensitive information will only be sent to the legitimate consumer.

Assertion and Message Integrity

In addition to verifying the identity of the consumer making the Web Service requests, digitally signing the security token and the message body also ensures that the token and the message have not been tampered with. This prevents such problems as man-in-the-middle attacks where a legitimate request might be intercepted and the user name in the security token replaced with another user name to see the paystub information coming back for the other user. By digitally signing the token, it cannot be tampered with. Any modification to the token would result in the inability to verify the signature on the producer end, and would result in a SOAP fault to be returned instead of the requested paystub information.

Supported Producers

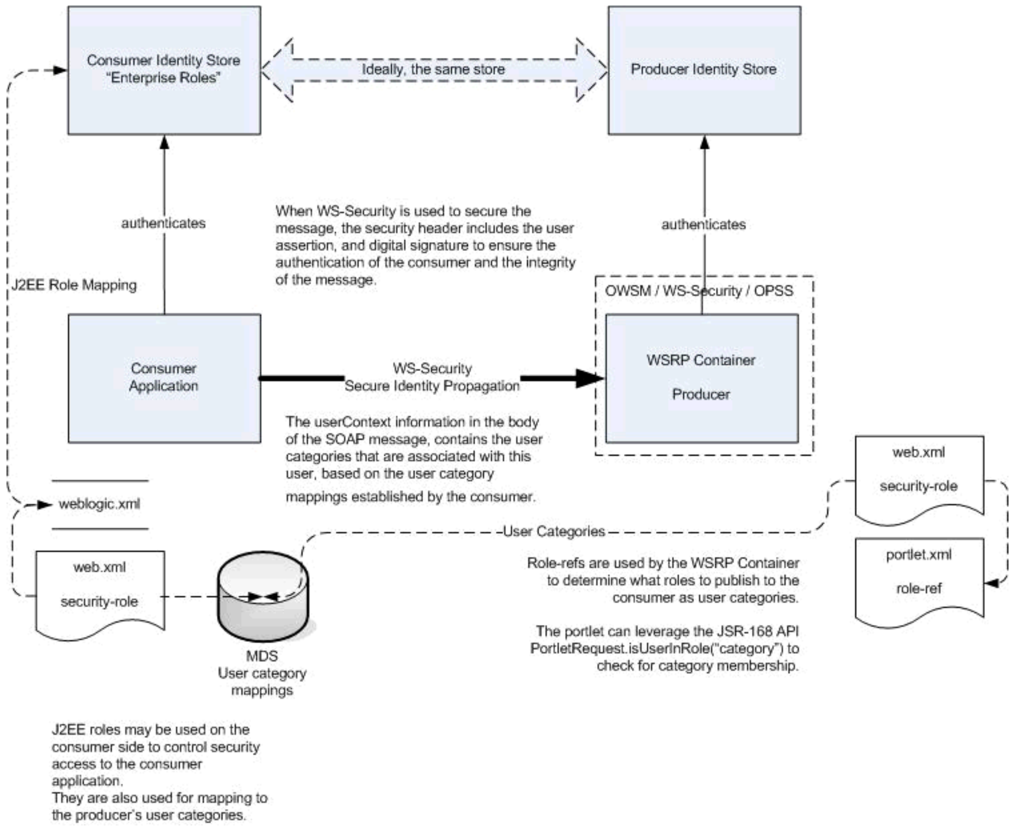
WS-Security implementation is supported by the Oracle WebCenter Suite WSRP container. Other WSRP vendors may also be able to support the WS-Security configuration of Username Token without password, with XML digital signature on the Username Token and the SOAP Message body.

Security Domain Implication

When using secure identity propagation as described in this section, the user name of the user authenticated to the consumer (WebCenter application) is propagated to the producer without any remapping or providing any credentials. There is an inherent assumption that the producer understands this user name and can locate this user in its associated security domain. Consequently, it is highly desirable to ensure that the consumer and producer share the same security provider (identity store) to simplify the management of this configuration.

Figure 24–7 summarizes the overall WSRP portlet security architecture.

Figure 24–7 WSRP Portlet Security Architecture


Consumer Identity Store
"Enterprise Roles"
Ideally, the same store
Producer Identity Store
authenticates
When WS-Security is used to secure the message, the security header includes the user assertion, and digital signature to ensure the authentication of the consumer and the integrity of the message.
authenticates
J2EE Role Mapping
OWSM / WS-Security / OPSS
Consumer Application
WS-Security
Secure Identity Propagation
WSRP Container
Producer
The userContext information in the body of the SOAP message, contains the user categories that are associated with this user, based on the user category mappings established by the consumer.
web.xml
security-role
weblogic.xml
User Categories
web.xml
security-role
Role-refs are used by the WSRP Container to determine what roles to publish to the consumer as user categories.
The portlet can leverage the JSR-168 API PortletRequest.isUserInRole("category") to check for category membership.
portlet.xml
role-ref
MDS
User category mappings
J2EE roles may be used on the consumer side to control security access to the consumer application.
They are also used for mapping to the producer's user categories.

24.12.3 Configuring Security for WSRP Portlets

Before you configure the producer for WS-Security, you must first deploy your standards-compliant portlet producer to the Oracle WebCenter Suite WSRP Container by performing the steps described in Chapter 33, "Testing and Deploying Your Portlets".

After you have deployed the producer, configure the producer for WS-Security by performing the following steps:

- Attaching a policy to the producer endpoint
- Creating the keystores
- Configuring the producer
- Configuring the consumer

These steps are described in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter* .

24.13 Implementing PDK-Java Portlet Security

This section describes the available security services for your PDK-Java portlet.

More On *OTN*

For more detailed information about the PDK classes referred to in this section, see the JavaDoc on OTN by clicking **Java Doc API** on the Portlet Development page available at

`http://www.oracle.com/technology/products/ias/portal/portlet_development_10g1014.html`

24.13.1 Assumptions

The following assumptions are made to perform the tasks in this section:

1. You have followed through and understood Section 29.2, "Creating Java Portlets".
2. You built a portlet using the wizard and successfully added it to a page.

24.13.2 Introduction to PDK-Java Portlet Security Features

This section introduces the major features that are available to secure your PDK-Java portlet producers.

24.13.2.1 Identity Propagation

When a user first logs in, they must enter their password to verify their identity and obtain access.

Once the user is authenticated, the producer code has access to the authenticated user's identity from the `PortletRenderRequest` that is available from the `HttpServletRequest` object as follows:

```
PortletRenderRequest pr = (PortletRenderRequest)request.getAttribute
   (HttpCommonConstants.PORTLET_RENDER_REQUEST);
   String userName = pr.getUser().getName();
```

When using this user identity for sensitive operations, it is important to ensure that you have configured this producer to use basic message authentication to secure the integrity of the identity assertion.

24.13.2.2 Authorization

Authorization determines if a particular user may view or interact with a portlet.

24.13.2.3 Message-level Security

To this point, user authentication and authorization are covered, which do not check the authenticity of messages received by a producer. To completely secure your producers, you should also secure the communication with a producer. If the communication is not secured, then it is possible for someone to imitate an application instance and fool the producer into returning sensitive information. There are three types of communication security:

- **Server Authentication** restricts access to a producer to a small number of recognized computers. 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, then the message is passed to the producer. If not, it is rejected before reaching the producer.

- **Message Authentication** provides consumer authentication and message integrity. It uses a shared key known to the client (WebCenter application) and the producer to digitally sign messages. See Section 24.13.5, "Message Authentication" for more information.
- **Message Encryption** relies on the use of the HTTPS protocol for communication between applications and producers. 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.
- **User Input Escape** causes the application to escape any user input strings and treat them as text only to protect against XSS attacks, where an attacker attempts to pass in malicious scripts through user input forms. For example, if a portlet title is customizable, then an attacker might attempt to pass scripts or commands to the portlet through the title parameter entry. For this reason, the default behavior is to escape user input and thus disable any incoming scripts. See Section 24.13.6, "User Input Escape" for more information.

24.13.3 Single Sign-On

Portlets may act as windows into an application. They display summary information and provide a way to access the full functionality of the application. Portlets display some portions of the application in the WebCenter application and typically enable the user to perform some application tasks.

An application may need to authenticate the user accessing the application through the portlet. The following are the possible application authentication methods:

- External Application. In this case, the Oracle Portal (Oracle Portal) user is different from the application user, but the application user name and password are managed by the Oracle Portal user.
- No Application Authentication. In this case, the communication between producer and consumer is not protected at all.

24.13.3.1 External Application

An external application uses a different authentication server than the WebCenter application. This means that when a user is already logged into the WebCenter application, you want to also log them into the external application without having to type in their user name or password.

Applications that manage the authentication of users can be loosely integrated with the WebCenter Framework if the administrator registers them as external applications. When a user who was previously authenticated by the WebCenter Framework accesses an external application for the first time, WebCenter Framework 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, then the user is prompted for the required information before making the authentication request. When a user supplies a user name and password for an external application, WebCenter Framework maps the new user name and password to the WebCenter application 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:

- 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 Oracle 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 producer in plain text, unless you implement Secure Sockets Layer (SSL).

24.13.3.2 No Application Authentication

The producer trusts the WebCenter application instance sending the request completely. The producer 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 the WebCenter application.

24.13.4 Portlet Security Managers

Portlet security managers may be implemented within a producer to restrict access to portlets depending on the user details. 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 producer 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 producer, then any user name may be passed in, even fictitious, unauthenticated ones.

`AuthLevelSecurityManager` has access to the following information about authorization level:

- Strongly authenticated.

  The user has signed into the WebCenter aplication, and requested the portlet in the context of that session.

- Public or not authenticated.

  The user has not logged in within the context of the current 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 before 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 producer.

For more information about the syntax of `provider.xml`, see the producer JavaDoc on OTN:

`http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html`

24.13.4.1 Implementing Your Own Security Manager

If your portlet requires special security arrangements which are not provided by the implementations shipped with the PDK, then you will need to supply your own custom `PortletSecurityManager` controller class. To do this, extend the `oracle.portal.provider.v2.security.PortletSecurityManager` class and supply implementations for the two methods specified by the interface. Then replace the class attribute of the `securityManager` controller element in the XML producer definition with you new class name and configure child elements appropriately.

24.13.5 Message Authentication

Message authentication uses a digital signature. The signature is generated using a Hashed Message Authentication Code (HMAC) algorithm and is based on user information, the shared key, and a UTC (Universal Time, Coordinated) timestamp. The producer authenticates the message using its own copy of the shared key. This technique can be used in Secure Sockets Layer (SSL) communication with a producer instead of client certificates.

For caching purposes, show request signatures are calculated each time a session is set up so producers using message authentication must always have Producer Sessions enabled. This also means that there is a trade-off between performance and security. Shorter session timeouts mean less chance of a message being re-sent illegally, but there is a performance overhead associated with reestablishing a provider session.

A single producer 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, then the producer administrator has to create a new key and distribute it to all of the application clients, who then must update their producer definitions. The way around this problem is to deploy different producer services, specifying a unique shared key for each service. Each producer service has its own deployment properties file so that each service is configured independently of the others. The overhead of deploying multiple producer services within the same producer adapter is relatively small.

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, then this should be used in conjunction with SSL to encrypt the message.

The advantage of message authentication is as follows:

- Ensures that the message received by a producer comes from a legitimate WebCenter application instance.

The disadvantages of message authentication are as follows:

- Causes administration problems if a producer serves more than one portal.

- Entails performance implications if made very secure by having a short session timeout.

24.13.6 User Input Escape

By accepting user input without escaping it to text, you run the risk of an XSS attack, where an attacker attempts to pass in malicious scripts through user input forms. For example, if a portlet title is customizable, then an attacker might attempt to pass scripts or commands to the portlet through the title string. PDK-Java provides the following features to ensure that you can protect your portlets from such attacks:

- Default Container Encoding
- Escape Methods

24.13.6.1 Default Container Encoding

To prevent any script inside a portlet title from being executed, the framework default container renderer class encodes any script characters. This default behavior is controlled through a JNDI variable, `escapeStrings`. When set to `true`, the markup tags in portlet titles are rendered as visible tag characters. For example, a title customization of `<i>title</i>` will be rendered as `<i>title</i>` not *title*. This mode is secure, but, if it is not the desired behavior, then you can set `escapeStrings` to `false` for that producer.

`escapeStrings` applies to all logical producers within a producer. You can set the value of `escapeStrings` from the Fusion Middleware Control Console as you would any other JNDI variable. See Section 30.2.4.2, "Setting JNDI Variable Values" for more information.

24.13.6.2 Escape Methods

If you have code that renders customized values, then you need to ensure that you escape those input values appropriately to avoid XSS attacks. This requirement applies to code for rendering pages in any mode. PDK-Java supplies two new static methods for this purpose. They are in the Java class `oracle.portal.provider.v2.url.UrlUtils`, and can be described as follows:

- `public static escapeString(`*`string_text`*`)` escapes any script characters in a given string. For example, less than < becomes `&lt`. This method is unaffected by the `escapeStrings` JNDI variable and is the secure, recommended method to use.
- `public static escapeStringByFlag(`*`string_text`*`)` escapes any script characters in a given string. This method is controlled by the `escapeStrings` JNDI variable and is therefore less secure and not the recommended method to use.

For example:

```
title = UrlUtils.escapeString(data.getPortletTitle());
```

25 Testing and Deploying Your WebCenter Application

This chapter describes how to test custom WebCenter applications on Integrated WebLogic Server (Integrated WLS Server) that comes packaged with Oracle JDeveloper. This chapter also describes how to deploy applications using JDeveloper, to Oracle WebLogic Managed Server instances configured to test application deployment or host applications for the production purpose. For information about deploying custom WebCenter applications through other tools, such as, Oracle Enterprise Manager Fusion Middleware Control, Oracle WebLogic Administration Console, and WLST commands, see the chapter, "Deploying WebCenter Applications" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

This chapter includes the following sections:

25.1 Introduction to Oracle WebLogic Servers

This section provides an overview of Integrated WLS Server and Oracle WebLogic Managed Server (managed server). Using Oracle JDeveloper, you can test your applications on Integrated WLS Server and deploy applications to managed servers that reside outside Oracle JDeveloper.

Integrated WLS Server

Integrated WLS Server (Default Server) comes packaged with Oracle JDeveloper. The Default Server connection appears as **IntegratedWLSConnection** in **Application Server** under **IDE Connections** in the Resource Palette. The Default Server is meant for quick and easy testing of applications in development environments.

When the Default Server is started, an instance of the server is created once for all applications. You can run multiple applications simultaneously. When an application is run on the Default Server, a server instance, named after the application, is created.

Testing custom WebCenter applications on Integrated WLS Server is easy because:

- Deployment is optimized and the requirement to zip and copy files is eliminated.
- Most files run directly from project directories. This feature enables you to make selected changes and refresh these changes while the application is running. For

example, any changes made to a JSPX page can be refreshed immediately, without redeploying the application.

- There is no requirement to undeploy applications after testing and debugging. Applications are undeployed when the Default Server becomes unavailable.

Oracle WebLogic Managed Server

A managed server hosts applications and the resources needed by those applications. A domain, which is a logically related group of Oracle WebLogic Server resources, can have any number of managed servers. An Administration Server manages these servers. Managed servers can be deployed to hold production applications or test or both.

25.2 Testing a Custom WebCenter Application on Integrated WLS Server

When you run an application on the Default Server, a server instance named after the application is created. You can watch the progress of each application in the Run Manager panel. The Run Manager panel also lets you stop the Default Server instances.

This section includes the following:

- Section 25.2.1, "How to Run a Custom WebCenter Application on Integrated WLS Server"
- Section 25.2.2, "What Happens When You Run a WebCenter Application on Integrated WLS Server"

25.2.1 How to Run a Custom WebCenter Application on Integrated WLS Server

Since Integrated WLS Server is preconfigured to run applications in Oracle JDeveloper, you are not required to create deployment profiles.

To run a custom WebCenter application, right-click the `.jspx` page in the project folder and select **Run**. Clicking **Run** triggers packaging and deployment of your custom WebCenter application on an instance of Integrated WLS Server named after the application.

You can see the DefaultServer Log window to know how the activity is progressing.

Note: When the Integrated WLS Server instance stops, the application is undeployed, and therefore, becomes unavailable.

For a more persistent testing scenario, you can deploy your application to Integrated WLS Server by right-clicking the application and then selecting **Deploy**. This deploys the application to the Default Server instance and it is always available when the Default Server is running. If you choose this method, then you must first create deployment profiles, as described in Section 25.3.2, "How to Create Deployment Profiles". If you deploy your application to Integrated WLS Server, then the Deployment Configuration dialog displays to let you configure and customize deployment settings. The file system MDS repository pre-created by JDeveloper displays in the **Repository Name** field.

25.2.2 What Happens When You Run a WebCenter Application on Integrated WLS Server

Oracle WebCenter temporarily modifies `mds-config` in the `adf-config.xml` file to reconfigure the directory paths for the running application. It provides the menu option **Build** > **Clean Runtime MDS Customizations** that lets you remove view-oriented customizations made during testing.

Secure attributes and credentials are migrated by default when running a custom WebCenter application on Integrated WLS Server.

Integrated WLS Server creates an archive directory of your custom WebCenter application at *`system_directory`*`/o.j2ee/drs`.

25.3 Deploying a Custom WebCenter Application to an Oracle WebLogic Managed Server Instance

This section describes how to deploy a custom WebCenter application to an Oracle WebLogic Managed Server instance using JDeveloper. From JDeveloper, you can deploy an application for the testing purpose, or to hold the application in a production environment.

For information about deploying custom WebCenter applications through Fusion Middleware Control, Oracle WebLogic Administration Console, and WLST commands, see the chapter, "Deploying WebCenter Applications" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

This section includes the following:

- Section 25.3.1, "What You May Need to Know About Database Connections and Application Security Migration When Deploying Custom WebCenter Applications"
- Section 25.3.2, "How to Create Deployment Profiles"
- Section 25.3.3, "How to Create and Provision an Oracle WebLogic Managed Server Instance"
- Section 25.3.4, "How to Create and Register the Metadata Service Repository"
- Section 25.3.5, "How to Create a WebLogic Managed Server Connection"
- Section 25.3.6, "How to Deploy a Custom WebCenter Application to an Oracle WebLogic Managed Server Instance"
- Section 25.3.7, "What Happens When You Deploy A WebCenter Application to an Oracle WebLogic Managed Server Instance"

25.3.1 What You May Need to Know About Database Connections and Application Security Migration When Deploying Custom WebCenter Applications

This section describes database connections- and application security-related points that you should consider before deploying any custom WebCenter applications.

Custom WebCenter applications use database connection types: **JDBC DataSource** and **JDBC URL**. Depending upon the database connections used by an application and tools to be used to deploy or test an application, you can choose which approach out of those discussed in this section is best suited to set up or migrate connection credentials in development and production modes.

This section also describes security migration for applications that use database connections specifying passwords and external applications with shared or public credentials.

This section includes the following sub sections:

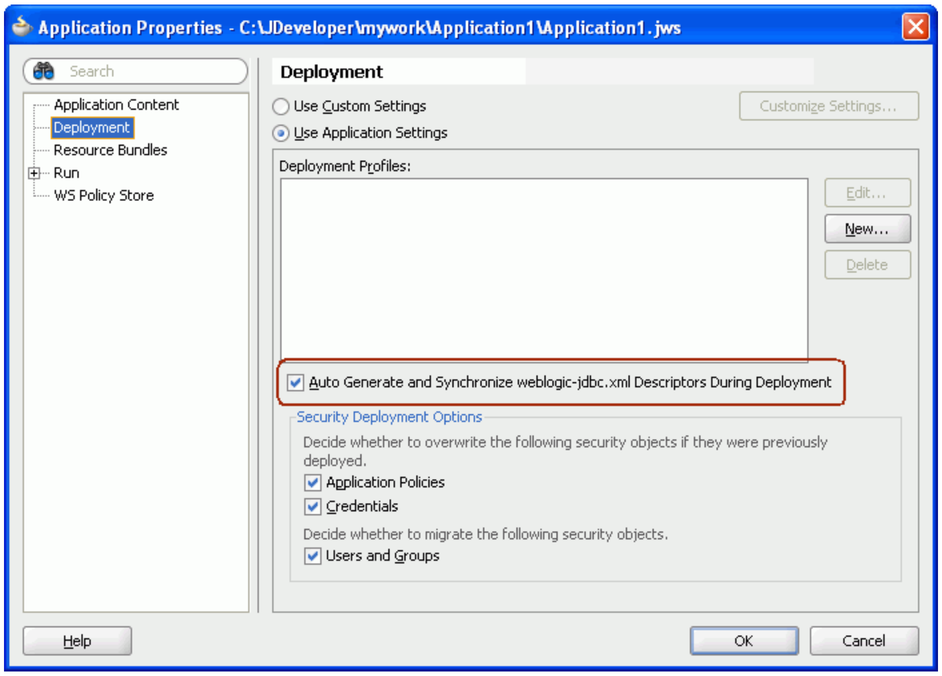
- Section 25.3.1.1, "What You May Need to Know About JDBC DataSource"
- Section 25.3.1.2, "What You May Need to Know About JDBC URL"
- Section 25.3.1.3, "What You May Need to Know About Application Security Migration"

25.3.1.1 What You May Need to Know About JDBC DataSource

This section describes what types of JDBC data sources can be created, what these data sources consist of, and which type to choose and when.

- *Application-level data source with password indirection*: JDeveloper generates this data source when you run an application on Integrated WLS Server. That is, to generate application-level data source with password indirection, the **Auto Generate and Synchronize weblogic-jdbc.xml Descriptors During Deployment** checkbox is selected by default in the Application Properties dialog, as shown in Figure 25–1. In this case, JDeveloper uses MBean calls to pass the database passwords to the Integrated WLS Server.

Figure 25–1 Auto Generate and Synchronize weblogic-jdbc.xml Descriptors During Deployment Checkbox


Application Properties - C:\JDeveloper\mywork\Application1\Application1.jws

Deployment

Use Custom Settings
[✓ Use Application Settings]

Deployment Profiles:

[✓] Auto Generate and Synchronize weblogic-jdbc.xml Descriptors During Deployment

Security Deployment Options

Decide whether to overwrite the following security objects if they were previously deployed.

[✓] Application Policies

[✓] Credentials

Decide whether to migrate the following security objects.

[✓] Users and Groups

To generate an application-level indirection data source, JDeveloper does the following:

– Generates a `*-jdbc.xml` file for each connection stored in the Application Resources.

– Sets the indirect password attribute in the `jdbc.xml` file:

```
<jdbc-driver-params>
```

```
<use-password-indirection>true</use-password-indirection>
</jdbc-driver-params>
```

– Adds each `*-jdbc.xml` file as a module in the `weblogic-application.xml` file.
– Adds a resource reference to each JDBC JNDI name in the `web.xml` file, if this file exists.

If you choose to generate application data source with password indirection, then you must add credential mappings using Oracle WebLogic Administration Console to be able to activate the application. For more information on adding credential mappings, see the section "Creating a JDBC Data Source" in *Oracle Fusion Middleware Configuring and Managing JDBC for Oracle WebLogic Server*.

Note: Do not use WLST commands to deploy applications that use application-level data sources with password indirection, because WLST cannot set password indirection credential mappings on the server.

- *Global data source*: Choose this type of data source if you plan to deploy applications through Fusion Middleware Control, Oracle WebLogic Administration Console. This data source is also recommended when you deploy an application to an EAR file and then deploy the EAR file through WLST (`wldeployer`) command to a managed server running in production mode.

  Deselect the **Auto Generate and Synchronize weblogic-jdbc.xml Descriptors During Deployment** checkbox so that JDeveloper does not create an application-level data source with password indirection. Instead, create the global data source using Oracle WebLogic Administration Console after deploying an application to a managed server in production mode. For more information on creating global data sources, see the section "Creating a JDBC Data Source" in *Oracle Fusion Middleware Configuring and Managing JDBC for Oracle WebLogic Server*.

25.3.1.2 What You May Need to Know About JDBC URL

For applications that use the JDBC URL connection type, a credential store consisting of `jps-config.xml` and `cwallet.sso` files is created for each application. These files are stored in the `META-INF` directory of the application.

25.3.1.3 What You May Need to Know About Application Security Migration

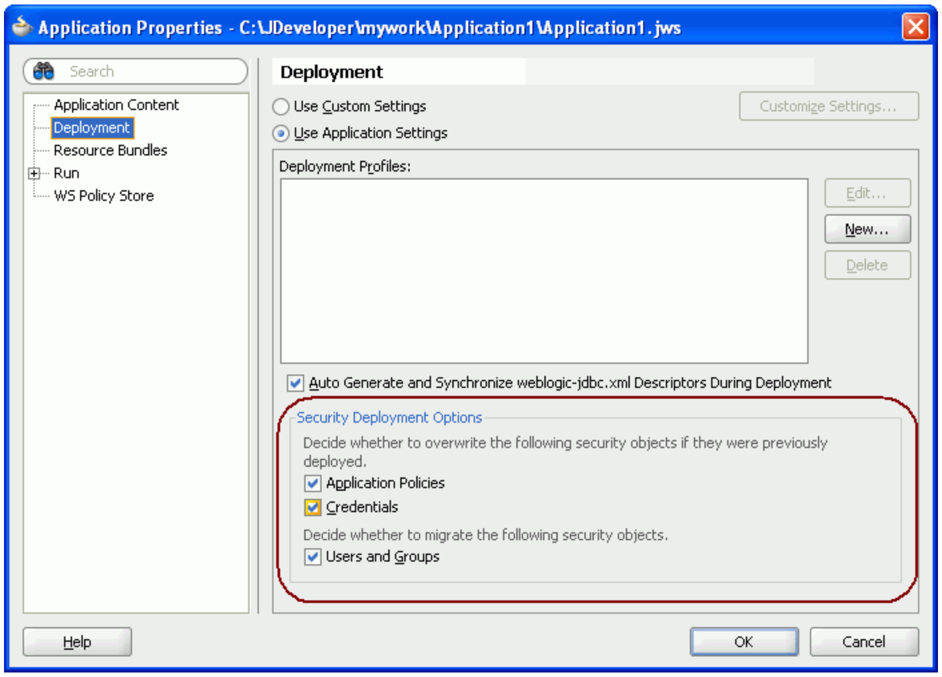
The following is applicable if you configured security in your application using the Configure ADF Security wizard, or the application contains connections with secure attributes, for example, database connection password, external applications with shared or public credentials:

- Credential Migration
- Identity Migration
- Policy Migration

Credential Migration

If you do not intend to migrate credentials, then deselect the **Credentials** checkbox in the **Security Deployment Options** section in the Application Properties dialog (see Figure 25–2)

Figure 25–2 Security Deployment Options


Application Properties - C:\JDeveloper\mywork\Application1\Application1.jws

Search

Application Content
Deployment
Resource Bundles
Run
WS Policy Store

Deployment

Use Custom Settings
Customize Settings...
Use Application Settings

Deployment Profiles:

Edit...
New...
Delete

[✓] Auto Generate and Synchronize weblogic-jdbc.xml Descriptors During Deployment

Security Deployment Options

Decide whether to overwrite the following security objects if they were previously deployed.

[✓] Application Policies

[✓] Credentials

Decide whether to migrate the following security objects.

[✓] Users and Groups

Help
OK
Cancel

If you retain the default selection in the **Security Deployment Options** section, then the credential migration will behave as follows, depending on if the WebLogic domain is in development or production mode:

- To enable credential migration for an application to be deployed in development mode, start the managed server with `-Djps.app.credential.overwrite.allowed=true`, by adding the following option to the `setDomainEnv.cmd` or `setDomainEnv.sh` file located in *`domain`*`\bin`:

  ```
  set EXTRA_JAVA_PROPERTIES=-Djps.app.credential.overwrite.allowed=true %EXTRA_
  JAVA_PROPERTIES%
  ```

- When you deploy a custom WebCenter application to a managed server running in production mode, secure attributes of the connections packaged within your application and any shared or public credentials specified for external applications are not migrated to the domain-level credential store. This is because secure properties are likely to be different between development and production environments. Therefore, you must reconfigure secure attributes of connections and shared or public credentials for external applications using WLST commands or Fusion Middleware Control, as described in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

  For more information on credential migration behavior, see the section "Configuring the Credential Store" in *Oracle Fusion Middleware Security Guide*.

Identity Migration

In Figure 25–2, in the **Security Deployment Options** section, the **Users and Groups** checkbox is selected by default. This indicates that the users and groups created in the application's `jazn-data.xml` file will be migrated to the identity store configured on the WebLogic Managed Server, if the authenticator configured in the domain allows creation of users and groups. It is recommended that you deselect this option because, typically, users and groups in the managed server's identity store should be used to access the application.

Policy Migration

The **Application Policies** checkbox must always be selected.

25.3.2 How to Create Deployment Profiles

To deploy your application to a WebLogic Server that resides outside JDeveloper, you must first create deployment profiles. A deployment profile packages or archives a custom WebCenter application and associated files so that the application can be deployed to an Oracle WebLogic Managed Server instance. Deployment profiles are created at project and application level. At the project level, a WAR profile is created. This file packages a Web application as a Web Application Archive (WAR) file. It contains the projects to be deployed and any associated files, such as, pages, Java classes, `weblogic.xml`, `web.xml`, and so on. At the application level, an EAR file is created. This file packages the complete application as an Enterprise Archive (EAR) file. The EAR file contains application artifacts such as, `adf-config.xml`, `connections.xml`, `weblogic-application.xml`, `jazn-data.xml`, metadata archive files (`.mar`), and all project-related WAR files. For custom WebCenter applications, the MAR is automatically generated and included in the EAR file, if there is MDS metadata to be packaged.

Note: You can deploy Oracle ADF applications, such as custom WebCenter applications, only as EAR files. Therefore, while creating deployment profiles, ensure that the WAR and MAR files of the application are included in the application's EAR file.

To create a project-level WAR deployment profile:

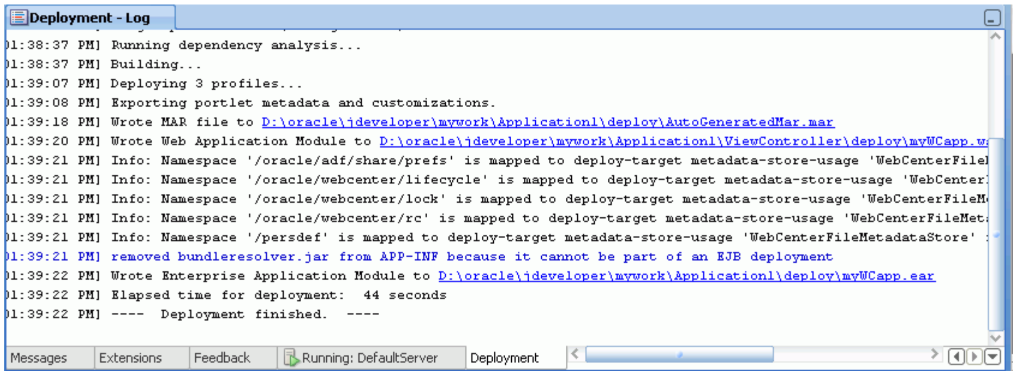
1. Right-click **ViewController** and select **New**.
2. In the New Gallery, expand **General**, select **Deployment Profile**, then **WAR File**, and then click **OK**.
3. In the Create Deployment Profile -- WAR File dialog, enter a name for your deployment profile and click **OK**.
4. In the WAR Deployment Profile Properties dialog, select the **Specify Java EE Web Context Root** option and enter a meaningful context root. Then click **OK**.
5. In the Project Properties dialog, click **OK**.

To create an application-level EAR deployment profile:

1. From the File menu, choose **New**.
2. In the New Gallery, expand **General**, select **Deployment Profile**, then **EAR File**, and then click **OK**.
3. In the Create Deployment Profile -- EAR File dialog, enter a name for your deployment profile and click **OK**.
4. In the Edit EAR Deployment Profile Properties dialog, select **Application Assembly**, then select the project WAR you want to make available in the project file. Click **OK**.
5. In the Application Properties dialog, click **OK**.

6. In the Application Navigator, right-click the application name, choose **Deploy**, *deployment profile name*, **to**, and then choose **to EAR file**. This creates the `.ear` file in the **deploy** folder located in `JDEV_HOME\mywork\application_name\deploy\`, as shown in Figure 25–3.

Figure 25–3 EAR Status in Deployment - Log



```
01:38:37 PM] Running dependency analysis...
01:38:37 PM] Building...
01:39:07 PM] Deploying 3 profiles...
01:39:08 PM] Exporting portlet metadata and customizations.
01:39:18 PM] Wrote MAR file to D:\oracle\jdeveloper\mywork\Application1\deploy\AutoGeneratedMar.mar
01:39:20 PM] Wrote Web Application Module to D:\oracle\jdeveloper\mywork\Application1\ViewController\deploy\myWCapp.wa
01:39:21 PM] Info: Namespace '/oracle/adf/share/prefs' is mapped to deploy-target metadata-store-usage 'WebCenterFile
01:39:21 PM] Info: Namespace '/oracle/webcenter/lifecycle' is mapped to deploy-target metadata-store-usage 'WebCenter
01:39:21 PM] Info: Namespace '/oracle/webcenter/lock' is mapped to deploy-target metadata-store-usage 'WebCenterFileM
01:39:21 PM] Info: Namespace '/oracle/webcenter/rc' is mapped to deploy-target metadata-store-usage 'WebCenterFileMet
01:39:21 PM] Info: Namespace '/persdef' is mapped to deploy-target metadata-store-usage 'WebCenterFileMetadataStore'
01:39:21 PM] removed bundleresolver.jar from APP-INF because it cannot be part of an EJB deployment
01:39:22 PM] Wrote Enterprise Application Module to D:\oracle\jdeveloper\mywork\Application1\deploy\myWCapp.ear
01:39:22 PM] Elapsed time for deployment:  44 seconds
01:39:22 PM] ----  Deployment finished.  ----
```

25.3.3 How to Create and Provision an Oracle WebLogic Managed Server Instance

To deploy a custom WebCenter application to a WebLogic Managed Server instance, you must first create a server instance and provision it with a required set of shared libraries, as described in the section "Creating and Provisioning a WebLogic Managed Server Instance" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

25.3.4 How to Create and Register the Metadata Service Repository

After creating the WebLogic Managed Server instance, you must create and register a Metadata Service Repository (MDS) schema for the application on the WebLogic Domain's Administration Server instance, as described in the section "Creating and Registering the Metadata Service (MDS) Repository" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

25.3.5 How to Create a WebLogic Managed Server Connection

From Oracle JDeveloper, you can deploy your applications to Oracle WebLogic Managed Server instances that reside outside JDeveloper. To do this, you must first create a connection to the server instance to which you want to deploy your application.

Before you create a connection to an Oracle WebLogic Managed Server instance, ensure that the server instance is up and running.

To create a WebLogic Managed Server connection:

1. From the File menu, select **New**.
2. In the New Gallery, expand **General**, select **Connections** and then **Application Server Connection**, and click **OK**.
3. In the Create Application Server Connection wizard, on Step 1, enter a name for the new connection, for example, `WebCenterServer`. Then click **Next**.
4. On Step 2, specify user name and password for authentication. Click **Next**.

5. On Step 3, enter the host name of the WebLogic Managed Server instance, for example, `webcenter.myserver.oracle.com` and the port number, for example, `7001`.
6. In the WLS Domain, specify the name of the domain in which the WebLogic Managed Server instance is created, for example, `wl_server`. Click **Next**.
7. On Step 4, click **Test Connection**. The connection is set up if the test is successful.

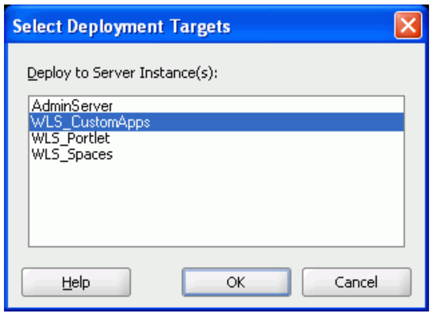
25.3.6 How to Deploy a Custom WebCenter Application to an Oracle WebLogic Managed Server Instance

After you have created the deployment profiles and server connection, you can deploy your custom custom WebCenter application to this server instance.

To deploy a WebCenter application:

1. In the Application Navigator, open the application to be deployed.
2. From the Application Navigator, select the application, choose **Deploy**, *deployment profile name*, **to**, and then choose the *connection name*.
3. In the Select Deployment Target dialog, as shown in Figure 25–4, select the name of the managed server, for example, `WLS_CustomApps`, and click **OK**.

Figure 25–4 Select Deployment Targets dialog

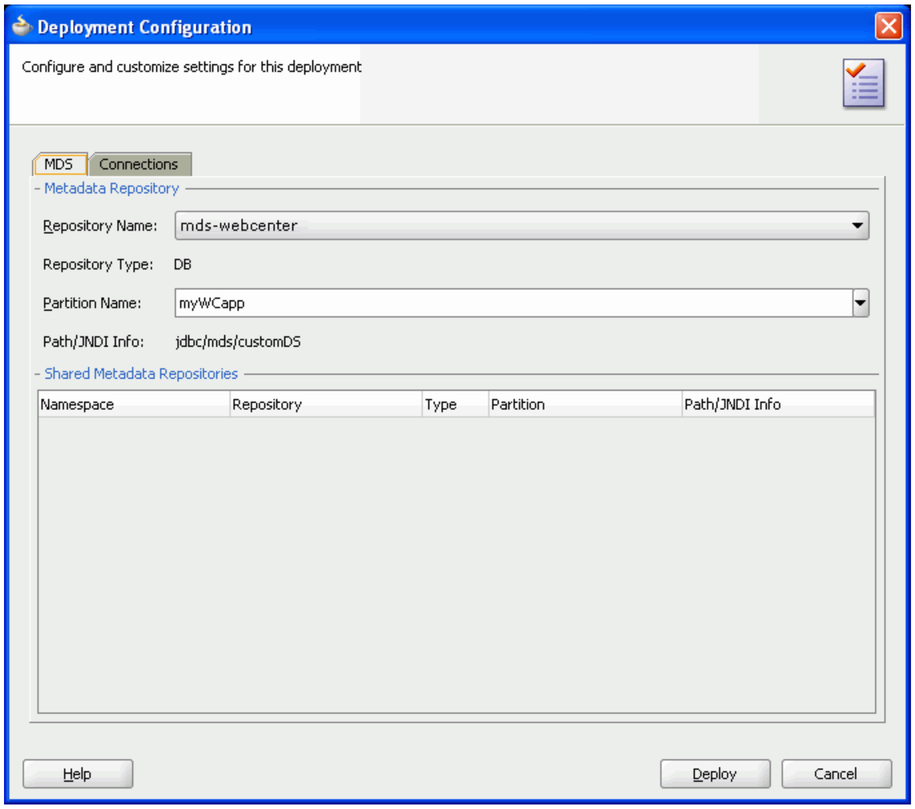

Select Deployment Targets

Deploy to Server Instance(s):

AdminServer
WLS_CustomApps
WLS_Portlet
WLS_Spaces

4. In the Deployment Configuration dialog, under the **MDS** tab, in the **Repository Name**, select the metadata repository to be used by the application you are deploying, as shown in Figure 25–5.

Figure 25–5 Deployment Configuration dialog


Deployment Configuration
Configure and customize settings for this deployment
MDS
Connections
Metadata Repository

Repository Name: mds-webcenter

Repository Type: DB

Partition Name: myWCapp

Path/JNDI Info: jdbc/mds/customDS

Shared Metadata Repositories

Namespace	Repository	Type	Partition	Path/JNDI Info

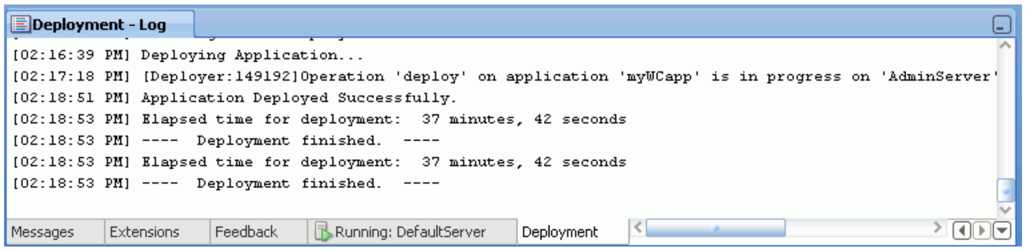
5. In the **Partition Name** field, specify the application's partition name. Each application is recommended to have its own partition. The application name can be used as the partition name.

6. Under the **Connections** tab, modify the connections packaged with the custom WebCenter application being deployed.

> **Note:** In the **Connections** tab, you cannot change secure properties of the connections. It is recommended that you modify these connections post-deployment using Fusion Middleware Control. For information, see the chapter "Managing Services" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

7. Click **Deploy**. The Deployment - Log displays the deployment status, as shown in Figure 25–6.

Figure 25–6 Deployment - Log



```
[02:16:39 PM] Deploying Application...
[02:17:18 PM] [Deployer:149192]Operation 'deploy' on application 'myWCapp' is in progress on 'AdminServer'
[02:18:51 PM] Application Deployed Successfully.
[02:18:53 PM] Elapsed time for deployment:  37 minutes, 42 seconds
[02:18:53 PM] ----  Deployment finished.  ----
[02:18:53 PM] Elapsed time for deployment:  37 minutes, 42 seconds
[02:18:53 PM] ----  Deployment finished.  ----
```

25.3.7 What Happens When You Deploy A WebCenter Application to an Oracle WebLogic Managed Server Instance

During the deployment to an Oracle WebLogic Managed Server instance, the EAR file is generated. The EAR file packages the metadata archive (MAR file), which includes the metadata content to be deployed to an MDS repository. Additionally, the `adf-config.xml` file is reconfigured with a modified `mds-config` for the target deployment environment. Application-wide features, security, caching, and change persistence remain unchanged. Properties for other Oracle components, if any, are also configured in this file. Similarly, JSF and JSTL shared libraries are added in the `weblogic.xml` file during packaging.

25.4 Transporting Customizations Between Environments

When migrating a deployed application to a new environment, post-deployment customizations made to pages, WebCenter Web 2.0 Services, and portlets (PDK-Java and WSRP version 2 producers) must be migrated too. Export and import utilities are available to assist this process. For more information, see the chapter, "Managing Backup, Recovery, Export, and Import of WebCenter" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

26 Working Productively in Teams

This chapter contains information about managing files in Oracle JDeveloper. It includes information about source control systems, namely CVS and Subversion.

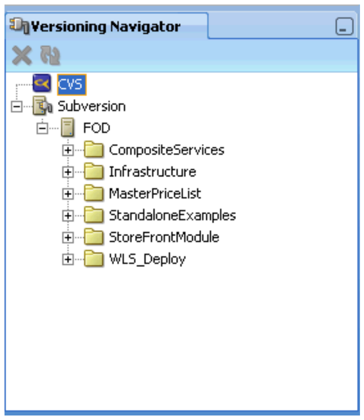
This chapter includes the following sections:

26.1 Enabling Source Control on WebCenter Applications

You can use CVS or Subversion for source control in a WebCenter application, similar to the way you would use a source control system in any other development environment.

There are two ways to use CVS or Subversion: In Oracle JDeveloper, either click the **Versioning** menu or use the Versioning Navigator (click **View - Versioning Navigator**). The Versioning Navigator is shown in Figure 26–1.

Figure 26–1 Versioning Navigator



For detailed information about CVS and Subversion, see the Oracle JDeveloper online help system and the "Using a Source Control System" section in the *Oracle Application Development Framework Developer's Guide*.

26.1.1 Importing Application Files into CVS

If you do not have your application files source controlled in CVS, then you must import them into CVS before you can get started.

To import application files into CVS:

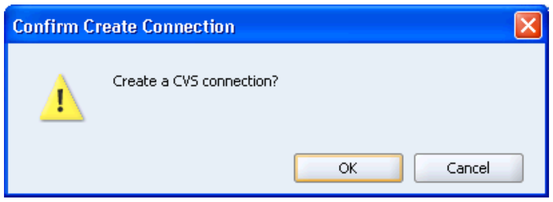
1. In Oracle JDeveloper, from the **Versioning** menu, select **CVS** and then **Check Out Module**.

   If you have a CVS connection, then you are taken directly to step 2. If you do not have a CVS connection, then you are prompted to create a connection, as follows:

   > **Note:** An alternative way to create a CVS connection is to right-click **CVS** in the Versioning Navigator and select **New CVS Connection**. Any existing CVS connection is shown in the Versioning Navigator.

   a. If the Confirm Create Connection dialog appears, then click **OK** (Figure 26–2).

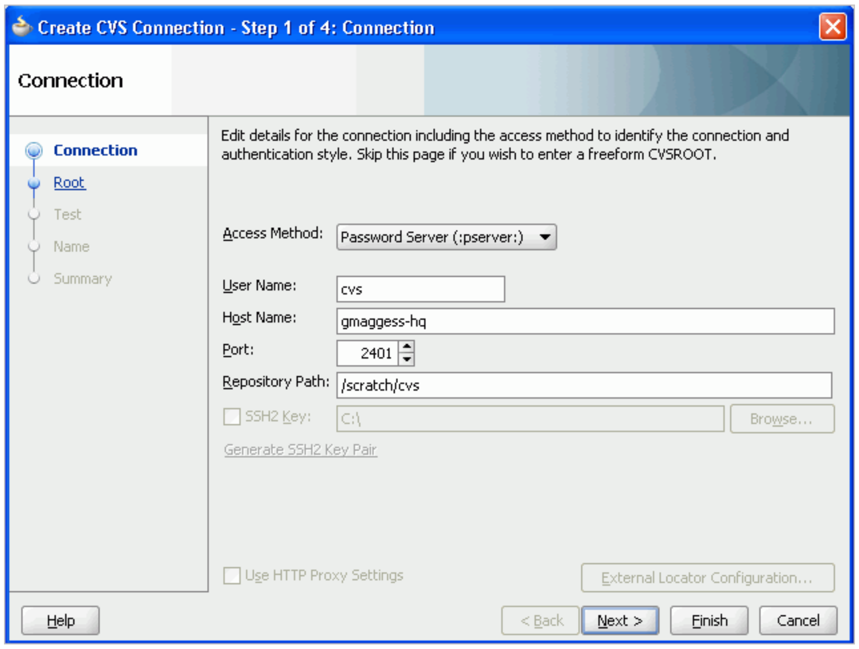
Figure 26–2 Confirm Create Connection Dialog



Create a CVS connection?

   b. In the Create CVS Connection wizard, on the Connection page, enter your connection information for CVS, then click **Next** (Figure 26–3).

Figure 26–3 Create CVS Connection - Connection

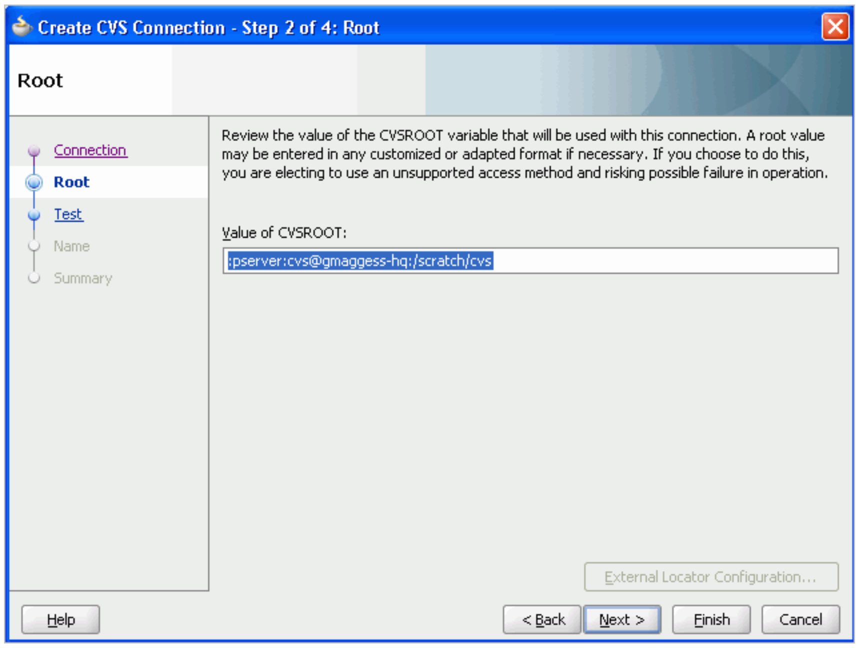


Edit details for the connection including the access method to identify the connection and authentication style. Skip this page if you wish to enter a freeform CVSROOT.

Access Method: Password Server (:pserver:)
User Name: cvs
Host Name: gmaggess-hq
Port: 2401
Repository Path: /scratch/cvs
SSH2 Key: C:\
Use HTTP Proxy Settings

   c. On the Root page (Figure 26–4), in the **Value of CVSROOT** field, enter a value for the CVSROOT. Click **Next**.

Figure 26–4 Create CVS Connection - Root


Create CVS Connection - Step 2 of 4: Root
Root
Connection
Root
Test
Name
Summary
External Locator Configuration...
Help
< Back
Next >
Finish
Cancel

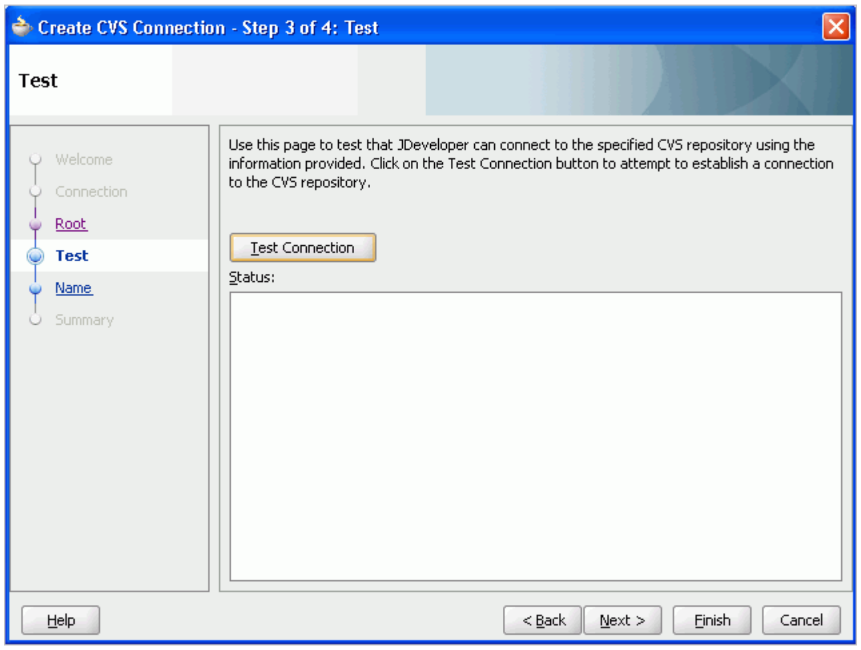
Review the value of the CVSROOT variable that will be used with this connection. A root value may be entered in any customized or adapted format if necessary. If you choose to do this, you are electing to use an unsupported access method and risking possible failure in operation.

Value of CVSROOT:

:pserver:cvs@gmaggess-hq:/scratch/cvs

d. On the Test page (Figure 26–5), click **Test Connection**. The Log In to CVS dialog displays, as shown in Figure 26–6.

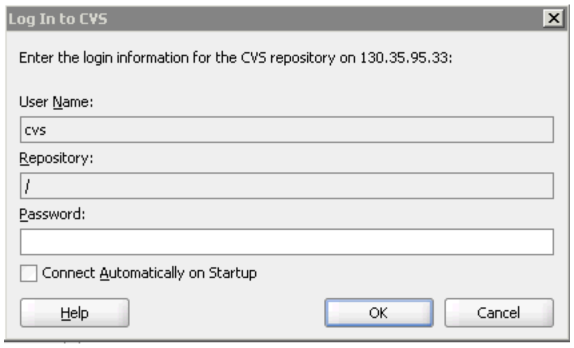
Figure 26–5 Create CVS Connection - Test Connection



Use this page to test that JDeveloper can connect to the specified CVS repository using the information provided. Click on the Test Connection button to attempt to establish a connection to the CVS repository.

Status:

Figure 26–6 Log In to CVS Dialog



Enter the login information for the CVS repository on 130.35.95.33:

User Name: cvs

Repository: /

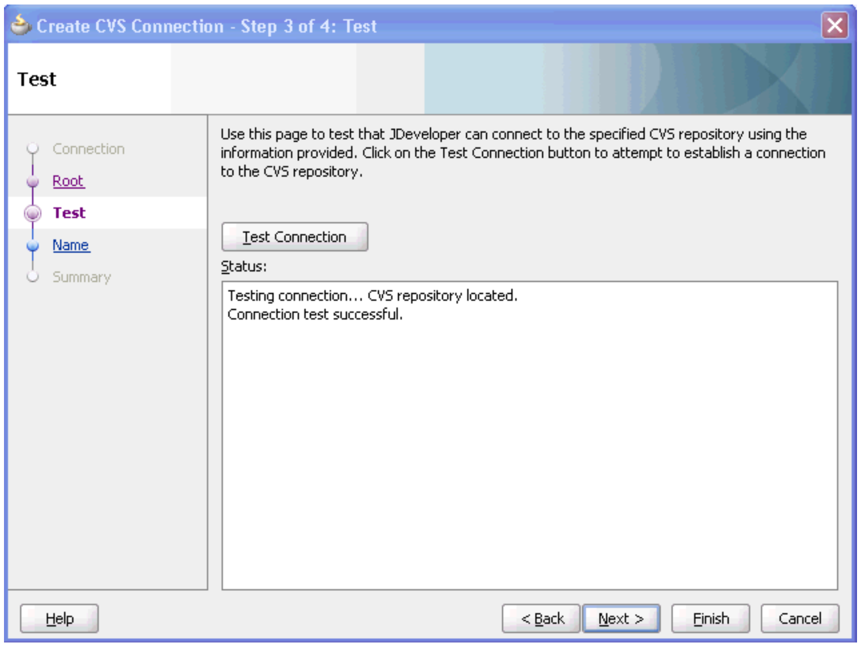
Password:

☐ Connect Automatically on Startup

e. Enter your password, and then click **OK**.

f. If the test is successful, then you should see something similar to Figure 26–7. Click **Next**.

Figure 26–7 Create CVS Connection - Test Connection Success



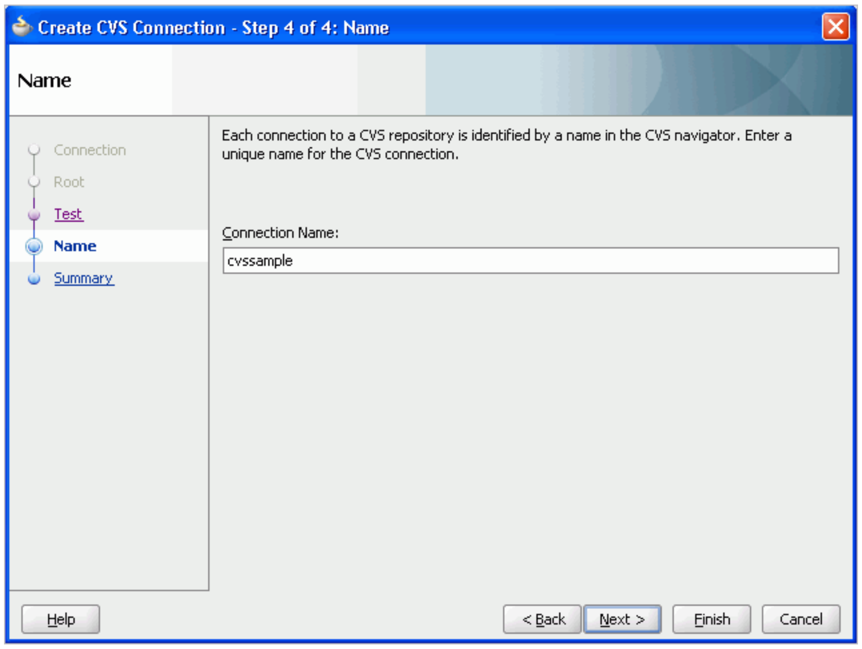
Use this page to test that JDeveloper can connect to the specified CVS repository using the information provided. Click on the Test Connection button to attempt to establish a connection to the CVS repository.

Status:

```
Testing connection... CVS repository located.
Connection test successful.
```

g. On the Name page, enter a value for **Connection Name**, as shown in Figure 26–8. Click **Next**.

Figure 26–8 Create CVS Connection - Name

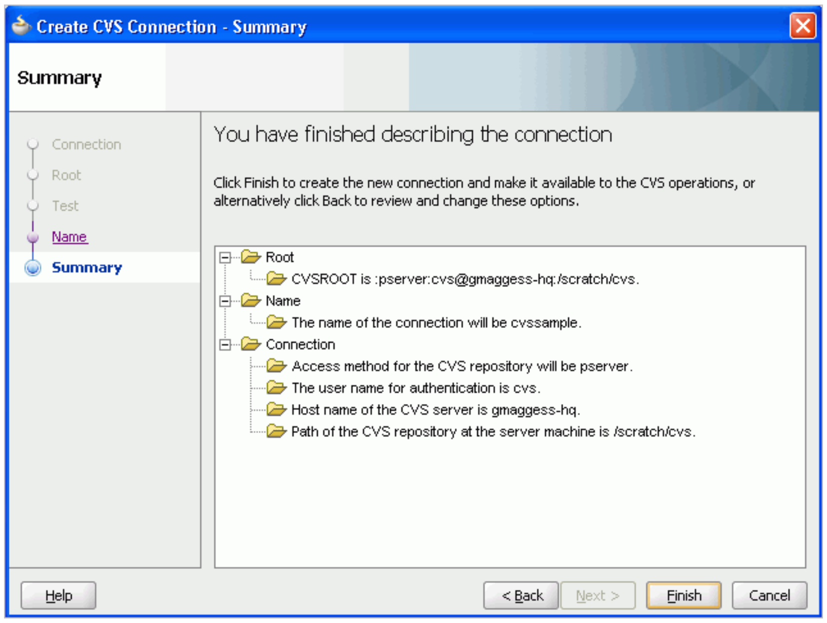


Each connection to a CVS repository is identified by a name in the CVS navigator. Enter a unique name for the CVS connection.

Connection Name: cvssample

h. The Summary page displays with the connection information. Click **Finish** (Figure 26–9).

Figure 26–9 Create CVS Connection - Summary



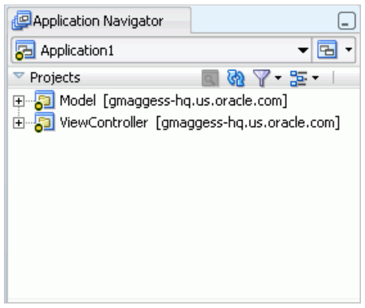
You have finished describing the connection

Click Finish to create the new connection and make it available to the CVS operations, or alternatively click Back to review and change these options.

- Root
  - CVSROOT is :pserver:cvs@gmaggess-hq:/scratch/cvs.
- Name
  - The name of the connection will be cvssample.
- Connection
  - Access method for the CVS repository will be pserver.
  - The user name for authentication is cvs.
  - Host name of the CVS server is gmaggess-hq.
  - Path of the CVS repository at the server machine is /scratch/cvs.

2. On the **Check Out from CVS** page, enter the application path.
3. Click **OK** to execute the import operation. You can see the connection now in the Application Navigator, as shown in Figure 26–10.

Figure 26–10 Application Navigator with CVS Connection



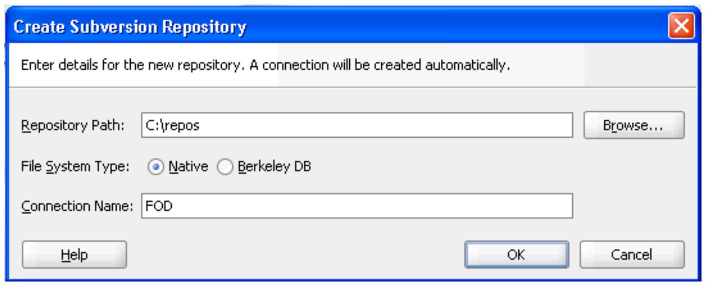
Model [gmaggess-hq.us.oracle.com]
ViewController [gmaggess-hq.us.oracle.com]

26.1.2 Creating a Subversion Repository

To create a Subversion repository:

1. From the **Versioning** menu, choose Subversion, and then **Create Repository**. The Create Subversion Repository dialog displays (Figure 26–11).

Figure 26–11 Create Subversion Repository



Enter details for the new repository. A connection will be created automatically.

Repository Path: C:\repos

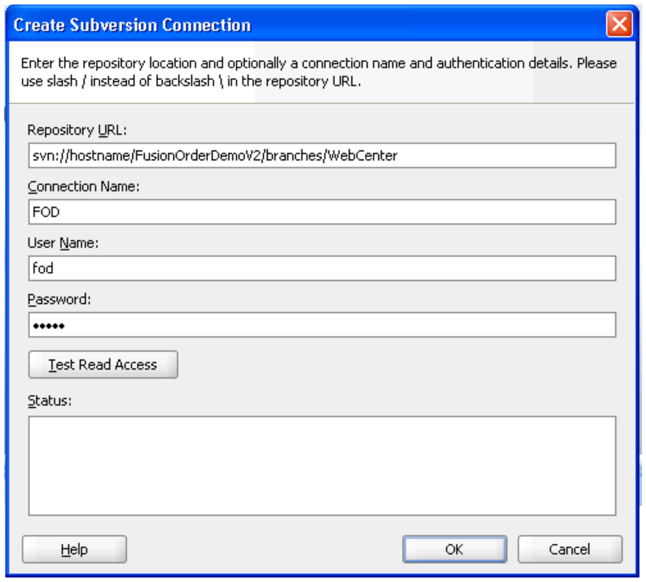
File System Type: [✓ Native] Berkeley DB

Connection Name: FOD

2. In the **Create Subversion Connection** dialog, enter connection details, and then click **OK**. (Figure 26–12)

> **Note:** An alternative way to create a connection is to right-click **Subversion** in the Versioning Navigator and select **New Repository Connection**.

Figure 26–12 Create Subversion Connection



Enter the repository location and optionally a connection name and authentication details. Please use slash / instead of backslash \ in the repository URL.

Repository URL: svn://hostname/FusionOrderDemoV2/branches/WebCenter

Connection Name: FOD

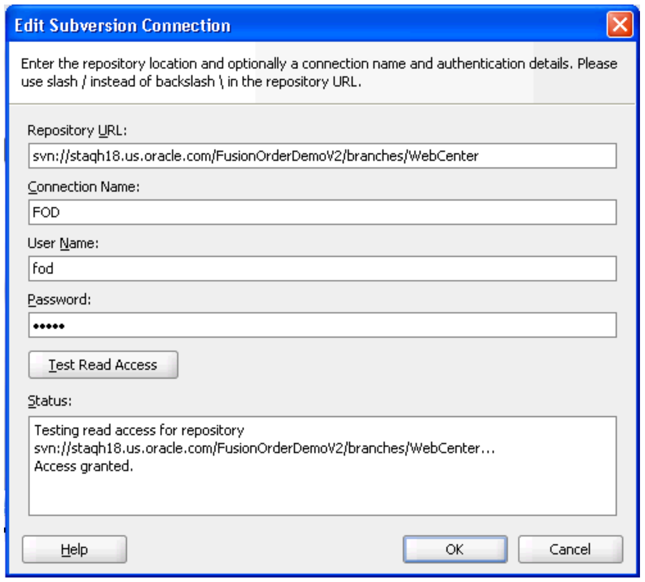
User Name: fod

Password: •••••

Status:

3. Later you can edit these pages on the Edit Subversion Connection page (Figure 26–13).

Figure 26–13 Edit Subversion Connection



Enter the repository location and optionally a connection name and authentication details. Please use slash / instead of backslash \ in the repository URL.

Repository URL: svn://staqh18.us.oracle.com/FusionOrderDemoV2/branches/WebCenter

Connection Name: FOD

User Name: fod

Password: •••••

Status:

```
Testing read access for repository
svn://staqh18.us.oracle.com/FusionOrderDemoV2/branches/WebCenter...
Access granted.
```

26.2 Understanding WebCenter Application Files Affected by Developers

One of the most important aspects of working with any source control system is understanding which files are affected by any particular action. Without this knowledge, you may inadvertently corrupt the source by checking in or checking out either too few or too many files given the actions you are performing on the project. This section helps you understand what files are needed for the main actions you may perform while building a custom WebCenter application.

26.2.1 Files Associated With Common Objects

The main objects with which you work in a WebCenter application are as follows:

- Pages
- Portlets
- Services
- Data controls

Each of these are fairly complex objects, made up of several different metadata files.

For information about page metadata files, see the "Introduction to the ADF Metadata Files" section in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*. For information about portlet and producer metadata files, see Appendix A, "Files for WebCenter Applications."

26.2.2 Developer Actions Affecting Metadata Files

Table 26–1 lists developer actions and the files that are affected by these actions. Take this information into account while managing your files in CVS or Subversion.

Table 26–1 Developer Actions Affecting Metadata Files

Action	Files Affected
Creating or editing one of the following connections: ▪ BPEL ▪ Content Repository; for example, Oracle Content Server, Oracle Portal, file system ▪ Database ▪ Discussions ▪ External Application ▪ Instant Messaging and Presence ▪ Mail ▪ Oracle Secure Enterprise Search ▪ Oracle PDK-Java Producer ▪ WSRP Producer	Adds or updates: ▪ `*app-root*/.adf/META-INF/connections.xml` with the connection ▪ `*app-root*/*projectname*/*projectname*.jpr` with connection libraries ▪ `*app-root*/.adf/META-INF/adf-config.xml` with service-level configuration, including the default connection An external application connection, as well as all connections that require an external application connection (including Mail, Content Repository, and Instant Messaging and Presence) also add or update the following files: ▪ `*app-root*/src/META-INF/jps-config.xml` with the credential store provider ▪ `*app-root*/src/META-INF/cwallet.sso` with the user name and password **Note**: All secure data specified for connections is stored in `cwallet.sso`, referenced in `jps-config.xml`. For example, the password for a database connection is stored in `cwallet.sso.`

Table 26–1 (Cont.) Developer Actions Affecting Metadata Files

Action	Files Affected
Creating a content repository data control	Adds or updates following data control files under *app-root/projectname/*adfmsrc: ▪ *projectpackage/*DataControls.dcx with the data control definition ▪ *projectpackage/dcname*.xml with the data control structure ▪ *projectpackage/dcname/**.xml with data structure definitions used by the data control ▪ *app-root/projectname/*adfmsrc/META-INF/adfm.xml with a pointer to the DCX file ▪ *app-root/projectname/*adfmsrc/return.xml with the definition for default operations used on collections Updates *app-root/projectname/projectname*.jpr with content repository libraries
Registering an Oracle PDK-Java or WSRP portlet producer	Adds or updates *app-root/*src/META-INF/jps-config.xml with the credential store provider Adds: ▪ Tag library ▪ *app-root/*mds/producer.pxml (pertaining to the registered producer) ▪ A number of files pertaining to the registered producer to *app-root/*mds/directory: producer Adds or updates *app-root/projectname/*adfmsrc/*projectpackagename*/DataBindings.cpx with factory classes to data bindings Updates: ▪ *app-root/*.adf/META-INF/connections.xml with a connection to the producer and the URL connection associated with it ▪ *app-root/*.adf/META-INF/adf-config.xml with configuration for the metadata repository and the credential store ▪ *app-root/projectname/*src/META-INF/orion-application.xml with the WebCenter shared libraries ▪ *app-root/projectname/projectname*.jpr with the libraries needed for any WebCenter application including external application libraries ▪ *app-root/projectname/*public_html/WEB-INF/web.xml with the portlet servlets and filters and also external application automated login entries Copies the keystore file to *app-root/*.adf/META-INF if the producer is WSRP and supports WS-Security

Table 26–1 (Cont.) Developer Actions Affecting Metadata Files

Action	Files Affected
De-registering an Oracle PDK-Java or WSRP portlet producer	Updates *app-root/*`.adf/META-INF/connections.xml` to remove the producer and URL connection Removes producer-related files from *app-root/*`mds/` **Note**: Although the producer is removed from the application, you may need to manually remove the metadata files, portlet view tag, and portlet bindings from the application. Everything else should be left alone in case other producers rely on *app-root/projectname/*`public_html/WEB-INF/web.xml` or *app-root/*`.adf/META-INF/adf-config.xml`. The metadata files are removed by the deregister operation; however, any customizations on the producer side might need to be manually removed if the producer was not available during deregistration.
Adding or removing a portlet from a page Adding or removing a portlet on a page from a portlet producer connection in the Resource Palette	When adding a portlet from a portlet producer in the Resource Palette, WebCenter first registers the producer with the application (if it is not already registered). See the "De-registering an Oracle PDK-Java or WSRP portlet producer" row for files affected. It then adds or removes the portlet from the page, affecting the files described in this row. Adds or updates: ■ *app-root/projectname/*`adfmsrc/`*projectpackagename*`/DataBindings.cpx` with a new PageDef mapping (if it does not already exist) ■ *app-root/projectname/*`public_html/WEB-INF/faces-config.xml` (This happens whenever you add an Oracle ADF component, such as a portlet, to a page.) ■ `adf-faces-config.xml` (This happens whenever you add an Oracle ADF component to a page.) ■ *app-root/projectname/*`public_html/WEB-INF/oracle-webservices.xml` if the portlet is WSRP Adds: ■ A number of files pertaining to the newly cloned portlet instance to *app-root/*`mds/` ■ A number of WSDL and XML files to *app-root/projectname/*`public_html/WEB-INF/wsdl` if the portlet is WSRP Updates: ■ *app-root/projectname/*`public_html/`*pagename*`.jspx` with the `<adfp:portlet>` tag (JSF faces tags are replaced with Oracle ADF faces tags.) ■ *app-root/projectname/*`adfmsrc/`*projectpackagename*`/pageDefs/`*pagename*`PageDef.xml` with the region binding and page variable definitions ■ *app-root/projectname/*`public_html/WEB-INF/web.xml` with `secure-ref` entries if the portlet is WSRP

Table 26–1 (Cont.) Developer Actions Affecting Metadata Files

Action	**Files Affected**
Adding a task flow for one of the following services: ■ Announcements ■ Discussions ■ Documents ■ Links ■ Mail ■ Page ■ Recent Activities ■ RSS Viewer ■ Search ■ Tags ■ Worklist ■ External Application	Updates: ■ `app-root/projectname/public_html/pagename.jspx` with appropriate ADF faces tags ■ `app-root/projectname/adfmsrc/projectpackagename/pageDefs/pagenamePageDef.xml` with the region binding ■ `app-root/projectname/adfmsrc/projectpackagename/DataBindings.cpx` with a listing of the page definitions ■ `app-root/projectname/adfmsrc/META-INF/adfm.xml` with a pointer to the data bindings file ■ `app-root/.adf/META-INF/adf-config.xml` with MDS configuration entries ■ `app-root/projectname/projectname.jpr` with service libraries, dependent libraries, and JSP tag libraries ■ `app-root/projectname/public_html/WEB-INF/web.xml` with ADF library, external application, and MDS configuration entries
Dragging a file or folder from a content repository connection to a page Dragging a content repository data control on a page	Adds or updates all connection-related files for any service that requires connections Copies over any required connection that was not present in the application already, which updates `app-root/.adf/META-INF/connections.xml` and `app-root/projectname/projectname.jpr` **Note**: The Recent Activities, Search, and Tags services add the search extension to `adf-config.xml` and create the sample `service-definition.xml` file. The `service-definition.xml` file lets developers define `QueryManager` implementations for their service.

Table 26–1 (Cont.) Developer Actions Affecting Metadata Files

Action	Files Affected
Adding one of the following components or services on a page: ■ Customizable Components ■ Instant Messaging and Presence ■ Links ■ Tags ■ Oracle Composer	Updates: ■ `app-root/.adf/META-INF/adf-config.xml` with MDS configuration entries and resource catalog service ■ `app-root/projectname/adfmsrc/projectpackagename/pageDefs/pagenamePageDef.xml` with the region binding ■ `app-root/projectname/adfmsrc/META-INF/adfm.xml` with a pointer to the data bindings file ■ `app-root/projectname/adfmsrc/projectpackagename/DataBindings.cpx` with the task flow binding definition factory and pageDef entries ■ `app-root/projectname/public_html/pagename.jspx` with appropriate ADF faces tags ■ `app-root/projectname/public_html/WEB-INF/faces-config.xml` with the ADF Controller Phase Listener ■ `app-root/projectname/public_html/WEB-INF/web.xml` with ADF library, external application, and MDS configuration entries ■ `app-root/projectname/projectname.jpr` with service libraries and dependent libraries **Note**: The `projectname.jpr` and `web.xml` files are updated if the service is dependent on an external application. **Note**: The Tags service has a design time dependency on the Search service, so it also adds Search service libraries.
Creating a JSR 168 portlet application or adding a portlet to an existing application that uses JSP file	Adds or updates: ■ `app-root/projectname/public_html/WEB-INF/portlet.xml` ■ `app-root/projectname/public_html/WEB-INF/oracle-portlet.xml` ■ `app-root/projectname/public_html/WEB-INF/web.xml` Adds Java and JSP source files for the portlet

Table 26–1 (Cont.) Developer Actions Affecting Metadata Files

Action	Files Affected
Creating a JSR 168 portlet application or adding a portlet to an existing application that uses the ADF/JSPX option	A portlet application generated with this option uses the Oracle Portlet Bridge to access a Faces application. Adds or updates: ■ *app-root/projectname/*`public_html/WEB-INF/portlet.xml` with configuration information for the portlet bridge ■ *app-root/projectname/*`public_html/WEB-INF/oracle-portlet.xml` with configuration information for the portlet bridge ■ *app-root/projectname/*`public_html/WEB-INF/web.xml` with configuration information for a Faces servlet ■ *app-root/*`src/META-INF/orion-application.xml` with shared library import statements for `adf.oracle.domain` and `oracle.webcenter.skin` Adds: ■ Java and JSPX source files for the portlet ■ *app-root/projectname/*`public_html/WEB-INF/faces-config.xml`
Creating a PDK portlet producer application	Adds or updates: ■ *app-root/projectname/*`public_html/WEB-INF/web.xml` with producer servlet definitions and producer deployment details ■ `service_id.properties` with producer configuration attributes ■ *app-root/*`src/META-INF/orion-application.xml` with shared library import statements for `oracle.portal` Adds Java and JSP files for the portlet
Registering a secure WSRP producer (WS-Security)	Updates *app-root/*`.adf/META-INF/connections.xml` with the secure producer connection definition **Notes**: The `connections.xml` entry for a secure producer looks different in terms of how the security policy is stored. Instead of the whole security policy description, the security policy is just a URI (that is, a "pointer" to the policy in the policy repository). The secure ref attributes in `connections.xml` are stored in the credential store (as defined by *app-root/*`src/META-INF/jps-config.xml`). The credential store configuration information (as defined by `jps-config.xml`) is stored in *app-root/*`.adf/META-INF/adf-config.xml`. At design time, the credential store is stored in an Oracle Wallet by default; that is, `META-INF/cwallet.sso`.

Table 26–1 (Cont.) Developer Actions Affecting Metadata Files

Action	Files Affected
Configuring ADF Security for a project	Adds or updates: ▪ `*app-root*/*projectname*/public_html/WEB-INF/web.xml` with login configuration, ADF authentication servlet, security constraints, logical roles and mappings ▪ `*app-root*/src/META-INF/jps-config.xml` with identity store, credential store, policy store, anonymous provider, and login modules ▪ `*app-root*/.adf/META-INF/adf-config.xml` with ADF identity store and credential store configuration ▪ `*app-root*/src/META-INF/jazn-data.xml` with an XML identity store ▪ `*app-root*/src/META-INF/jazn-data.xml` with application permissions

26.3 Implementing Common Requirements Once

It is good practice for the project administrator to implement any common developer requirements once and then check in that version for all to use. By planning ahead and having the administrator take care of these common requirements up front, you can reduce redundancy and error.

For example, suppose two developers need to add OmniPortlet on different pages of an application. If the project administrator has registered the OmniPortlet producer, then it is available for both of them to use. If not, then each developer likely will register the OmniPortlet producer separately, leading to unnecessary duplication and confusion.

Another example: suppose many developers need content from the same content repository. One person should set up and check in the needed connection first. Other developers then can simply reuse the same data control.

26.4 Portlet Producer Considerations

When working in a team environment, bear in mind the following considerations pertaining to portlet producers:

- Section 26.4.1, "Portlet Producer Connections"
- Section 26.4.2, "Portlet Producer Name Clashes"
- Section 26.4.3, "Combining Portlets from Different Portlet Producers"

26.4.1 Portlet Producer Connections

When building an EAR file for your project, connections are loaded for the entire application rather than individual projects. For example, suppose you have an application with two projects: P1 and P2. P1 has 100 registered producers and P2 has no producers. When you build an EAR file for either project, all 100 of P1's producer connections are loaded into `connections.xml`. Note, though, that you can also edit `connections.xml` manually.

When you run the predeployment tool to create a targeted EAR file, all of the connections in `connections.xml` must be accessible. Hence, in our example, the generation of a targeted EAR file for either project would fail if any of the 100 portlet producer connections for P1 are unavailable for some reason. Given this behavior, you must carefully consider how you plan to subdivide your overall development effort into applications and projects.

26.4.2 Portlet Producer Name Clashes

While Oracle WebCenter Framework lets you register two portlet producers under the same name, it generally is better to avoid this situation. For example, if two developers working on the same application inadvertently register a portlet producer with the same name, then usually it is best to change one of the names to be unique. If you have two portlet producers registered under the same name, then it becomes very difficult to distinguish between them when debugging errors or performing administrative tasks on the portlet producers.

26.4.3 Combining Portlets from Different Portlet Producers

In some cases, you might have multiple developers building portlets and ultimately you want those portlets to be combined under a single portlet producer. The developers must be conscious of some potential issues.

- Portlet names and JSP paths could clash. Portlet developers should use prearranged class package names and JSP paths to avoid naming clashes when portlets are combined within one portlet producer in one application.
- When you create a JPS portlet in the Portlet wizard, the directory for the portlet modes defaults to `portlet`*`n`*`\html\`*`mode_name`*, where *`n`* is a number that increments for each portlet you create. To avoid directory and file name clashes, portlet developers should change the directory name on the Content Type and Portlet Modes page of the Portlet wizard. Select the portlet mode and then change the directory name in the corresponding field to something unique.
- When you combine portlets into one portlet producer, you must manually merge the portlet descriptor files, avoiding identifier clashes as you do so as follows:
  - JPS portlet identifiers in the `portlet.xml` and `oracle-portlet.xml` files are generated automatically starting from `portlet1`. When you manually merge multiple portlet descriptor files into one, you must change any portlet identifiers that clash with one another.
  - Similarly, the PDK-Java portlet identifiers in `provider.xml` are automatically generated starting from `1`. When you manually merge multiple `provider.xml` files, you must change any portlet identifiers that clash with one another.
- You must manually merge any web descriptor (`web.xml`) changes; for example, security role information.
- For PDK-Java portlets, you also might need to manually merge `.properties` files.
- When using many different portlets, it is possible that parts of files are not checked in properly because they are overwritten by someone else or because JDeveloper does not pick up the changed files correctly. For example, you might deploy an application with many types of portlets and see the following error in two of the three OmniPortlets:

```
mdsId
oracle/adf/portlet/OmniPortlet_1172537210873/ap/Portlet100_0b4156e9_0111_1000_
8001_82235f273a39.pxml not found
```

```
mdsId
oracle/adf/portlet/OmniPortlet_1172537210873/ap/Portlet100_250498fc_0111_1000_
8002_a9fe7286a54e.pxml not found
```

If you receive error messages like these, then make sure that all `*.pxml` files from the development environment are copied over to the deployed environment.

- Your portlet customizations might not show up in your deployed environment. Make sure that the content of the `oracle\adf\portlet\<`*`portletInstanceName`*`>.pxml` file is correct and that it refers to the proper *`*.pxml`* files.

Part V

Building Portlets

Part V contains the following chapters:

27
Overview of Portlets

This chapter provides an overview of portlets, and describes, with the help of examples, the use of portlets. It explains portlet anatomy and the resources to create portlets. This chapter also explains the features, technologies, and tools to help you decide which portlet building technology best suits your needs.

This chapter includes the following sections:

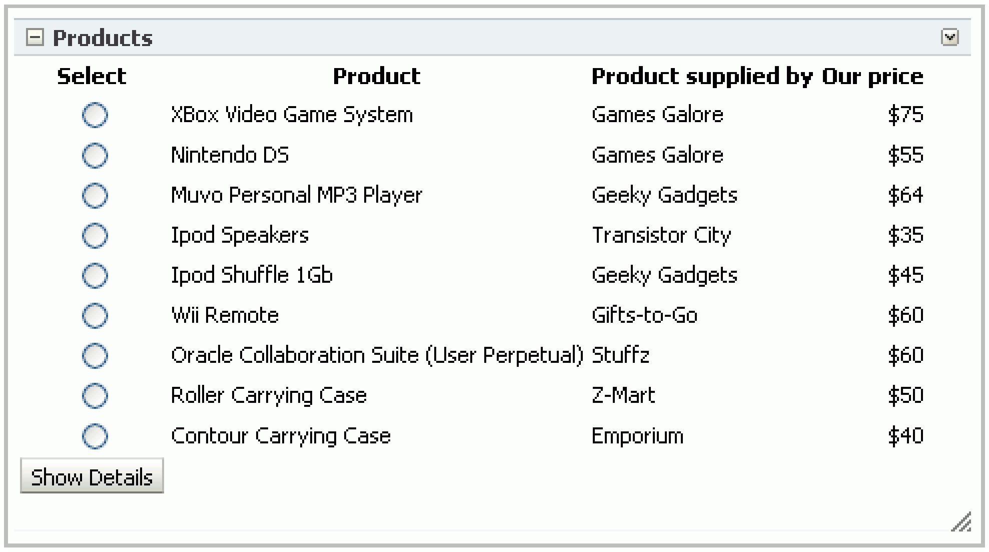
- Section 27.1, "Introduction to Portlets"
- Section 27.2, "Portlet Technologies Matrix"

27.1 Introduction to Portlets

A portlet is a reusable Web component that can draw content from many different sources. Figure 27–1 illustrates the Products portlet, a portlet that shows the products available on a shopping web site.

For information about how to create this portlet, see "Building Portlets and Wiring Them Together in Your Application" in the *Oracle Fusion Middleware Tutorial for Oracle WebCenter Developers*

Figure 27–1 The Products Portlet



Select	Product	Product supplied by	Our price
○	XBox Video Game System	Games Galore	$75
○	Nintendo DS	Games Galore	$55
○	Muvo Personal MP3 Player	Geeky Gadgets	$64
○	Ipod Speakers	Transistor City	$35
○	Ipod Shuffle 1Gb	Geeky Gadgets	$45
○	Wii Remote	Gifts-to-Go	$60
○	Oracle Collaboration Suite (User Perpetual)	Stuffz	$60
○	Roller Carrying Case	Z-Mart	$50
○	Contour Carrying Case	Emporium	$40

Portlets provide a means of presenting data from multiple sources in a meaningful and related way. Portlets can 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. Because different portlets can be placed on a common page, the user receives a single-source experience. In reality, the content may be derived from multiple sources.

In a custom WebCenter application, a portlet may or may not be rendered in an inline frame, or `<iframe>`. Inline frames enable the placement of a document within a rectangular region that includes scroll bars and borders.

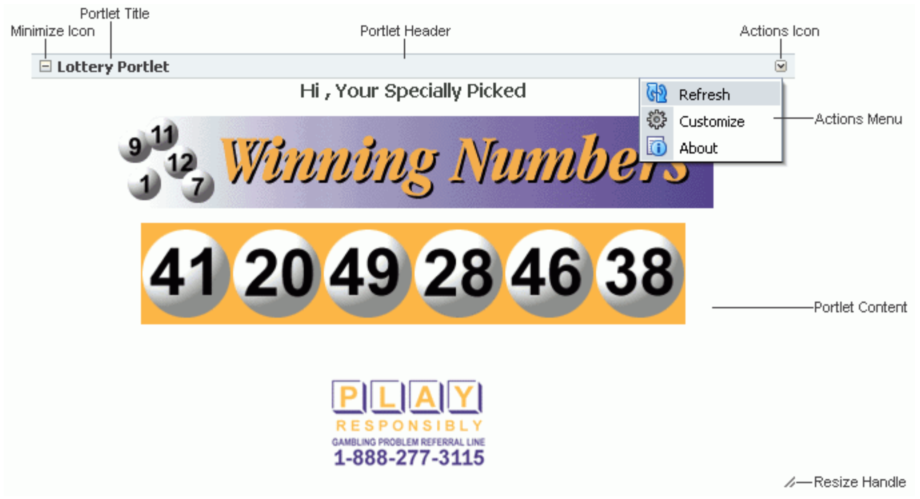
Note: For more information about portlets and inline frames, see Section 9.4.13, "What You May Need to Know About Iframes."

Within this inline frame, portlets can display many types of content, including HTML, formatted text, images, or elements of an HTML form.

27.1.1 Portlet Anatomy

Portlet anatomy is the visual representation of the portlet on a page. Figure 27–2 illustrates a typical portlet anatomy of a page in a custom WebCenter application. Note that the same portlet displayed in a different application could look different.

Figure 27–2 Portlet Anatomy



What is rendered on the page is controlled not only by the portlet's own logic, but by the attributes of the portlet tag that binds the portlet to the page. Values for these attributes are specified at the design time of the application that consumes the portlet, rather than through the portlet's own logic.

Note: For information about attributes of the `adfp:portlet` tag, see Section 9.4, "Setting Attribute Values for the Portlet Tag."

For example, at application design time you can specify through portlet tag attributes that the runtime portlet should display a header and that its border should be of a specified thickness and color. In the header, you can include a portlet title and an Actions menu icon. The Actions menu icon is displayed on a portlet only when the portlet header is displayed. If you choose not to display a header, then the Actions menu is displayed on a FadeIn-FadeOut toolbar that renders on a mouse rollover.

These elements are sometimes referred to as *portlet chrome*. The appearance of portlet chrome can be controlled through a style sheet and through style-related attributes of the `adfp:portlet` tag. The values of style-related attributes take precedence over styles specified through a style sheet, or *skin*.

Note: For information about skins and style-related attributes, see Section 4.2.16, "How to Apply Styles to Components."

Through portlet tag attributes, you can include or omit display commands on the Actions menu. Actions menu items controlled through portlet tag attributes include *Maximize* and *Restore*. Maximize causes the maximized portlet to displace all other displayed portlets. These are displayed again when a user selects Restore.

Note: The Maximize attribute is meaningful for a portlet only when the portlet is placed inside a `PanelCustomizable` core customizable component.

Other Actions menu items controlled through portlet tag attributes include the display or omission of the mode settings that were specified when the portlet was developed. If the portlet was built without including additional modes, then these commands do not appear on the Actions menu even when you indicate that they should through portlet tag attributes. In other words, portlet tag attributes are sometimes on/off switches that enable or disable the portlet's own built-in functionality.

Mode settings that appear on the Actions menu include such modes as *About*, *Help*, *Personalize*, and *Customize*.

Users select Personalize to alter their personal view of the portlet. The Personalize command is displayed on the Actions menu only to authenticated users (that is, users who are logged in). It does not display to Public or unauthenticated users. You must implement some form of application security for users to be able to personalize their portlet views.

Note: If you are a developer creating portlets, and you want to test the Personalize mode without creating a complete security model for your application, see Section 24.8, "Configuring Basic Authentication for Testing Portlet Personalization."

Customization enables application administrators to edit a portlet's default settings at runtime. All users see the results of a customization.

Note: A typical customization setting is Portlet Title. At runtime, the portlet administrator can determine what title should appear in the portlet header. The Portlet Title can also be set at design time through portlet properties, using the `text` attribute of the `adfp:portlet` tag. Consider however that supplying a value to the `text` attribute at design time prevents customization of the Portlet Title at runtime.

For additional information about portlet tag attributes, see Section 9.4, "Setting Attribute Values for the Portlet Tag."

27.1.2 Portlet Resources

Portlet resources include the many prebuilt portlets available out of the box from many sources, including Oracle Portal, Oracle E-Business Suite, and third-party sources. Portlet resources also include programmatic portlets built through the custom WebCenter application's JSR 168 (standards-based) and Oracle PDK-Java wizards, and through other portlet-building tools. Each of these tools offers different product features that are targeted toward different developer roles.

For specific information about each tool and its benefits, see Section 27.2, "Portlet Technologies Matrix."

27.1.2.1 JSF Portlets

What Are They?

JSF portlets are created using the Oracle JSF Portlet Bridge. The Oracle JSF Portlet Bridge enables application developers to expose their existing JSF applications and task flows as JSR 168 portlets. The Oracle JSF Portlet Bridge simplifies the integration of JSF applications with WSRP portlet consumers, such as Oracle Portal. You can create JSF portlets using the JSR 168 Java Portlet Wizard by invoking it from the New Gallery.

JSF portlets do not require separate source code from that of the JSF application. Since these portlets are created using the Oracle JSF Portlet Bridge, you need only maintain one source for both your application and your portlets. Similarly, when you deploy your JSF application, JSF portlets are also deployed with it. Therefore, using the bridge eliminates the need to store, maintain, and deploy your portlets separately from your application.

Who Is the Intended User?

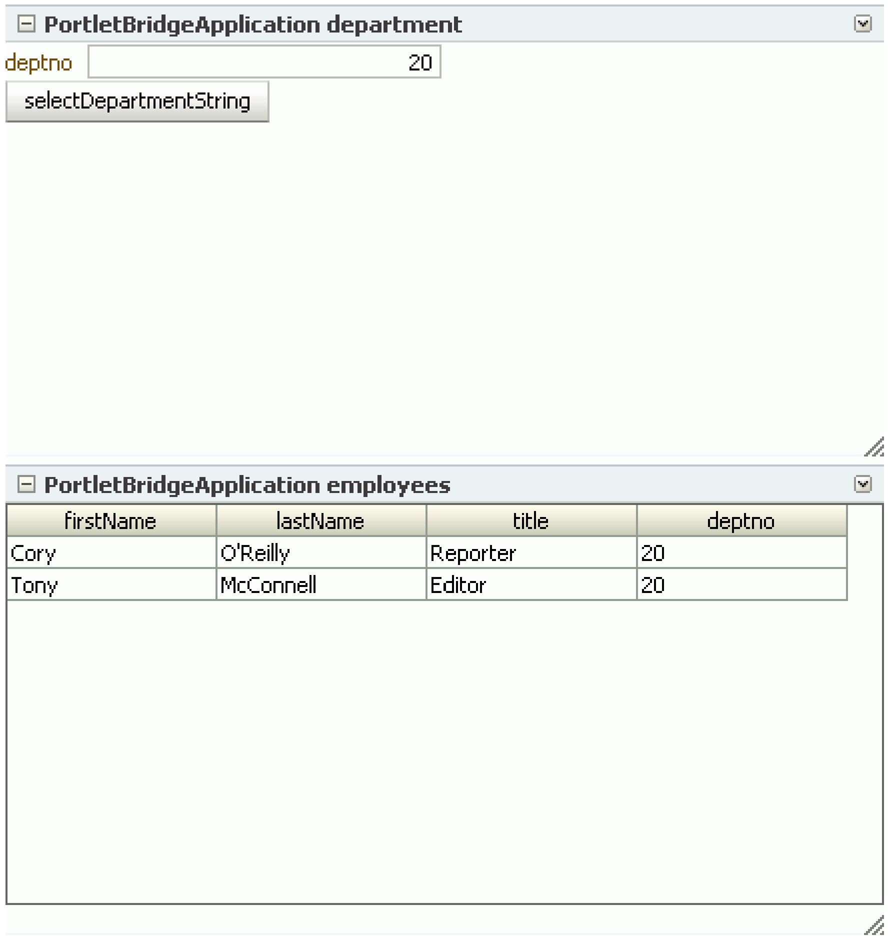
Application developers with the knowledge of Faces and WSRP.

When Should They Be Used?

JSF portlets are best suited when application developers intend to display contents from a JSF application as a portlet without hosting the entire application, or without separately building a portlet for the same. When portletized, the consumption of the portlet is same as registering any WSRP producer using their provider URLs. ADF components that is task flows, can be exposed using the Oracle JSF Portlet Bridge as well.Figure 27–3 shows two portlets, one where users can enter a department number, and one that displays the employee information for the specified department. These portlets were created from ADF task flows using the Oracle JSF Portlet Bridge.

For information about how to create these portlets, see Section 28.3, "Creating JSF Portlets: Example."

Figure 27–3 JSF Portlets


PortletBridgeApplication department
deptno
20
selectDepartmentString
PortletBridgeApplication employees

firstName	lastName	title	deptno
Cory	O'Reilly	Reporter	20
Tony	McConnell	Editor	20

27.1.2.2 Rich Text Portlet

What Is It?

The Rich Text portlet, based on the WSRP 2.0 standard, offers browser-based rich text editing at runtime. Select **Customize** from the portlet's **Actions** menu to invoke a toolbar with all the rich-text editing tools you need to insert, update, and format display text. Click the editor's **Submit** button to save your changes and hide the toolbar. (Selecting **Refresh** from the portlet's Actions menu also hides the toolbar).

With the Rich Text portlet, when an authorized user selects the **Customize** menu item the portlet enters edit mode while the page that contains the portlet remains in view mode.

Portlet configuration settings—available at design time—include controls for the portlet's name, description, and display settings.

Who Is the Intended User?

Application developers can add the Rich Text portlet to a page and provide initial content. Unless the application developer prohibits it, end users with the appropriate privileges can update the content of the Rich Text Portlet at runtime.

When Should It Be Used?

The Rich Text portlet is a useful tool for runtime posting of enterprise announcements and news items. It provides straightforward, easy-to-use controls for entering and formatting display text.

27.1.2.3 Prebuilt Portlets

What Are They?

Prebuilt portlets are available through Oracle's partnerships with leading system integrators, software vendors, and content providers. You can access these portlets by using the keywords `portal` or `portlet` when searching the Oracle PartnerNetwork Solutions Catalog, available at `http://solutions.oracle.com`. Examples of these include portlets for the following purposes:

- Generating point-to-point driving directions
- Accessing Information Technology (IT) information from a wide variety of sources
- Viewing summary information about news, stocks, and weather

Who Is the Intended User?

Fully developed, downloadable portlets are best suited for use by application developers who understand how to download, install, and register producers in Oracle WebCenter Framework. They are available for use by all levels of experience.

When Should They Be Used?

Use prebuilt portlets when your needs are satisfied by the functions the portlets offer and the level of personalization readily available is sufficient to complete the desired task.

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

27.1.2.4 Parameter Form and Parameter Display Portlets

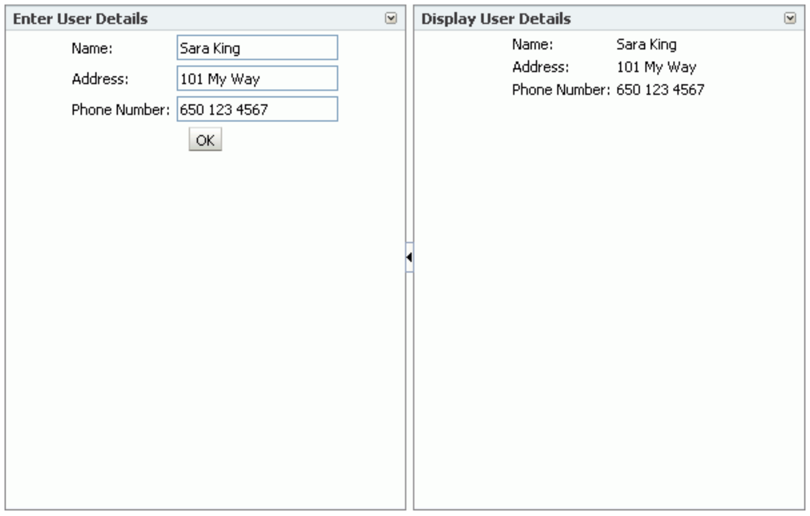
What Are They?

The Parameter Form and Parameter Display portlets provide a quick and easy way to pass values between components. They are provided by the WSRP Tools producer.

The Parameter Form portlet has three output parameters that are set when values are submitted in the form inside the portlet. The parameters can then be used to drive the content of other portlets. You can customize the Parameter Form portlet to determine how many of the three fields are displayed on the form, depending on how many parameters you require.

The Parameter Display portlet enables you to quickly test the wiring from the Parameter Form portlet. However, typically you use the values passed from the Parameter Form portlet to drive the content of some other portlet, for example to pass a zip code to a weather portlet, or a stock symbol to a stock ticker portlet.

Figure 27–4 Parameter Form and Display Portlets



Name: Sara King
Address: 101 My Way
Phone Number: 650 123 4567

Name: Sara King
Address: 101 My Way
Phone Number: 650 123 4567

For more information about linking portlets, see Section 9.7, "Contextually Linking WSRP 2.0 Portlets."

Who Is the Intended User?

The Parameter Form and Parameter Display portlets are best suited for use by application developers who want to provide contextual linking between portlets on a page. These portlets can be added to a page by any user with the appropriate privileges.

When Should They Be Used?

Use the Parameter Form and Parameter Display portlets when your needs are satisfied by the functions the portlets offer and the level of personalization readily available is sufficient to complete the desired task.

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

27.1.2.5 Web Clipping

What Is It?

Web Clipping is a browser-based declarative tool that enables the integration of any Web application with a custom WebCenter application. Web Clipping is designed to provide quick integration by leveraging the Web application's existing user interface. Web Clipping has been implemented as a PDK-Java producer.

To create a Web Clipping portlet, the custom WebCenter application developer uses a Web browser to navigate to a Web page that contains desired content. Through the Web Clipping Studio, the application developer can drill down through a visual rendering of the target page to choose the desired content.

Web Clipping supports the following:

- **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, then it is still 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).
- **Personalization,** allowing an application developer to expose input parameters that users can modify when they personalize the portlet. These parameters can be exposed as public parameters that an application developer can map as page parameters. This feature enables users to obtain personalized clippings.
- **Integrated authenticated Web content through Single Sign-On,** including integration with external applications, which enables you to leverage Oracle 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 authentication for each user. You can specify the realm of the proxy server and whether all users automatically log in using a user name and password you provide, each user logs in using an individual user name and password, or all users log in using a specified user name and password.
- **Resource Tunneling** of images.
- **Open Transport API** for customizing authentication mechanisms to clipped sites.

Who Is the Intended User?

Web Clipping is best suited for use by application developers and component developers who want to leverage an existing Web page for rapid portlet development. This portlet can be added to a page by any user with the appropriate privileges.

When Should It Be Used?

Use Web Clipping when you want to repurpose live content and functionality from an existing Web page and expose it in your custom WebCenter application as a portlet. Consider alternatives to change the way information is presented in the clipped portlet. That is, you do not need to control the User Interface (UI) or application flow, and you are accessing Web-based applications. For a greater level of control, use OmniPortlet's Web page data source instead of Web Clipping. (For more information, see Section 27.1.2.6, "OmniPortlet.")

The following are some examples of when you can consider using the Web Clipping portlet:

- **Stock chart portlet.** 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.
- **Web mail portlet.** Your users want to access their confidential Web mail accounts through a portlet and to display their in-boxes in the portlet.

For more information about using Web Clipping, see Chapter 32, "Creating Content-Based Portlets with Web Clipping."

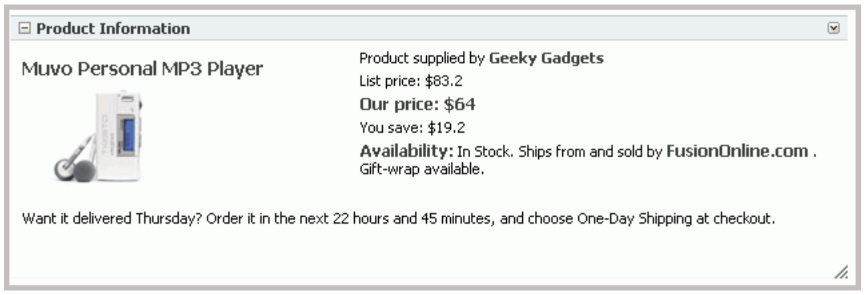
27.1.2.6 OmniPortlet

What Is It?

OmniPortlet is a declarative portlet-building tool that enables you to build portlets against a variety of data sources, including XML files, character-separated value files (CSV, for example, spreadsheets), Web Services, databases, and Web pages. OmniPortlet users can also choose a prebuilt layout for the data. Prebuilt layouts include tabular, news, bullet, form, chart, or HTML. HTML layout enables OmniPortlet users to write their own HTML and inject the data into the HTML. Figure 27–5 shows a portlet created with OmniPortlet.

For information about how to create this portlet, see "Building Portlets and Wiring Them Together in Your Application" in the *Oracle Fusion Middleware Tutorial for Oracle WebCenter Developers*

Figure 27–5 Product Information Portlet



Muvo Personal MP3 Player

Product supplied by **Geeky Gadgets**
List price: $83.2
Our price: $64
You save: $19.2
Availability: In Stock. Ships from and sold by **FusionOnline.com** .
Gift-wrap available.

Want it delivered Thursday? Order it in the next 22 hours and 45 minutes, and choose One-Day Shipping at checkout.

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

Who Is the Intended User?

Business users with a minimum knowledge of the URLs to their targeted data may find OmniPortlet a valuable tool.

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.

The following are some examples of when you can consider using OmniPortlet:

- **RSS news feed portlet:** You want to create a portlet that displays live, scrolling news information to your users. The data comes from a Really Simple Syndication (RSS) news feed, such as Oracle Technology Network Headlines. You also want the portlet to contain hyperlinks to the news source.
- **Sales chart portlet:** You want to present up-to-date information about 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.

For more information about OmniPortlet, see Chapter 31, "Creating Portlets with OmniPortlet."

27.1.2.7 Programmatic Portlets

What Are They?

Programmatic portlets are portlets that you write yourself, in Java, using either the standard Java Portlet Specification (JPS) or PDK-Java. Oracle WebCenter Framework provides two declarative wizards for simplifying the creation of standards-based JSR 168 portlets and Oracle PDK-Java portlets. These wizards assist in the construction of the framework within which you create the portlet. Each wizard may include easy steps for the following:

- Configuring general portlet properties
- Specifying names and search terms
- Setting allowable content types and mapping display modes
- Specifying user-customizable preferences
- Adding security roles
- Enabling default caching
- Adding initialization parameters
- Adding navigation parameters

Who Is the Intended User?

Use of the wizards is easy, but the creation of portlet logic is best performed by experienced and knowledgeable Java developers who are comfortable with the Java Portlet Specification or PDK-Java and who understand the configuration of producers.

When Should They Be Used?

Use programmatic portlets when you have very specialized business rules or logic or when you require personalized authentication, granular processing of dynamic results, and complete user interface control. Additionally, use programmatic portlets when you need to satisfy any of the following conditions:

Consider using the programmatic approach when the out-of-the-box portlets do not address your needs.

The following list provides a couple of examples of when you can consider using programmatic portlets:

- **Photo Album portlet:** You want to create a a portlet that facilitates uploading, storing, and viewing user photos.
- **Shopping Cart portlet:** You want to create a portlet that facilitates the viewing and purchasing of, for example, company-branded items, such as mouse pads, pens, flash drives, tee shirts, and the like.

For more information about using programmatic portlets, see Chapter 29, "Creating Portlets with the Portlet Wizard" and Chapter 30, "Coding Portlets."

27.1.2.8 Deciding Which Tool to Use

Figure 27–6 illustrates the spectrum of portlet resources described in the previous section. Notice how one end of the spectrum is geared toward a more declarative development environment (that is, develop-through-wizard) while the other end focuses more on hand-coding. You can choose your tool depending on which type of environment is most comfortable and suitable to your skill-base.

For more information about deciding which tool to use, see Section 27.2, "Portlet Technologies Matrix."

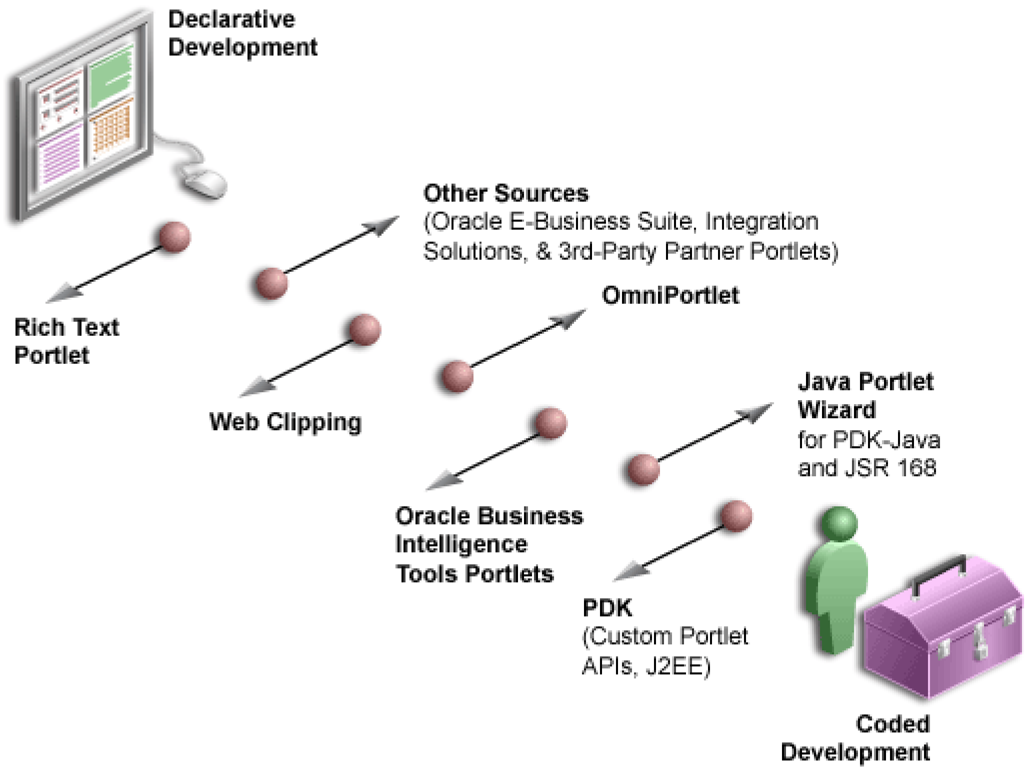

Declarative Development
Other Sources (Oracle E-Business Suite, Integration Solutions, & 3rd-Party Partner Portlets)
OmniPortlet
Rich Text Portlet
Web Clipping
Java Portlet Wizard for PDK-Java and JSR 168
Oracle Business Intelligence Tools Portlets
PDK (Custom Portlet APIs, J2EE)
Coded Development

Figure 27–6 Portlet Resources from Declarative to Coded Development

27.2 Portlet Technologies Matrix

Table 27–1 summarizes the technologies and tools you can use with Oracle WebCenter Framework. The matrix describes the tools and technologies that are covered in more detail in this guide: Oracle JSF Portlet Bridge, OmniPortlet, Web Clipping portlet, and programmatic portlets, including standards-based (JSR 168) portlets and PDK-Java portlets.

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 BI Discoverer. These other tools are not covered in this guide.

The other sections in this chapter provide further detail on the characteristics listed in Table 27–1. Use the table to quickly scan all the features and characteristics, then see the subsequent sections for more in-depth information.

Table 27–1 Portlet Building Technologies Comparison Matrix

Oracle JSF Portlet Bridge	Web Clipping	OmniPortlet	Programmatic Portlets Standards-Based/PDK-Java
General Suitability			
Oracle JSF Portlet Bridge exposes JSF and ADF artifacts as JSR 168 portlets.	A simple wizard-based tool, accessible from a browser, that assists in retrieving and presenting Web content that originates from other Web sites in a custom WebCenter application.	Wizard-based tool, accessible from a browser, that assists in retrieving and presenting data from a wide variety of data sources.	PDK-Java offers Oracle-specific application programming interfaces (APIs) for building portlets for use in custom WebCenter applications and Oracle Portal. Standards-based portlets additionally work with portals of other vendors. Oracle WebCenter Framework supports both WSRP and JSR-168 standards.
Expertise Required			
No expertise required.	No expertise required.	Basic understanding of one or more supported data sources and the concepts of portlet and page parameters.	Java, Servlet, JSP knowledge.
Supported Data Sources (for details, see Section 27.2.2, "Expertise Required")			
No limitations.	Any Web site accessible on the network over HTTP or HTTPS.	CSV, XML, Web Service, SQL, Web site, JCA.	No limitations.
Deployment Type			
WSRP producers using Oracle JDeveloper or manually.	PDK-Java producers	PDK-Java producers	PDK-Java uses PDK-Java producers. Standards-based portlets use WSRP producers.
Caching Style			
Expiry-based caching, validation-based caching (auto invalidate when personalized).	Expiry-based caching, validation-based caching (auto invalidate when personalized).	Expiry-based caching, validation-based caching (auto invalidate when personalized).	Expiry-based, validation, and invalidation caching, Edge Side Includes. **Note:** JSR 168 does not support validation based caching. WSRP 1.0 does. If you use a pure WSRP portlet, then validation-based caching is also supported. If you host a JSR portlet on WSRP (as is done in Oracle WebCenter Framework) then validation-based caching is not supported.
Development Tool			
Oracle JDeveloper Oracle JSF Portlet Bridge/JSF Portletization dialog.	Browser - wizard.	Browser - wizard.	Oracle JDeveloper Java Portlet wizard (or any other Java development environment).
Portlet Creation Style			
Develop a JSF application first, convert it into a JSF portlet later.	design time at runtime.	design time at runtime.	Develop first, add later.
User Interface Flexibility			
Very Flexible.	N/A	Very flexible, by using the HTML layout.	Very flexible.

Table 27–1 (Cont.) Portlet Building Technologies Comparison Matrix

Oracle JSF Portlet Bridge	Web Clipping	OmniPortlet	Programmatic Portlets Standards-Based/PDK-Java
Ability to Capture Content from Web Sites			
No. Depends upon the application being portletized.	Yes, by its nature.	Yes, by using the Web Page Data Source.	Yes. For PDK-Java, use the `oracle.portal.provider.v2.*` package. For standards-based portlets, use the `java.net` package.
Ability to Render Content Inline			
Yes.	Yes	No. However, inline rendering can be achieved through public portlet parameters.	Yes. For PDK-Java, use private portlet parameters. Standards-based portlets include servlets and JSPs, using the method `PortletContext.getRequestDispatcher()`.
Charting Capability			
No.	No	Yes, 2D-3D charts.	Yes, using BI Beans.
Public Portlet Parameter Support			
Yes	Yes	Yes	Yes **Note:** In standards-based portlets, support is available for public parameters using WSRP 2.0's navigational parameter feature along with Oracle WebCenter Framework extensions to JSR 168.
Private Portlet Parameter Support			
Yes	No	No	Yes
Ability to Hide and Show Portlets Based on User Privileges			
Yes.	No, though it is possible to apply security managers that are not exposed through the user interface (UI).	No, though it is possible to apply security managers that are not exposed through the UI.	Yes. For PDK-Java, by using the Security managers. For standards-based portlets, the Servlet security model is supported by using methods such as `PortletRequest.isUserInRole()` and `PortletRequest.getUserPrincipal()`.
Multilingual Support			
Yes.	N/A	Yes	Yes
Pagination Support			

Table 27–1 (Cont.) Portlet Building Technologies Comparison Matrix

Oracle JSF Portlet Bridge	Web Clipping	OmniPortlet	Programmatic Portlets Standards-Based/PDK-Java
Yes.	N/A	No	Yes, programmatically
Authenticating to External Applications			
Through custom user attributes.	External application integration supported.	Basic authentication support if the data source requires it.	For PDK-Java portlets, external application integration is supported. LDAP integration is supported when the portlet is running behind the same firewall as the LDAP server. For standards-based portlets, though not specifically supported, external application support is feasible through custom user attributes.) LDAP integration is supported.

27.2.1 General Suitability

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

27.2.1.1 Oracle JSF Portlet Bridge

The Oracle JSF Portlet Bridge is a dialog-based tool that enables page developers to expose their existing JSF applications and task flows as JSR 168 portlets. Using the Oracle JSF Portlet Bridge to create JSF portlets from JSF applications does not require a technical background.

27.2.1.2 Web Clipping

Web Clipping is a simple wizard-based tool that assists with retrieving and presenting Web content that originates from other Web sites in a custom WebCenter application. Web Clipping does not require a technical background.

Examples of portlets you can build using Web Clipping

The examples of portlets that you can build by using Web Clipping are as follows:

- Stock chart portlet
- Web mail portlet
- News portlet containing dynamic content from an existing Web site

27.2.1.3 OmniPortlet

OmniPortlet is an easy-to-use, wizard-based tool for presenting information from a wide variety of data sources in a variety of formats. OmniPortlet runs completely in the browser. Drop OmniPortlet on a page, click the Define link, and choose a data source and a presentation format. Select from a wide variety of data sources as follows:

- Spreadsheet
- SQL
- XML

- Web Service
- Web page

OmniPortlet does not require the use of an additional development tool or a strong technical background. Even so, it can be used for building reusable, high-performing portlets.

Examples of portlets you can build using OmniPortlet

The examples of portlets that you can build by using OmniPortlet are as follows:

- RSS news feed portlet
- Sales chart portlet

27.2.1.4 Programmatic Portlets

If the wizard-based portlet building tools do not satisfy your needs, then 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, a tool available through the Oracle WebCenter, assists with building 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.

Examples of portlets you can build using Java

The examples of portlets that you can build by using Java are as follows:

- Discussion forum portlet
- E-mail portlet

27.2.2 Expertise Required

While some 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.

27.2.2.1 Oracle JSF Portlet Bridge

The Oracle JSF Portlet Bridge does not require a technical background. However, you must have an understanding of Faces and WSRP.

27.2.2.2 Web Clipping

Web Clipping does not require a technical background. However, to parameterize the Web page content that you clipped, you must have an understanding of public portlet parameters and page parameters.

27.2.2.3 OmniPortlet

OmniPortlet requires a basic knowledge of the data source you want to leverage in your portlet. Table 27–2 lists the types of data sources that can be used with OmniPortlet and describes the type of information required to work with each type.

Table 27–2 OmniPortlet Data Sources

Data Source	Required Information
Spreadsheet	The URL that points to the spreadsheet containing the data 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 Web Services Description Language (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 OmniPortlet Wizard's Type page, a J2EE 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).

27.2.2.4 Programmatic Portlets

To build Java portlets, you must know at least a subset of Java EE. 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.

27.2.3 Deployment Type

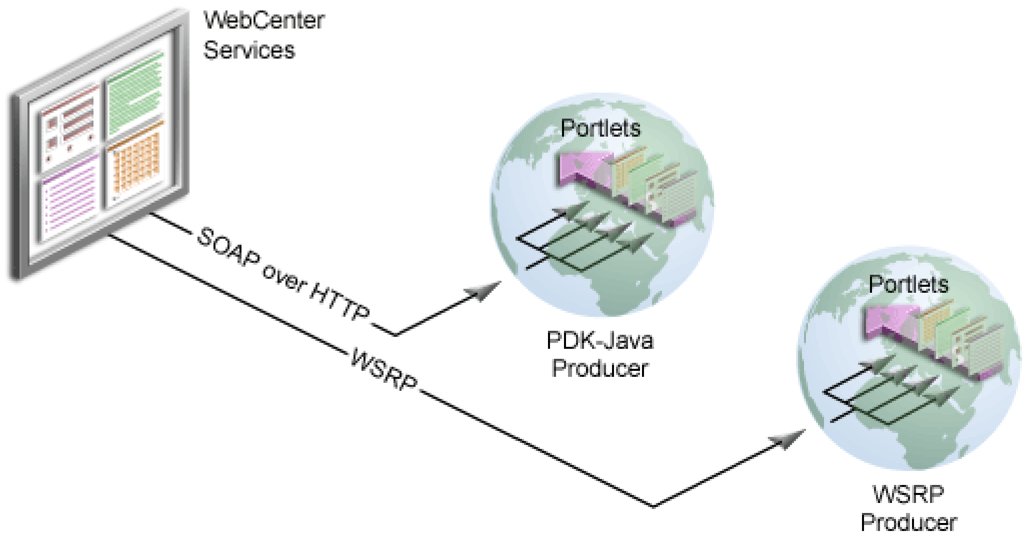
Before a portlet can be consumed by an application, you must first deploy it, then register the producer you've deployed the portlet to. As shown in Figure 27–7, portlets can be deployed through the following two producer types:

- PDK-Java producers
- WSRP producers

Within WSRP, both version 1.0 and 2.0 are supported.

PDK-Java portlets are deployed to a Java EE application server, which is often remote and communicates with the consumer through Simple Object Access Protocol (SOAP) over HTTP. JSR 168 portlets are deployed to a WSRP producer, which is also remote and communicates with the consumer through WSRP (Web Services for Remote Portlets). JSF Portlets are deployed as JSR 168 portlets, that is, to a WSRP producer, either from Oracle JDeveloper or manually.

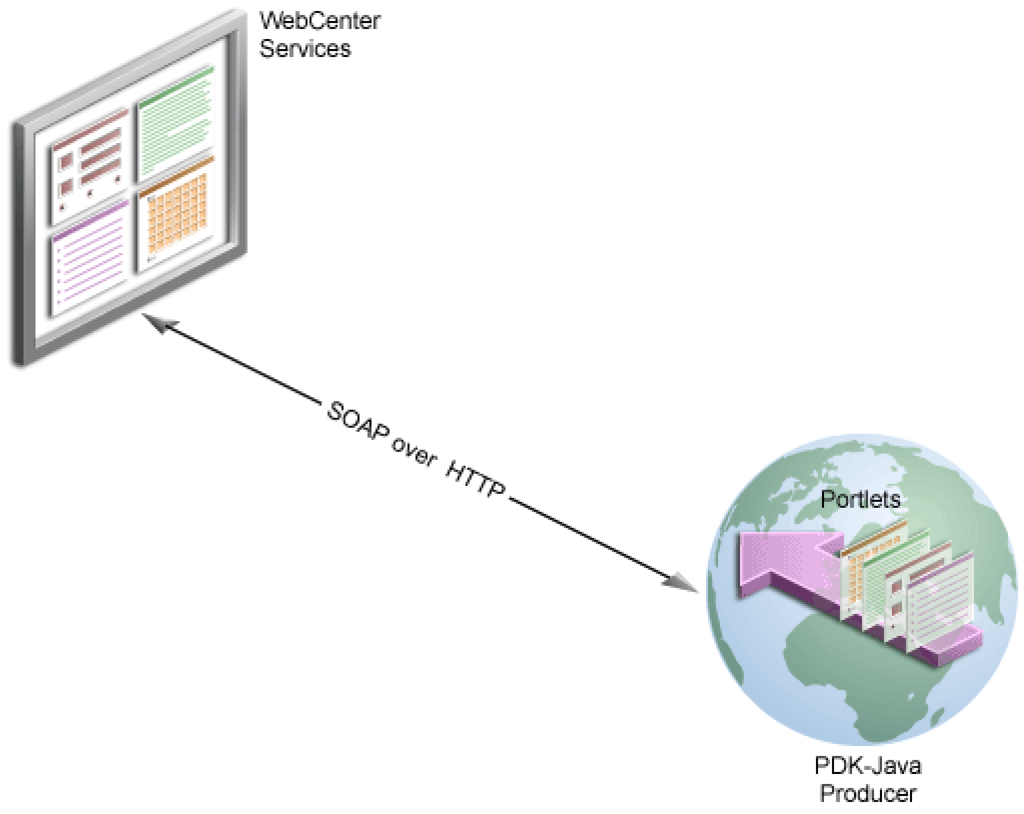
Figure 27–7 Portlet Producer Overview



27.2.3.1 PDK-Java Producers

PDK-Java producers use open standards, such as XML, SOAP, HTTP, or Java EE for deployment, definition, and communication with applications. Figure 27–8 shows how Oracle Portal incorporates portlets from a PDK-Java producer and the PDK-Java producer communicates with custom WebCenter application using SOAP over HTTP.

Figure 27–8 PDK-Java Producers



There are several benefits to developing portlets and exposing them through PDK-Java producers as follows:

- Deploy portlets remotely.
- Leverage existing Web application code to create portlets.
- Specify producers declaratively.
- Use standard Java technologies (for example, servlets and JSPs) to develop portlets.

To expose your portlets using a PDK-Java producer, you must first create a producer that manages your portlets and communicates with Oracle WebCenter Framework using SOAP. To learn how to expose your portlets using a PDK-Java producer, see Section 29.2, "Creating Java Portlets."

27.2.3.2 WSRP Producers

Oracle WebCenter supports Web Services for Remote Portlets (WSRP) versions 1.0 and 2.0. WSRP 2.0 support is for a preliminary (that is, pre-production) version of WSRP 2.0. This emerging standard provides support for inter-portlet communication and export or import of portlet customizations.

Use of WSRP 2.0 requires use of Oracle-specific extensions. For example, portlets produced by WSRP 2.0 producers must be deployed to Oracle's container to take advantage of the benefits this newer version provides. This is because standard portlet APIs (such as JSR 286) have not yet evolved to the level of the WSRP 2.0 communication protocol.

The Producer Registration wizard is the entry point for registering both WSRP 1.0 and 2.0 producers. The wizard automatically recognizes whether WSRP 1.0 or 2.0 is in play.

Architecturally, WSRP producers are very similar to PDK-Java producers. WSRP is a communication protocol between custom WebCenter application servers and portlet containers. WSRP is a standard that enables the plug-and-play of visual, user-facing Web services with 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 application that supports this standard.

Note: For more information about the WSRP architecture, see "The Relationship Between WSRP and JSR 168" in Chapter 29, "Creating Portlets with the Portlet Wizard."

To make standard portlets (such as JSR 168, .NET, Perl) available to a custom WebCenter application, you must package them in a portlet application and deploy them to a WSRP container. To learn more about WSRP, see the WSRP and JSR 168 Standards page on the Oracle Technology Network:

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

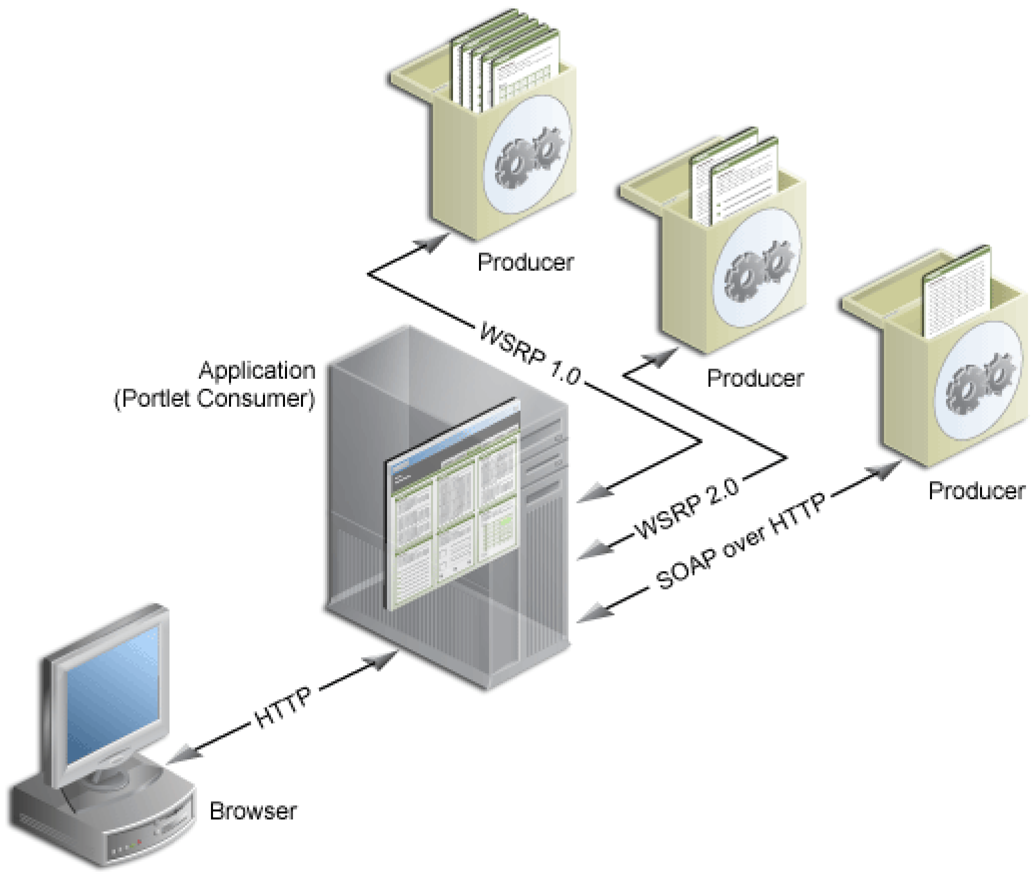
To learn how to package your portlets in a WSRP container, see Chapter 33, "Testing and Deploying Your Portlets." You can also test your WSRP producers online using the Oracle Portal Verification Service:

`http://portalstandards.oracle.com/portal/page/portal/OracleHostedWSRPPo` rtal/Welcome

27.2.3.3 Producer Architecture

Figure 27–9 illustrates the basic architecture of portlet producers.

Figure 27–9 Producer Architecture



When users display a page in their Web browsers, the flow of the request works as follows:

1. The user requests a page from the Web browser by entering a URL in the browser's address field.
2. The browser transmits the request to the application over HTTP.
3. The application contacts the portlet producers that provide the portlets that display on the requested page.
4. The producers 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 producers return the portlet content back to the application using their relevant protocols:
   - JSR 168 portlets and JSF portlets are initialized by WSRP producers, which communicate using the WSRP 1.0 or 2.0 protocol.
   - PDK-Java portlets are initialized by PDK-Java producers, which communicate using SOAP over HTTP.

Note: For more information about the portlet and producer architecture, visit the Portlet Development page on Portal Center (`http://portalcenter.oracle.com`).

JSF portlets, Web Clipping, OmniPortlet, and Java portlets communicate with Oracle WebCenter Framework through either WSRP or PDK-Java producers. You must register these producers with Oracle WebCenter Framework before you can use the portlets they produce in your custom WebCenter application.

The latest versions of Web Clipping and OmniPortlet are available through Application Development Runtime Service (ADRS). For more information, see Section 3.8, "Using Integrated WLS."

27.2.4 Caching Style

Portlet caching is key to rapid response to user requests. Portlets implement validation-based and expires-based caching using Java Object Cache. Invalidation-based caching continues to be implemented by a Web Cache that fronts the PDK-Java producer running the Web Clipping portlet and OmniPortlet.

Caching rules can be specified at a portlet's container level, encoded in the portlet's own logic, or, for JSR 168 portlets, established through the portlet wizard. Provided it is specified, container-level caching takes over when caching is not part of the portlet code.

At the application level, Oracle WebCenter Framework supports use of a Java cache for the establishment of application-level caching rules.

When not using caching, you may find accessing various data sources with Web Clipping and OmniPortlet to be time consuming. When you enable caching at the application level, you instruct the Java cache to maintain a copy of the portlet content. When data that was previously cached is requested, no time is lost in contacting the data source and regenerating its content. Instead, the previously cached portlet content is returned.

A portlet's content weighs heavily in determining the type of caching the portlet should use. For example:

- **Expiry-based caching**: Consider using 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:** Consider using validation-based caching 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.

27.2.4.1 Oracle JSF Portlet Bridge

Oracle JSF Portlet Bridge supports expiry- and validation-based caching. JSF portlets can be cached in full, or Edge Side Includes (ESI) can be used to cache fragments of portlets.

27.2.4.2 Web Clipping and OmniPortlet

In addition to invalidation-based caching, expiry-based caching can be specified for the Web Clipping portlet and OmniPortlet. Additionally, these portlets are refreshed automatically when they are personalized.

27.2.4.3 Programmatic Portlets

Java portlets support expiry- and validation-based caching. These portlets can be cached in full, or Edge Side Includes (ESI) can be used to cache fragments of portlets.

27.2.5 Development Tool

This section describes the development tools you can use to build different types of portlets.

27.2.5.1 Oracle JSF Portlet Bridge

Oracle JSF Portlet Bridge/JSF Portletization dialog is used to create a JSF portlet based upon a page or a task flow.

27.2.5.2 Web Clipping and OmniPortlet

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

27.2.5.3 Programmatic Portlets

Although you can use any Java development environment to build Java portlets, it is highly recommended that you use Oracle JDeveloper, a professional IDE. While you can consider other IDEs, Oracle JDeveloper 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 only add your own business logic to the skeleton. Oracle JDeveloper can also package and deploy your applications to your Java EE container. Also, Oracle JDeveloper helps you test your portlet producer. Oracle recommends that you use the Integrated WLS, provided through Oracle WebCenter Framework, as your development Java portlet runtime environment. For more information, see Chapter 3, "Preparing Your Development Environment."

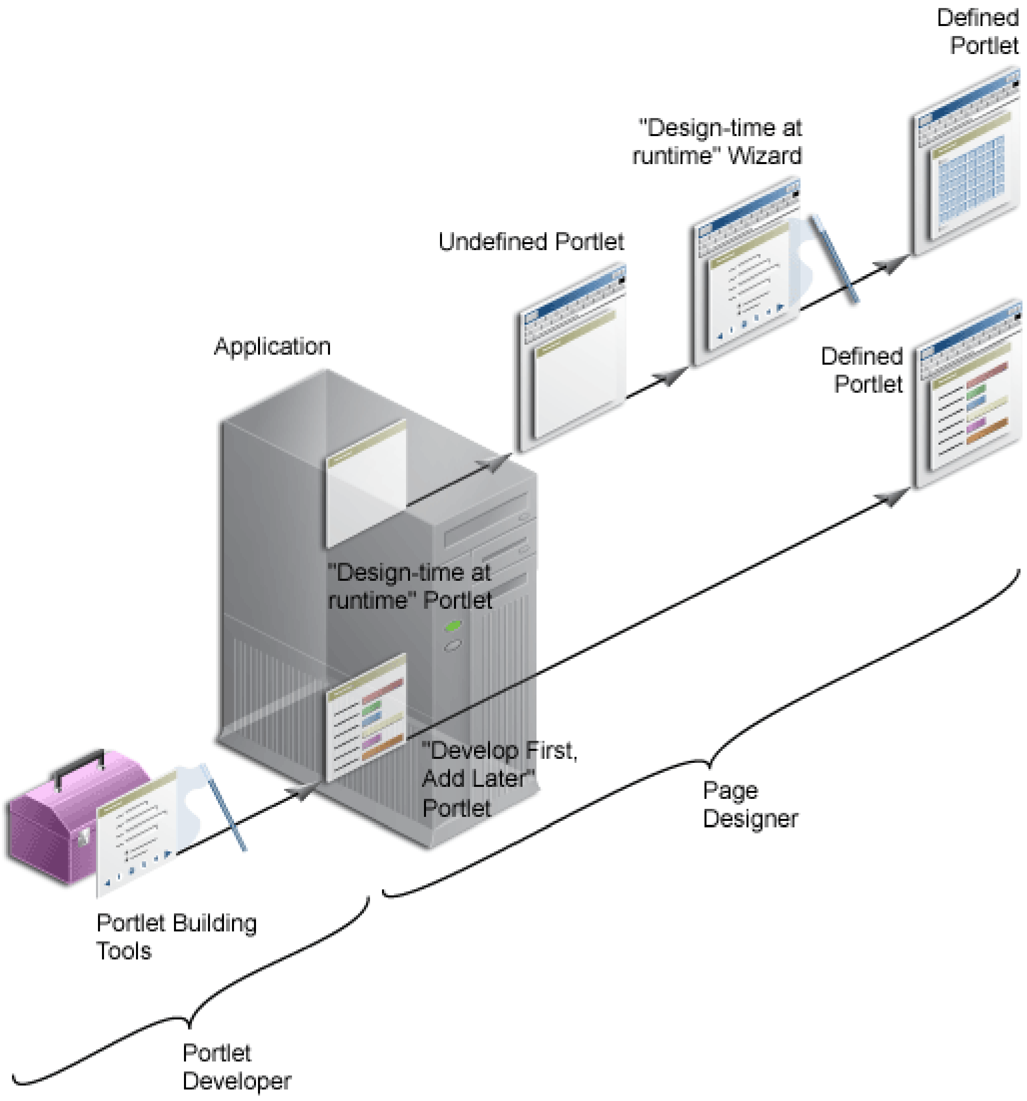
27.2.6 Portlet Creation Style

Oracle WebCenter Framework supports the following types of portlet creation (Figure 27–10):

- Develop first, add later
- Design time at runtime

Develop first, add later portlet creation is usually the task of the *portlet* developer; design time at runtime portlet creation is the *application* developer's responsibility.

Figure 27–10 Portlet Creation Style



27.2.6.1 Oracle JSF Portlet Bridge

Develop a JSF application first and then portletize the application using the JSF Portletization dialog at design time.

27.2.6.2 OmniPortlet and Web Clipping

OmniPortlet and Web Clipping both offer a "design time at runtime" portlet creation style. Register the portlet producers with the application that consumes the portlets, add the portlets to an application page, run the application, and then define the portlets in-place on the page.

27.2.6.3 Programmatic Portlets

Typically programmatic portlets offer a "develop first, add later" portlet creation style. Two wizards are available through Oracle WebCenter Framework to assist with the creation of Oracle PDK-Java and JSR 168 portlets. The wizards generate the basic files required for portlet creation. The developer hand-codes the portlet logic. The development sequence for programmatic portlets is to create the portlet, deploy it to a producer, register the producer with the application that consumes the portlet, and then add the portlet to an application page.

Note: With extensive coding, you can create "design time at runtime" Java portlets. For example, Web Clipping and OmniPortlet are both Java portlets.

27.2.7 User Interface Flexibility

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

27.2.7.1 Oracle JSF Portlet Bridge

Oracle JSF Portlet Bridge ensures that the portletized JSF application functions as a JSF portlet.

27.2.7.2 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.

27.2.7.3 OmniPortlet

OmniPortlet enables you to use different pre-built layouts, such as scrolling news, tabular, and chart. You can also use the built-in HTML layout to personalize the look and feel of your portlet using HTML and JavaScript.

Note: When using JavaScript in portlets, developers must ensure that the JavaScript identifiers are qualified. That is, identifiers must be unique for each portlet instance and must not clash with the JavaScript on the page.

27.2.7.4 Programmatic Portlets

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

27.2.8 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.

27.2.8.1 Oracle JSF Portlet Bridge

Oracle JSF Portlet Bridge's capability to capture contents from Web sites depends upon the JSF application being portletized.

27.2.8.2 Web Clipping

For portlets that display 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, then the Web Clipping engine can find the table again using the internal "fuzzy match" algorithm. Portlets built with Web Clipping can also maintain sessions to the remote Web sites. Web Clipping also supports user personalization of HTML form values.

27.2.8.3 OmniPortlet

For portlets using the data but not the layout from a remote Web site, the best choice is OmniPortlet. Use OmniPortlet 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. OmniPortlet is a powerful tool that extracts data from Web pages by using its Web Page data source.

27.2.8.4 Programmatic Portlets

Java portlets can take advantage of low-level Java networking APIs to retrieve and process content from remote Web sites. To avoid unnecessary development efforts, before choosing Java always ensure that Web Clipping or OmniPortlet are not viable options.

27.2.9 Ability to Render Content Inline

Active elements in portlets, such as links or form buttons, enable users to navigate to remote URLs. In a News portlet, for example, a user can click a hyperlink to navigate to a news site with detailed information about news of interest. For example, a user clicks a news-summary link in a News portlet, leaves the application page, and lands on the news site.

You may have a requirement to keep your users within the context of the application page by rendering the requested content within the same portlet container. For example, a user clicks a news-summary link in a News portlet, and the portlet refreshes with the detailed news article.

This maintenance of context is what rendering content inline means .

27.2.9.1 Oracle JSF Portlet Bridge

Oracle JSF Portlet Bridge converts a JSF application with multiple pages into JSF portlets. The contents of these pages are rendered inline.

27.2.9.2 Web Clipping

The Web Clipping portlet supports URL rewriting for achieving inline content rendering. It can process the links originating from the source Web site and rewrite them to achieve the desired functionality.

The following options are available:

- Select not to rewrite the URLs within the portlet, in which case clicking the links takes users out of the custom WebCenter application to the Web site that provides the clipping. Whenever the link brings the user to a place that requires authentication, the user must enter login information before the link target is displayed.
- If the Web Clipping provider is registered with an external application and the clipping requires authentication, then you can instruct Web Clipping to rewrite all URLs within the portlet to point to the Login Server. In this case, navigation causes the user to leave the custom WebCenter application, while also using the Login Server to log the browser into the External Application.
- Select to rewrite all URLs within the portlet (inline rendering) to point back to the page so that all browsing within the Web Clipping portlet remains within the custom WebCenter application. If the Web Clipping provider is registered with an

External Application, then this causes the Web Clipping provider to log itself in to the External Application. In this case, the navigation within the custom WebCenter application through the Web Clipping provider is authenticated in the External Application.

27.2.9.3 OmniPortlet

Rendering content inline is not supported, but you can achieve inline rendering using public portlet parameters.

27.2.9.4 Programmatic Portlets

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

If you use Struts in your portlet, then the PDK-Struts integration framework renders your content always in the same portlet container. Oracle recommends, however, that you use ADF Faces navigation for your new custom WebCenter application portlets.

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

27.2.10 Charting Capability

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

27.2.10.1 Oracle JSF Portlet Bridge

Oracle JSF Portlet Bridge does not have charting capability. However, it supports portletization of a JSF application that contains ADF charts.

27.2.10.2 Web Clipping

Web Clipping clips pre-existing content. So, while it does not create charts, it can retrieve and present HTML content that contains charts.

27.2.10.3 OmniPortlet

OmniPortlet supports bar, line, and pie chart types. Charts are dynamically generated images, which can include hyperlinks.

27.2.10.4 Programmatic Portlets

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

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

27.2.11 Public Portlet Parameter Support

Typically, a portlet's state is opaque (private); however, in Oracle WebCenter Framework portlets can describe public inputs (parameters) so values can be coordinated by the consuming application with other constituents of that application.

Inputs can include, public portlet parameters and private portlet parameters. These can be described as follows:

- **Public portlet parameters:** Use public portlet parameters to pass values to a portlet. Public parameters can assist with rendering portlet content that is specific to a particular page or user. Portlet parameters are created by the component developer and then exposed to the application developer through the user interface. After adding a portlet to a page, application developers can assign values to public portlet parameters to make the information displayed in the portlet specific to the page.

  Assigned values can be specific (such as a constant), a system variable (for example, the user name), or a page parameter. At runtime, the portlet receives the values from the sources specified.

- **Private portlet parameters:** Use private portlet parameters to implement internal navigation in a portlet. Parameters are passed to the portlet every time the page is requested. Private portlet parameters can be passed exclusively from the portlet instance to the same portlet instance. Private portlet parameters do not require a full page refresh.

Note: A Refresh action is available for inclusion on the portlet's Actions menu (`isNormalModeAvailable`). It refreshes the portlet without triggering a full page refresh. For details about setting this action programmatically, see Section 9.4, "Setting Attribute Values for the Portlet Tag."

Portlets supporting public portlet parameters enable application developers to tailor data input for each portlet instance. The component developer can focus on the portlet logic, while the application developer can address the interaction between the application page and its portlets.

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

27.2.12 Private Portlet Parameter Support

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

27.2.12.1 Oracle JSF Portlet Bridge

In your JSF portlets, you can implement internal navigation using private portlet parameters.

27.2.12.2 OmniPortlet and Web Clipping

With the OmniPortlet and Web Clipping portlets, component developers do not have access to private portlet parameters.

27.2.12.3 Programmatic Portlets

In your Java portlets, you can implement internal navigation using private portlet parameters.

27.2.13 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.

27.2.13.1 Oracle JSF Portlet Bridge

Oracle JSF Portlet Bridge supports the standard servlet mechanisms.

27.2.13.2 Web Clipping and OmniPortlet

Dynamically hide and show portlets built with Web Clipping and OmniPortlet by using security managers. Although Web Clipping and OmniPortlet do not expose security managers through the user interface, they make them available for editing through their XML provider definition files.

27.2.13.3 Programmatic Portlets

PDK-Java provides security managers for Java portlets. For example:

- **Group security manager:** The group security manager controls access to portlets based on group membership. For example, it shows the portlet to users who are members of a specified group, and hides the portlet from non-members.
- **Authentication level security manager:** The authentication level security manager controls access to the portlets based on authentication level. For example, it shows the portlet to authenticated users, and hides it from public users.

JSR 168 portlets support the standard servlet mechanisms.

27.2.14 Multilingual Support

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

Web Clipping, OmniPortlet, JSF portlets, and Java portlets display textual information in the language selected by the end user.

27.2.15 Pagination Support

Support for pagination is useful when a portlet must display a relatively large set of records.

27.2.15.1 Oracle JSF Portlet Bridge

Oracle JSF Portlet Bridge supports pagination, if the application portletized has pagination implemented in it.

27.2.15.2 Web Clipping

Web Clipping does not support pagination.

27.2.15.3 OmniPortlet

OmniPortlet does not support pagination.

27.2.15.4 Programmatic Portlets

With Java, portlet pagination can be implemented programmatically.

27.2.16 Authenticating to External Applications

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

27.2.16.1 Oracle JSF Portlet Bridge

Since any portletized application can function as a WSRP portlet, the external application support is feasible through custom user attributes.

27.2.16.2 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 must do is provide an External Application ID when registering the Web Clipping producer.

27.2.16.3 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 individually for each user. OmniPortlet can storing database credentials and HTTP basic authentication user name-password pairs. Credentials are stored in a secured metadata services repository.

27.2.16.4 Programmatic Portlets

Java portlets support programmatic integration with the external application framework and any LDAP server, such as Oracle Internet Directory.

28

Creating Portlets with the Oracle JSF Portlet Bridge

This chapter explains how to use the Oracle JSF Portlet Bridge to expose an application as a portlet.

This chapter includes the following sections:

- Section 28.1, "Introduction to the Oracle JSF Portlet Bridge"
- Section 28.2, "Creating a Portlet from a JSF Application"
- Section 28.3, "Creating JSF Portlets: Example"

28.1 Introduction to the Oracle JSF Portlet Bridge

The Oracle JSF Portlet Bridge allows application developers to quickly and easily expose their existing JSF applications, and Oracle ADF applications and task flows as JSR 168 portlets.

Note: Unless otherwise noted, the term JSF applications encompasses Oracle ADF applications as well.

The Oracle JSF Portlet Bridge:

- Simplifies portlet development by enabling you to express portlet functionality using JSF rather than relying on the JSR 168 portlet APIs.
- Simplifies exposing your JSF application to JSR 168 portlet consumers, such as Oracle Portal and Oracle WebCenter Spaces.
- Eliminates the need to store, maintain, and deploy your portlets separately from your application by enabling the application to run simultaneously as a regular web application and as a portlet from the same installation.
- Enables you to create portlets at a more granular level by exposing task flows as portlets. Because portletized task flows are WSRP portlets, this also enables you to use distributed task flows.

Note: The Oracle JSF Portlet Bridge is based on and conforms to JSR 301. JSR 301 is the standards effort to define the functionality for the Portlet 1.0 Bridge for JavaServer Faces. Oracle is the specification lead for this standard. More information is available at:

`http://www.jcp.org/en/jsr/detail?id=301`

28.2 Creating a Portlet from a JSF Application

The Oracle JSF Portlet Bridge enables you to expose a JSF application or task flow as a portlet. You do this declaratively, using the Create Portlet Entry dialog; no coding is required. Using the Create Portlet Entry dialog, you can configure Oracle JSF Portlet Bridge on a JSF application to expose the application as a JSR 168 portlet. As part of this configuration, you indicate the initial JSF view (or task flow view) that the Oracle JSF Portlet Bridge should invoke when the portlet is rendered. From that point on the Oracle JSF Portlet Bridge works with the JSF application to navigate through the additional views that are reachable from this initial view. So in the typical situation when you are exposing the entire JSF application as the portlet, you configure the Oracle JSF Portlet Bridge to render the application's initial view in the portlet and the rest of the navigation works naturally within that same portlet.

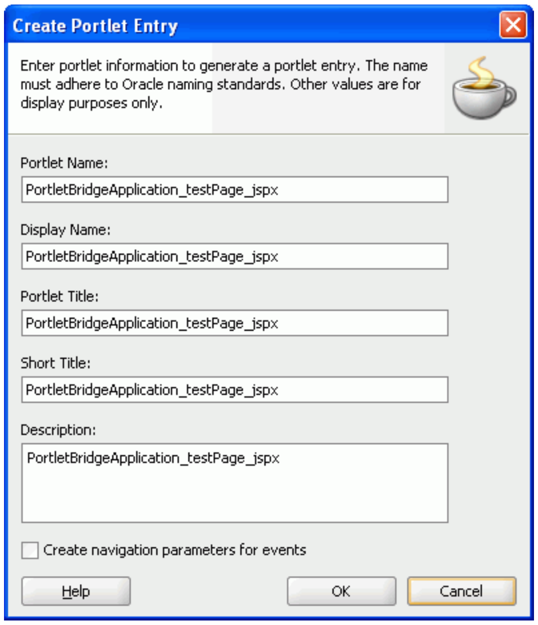
28.2.1 How to Create a JSF Portlet Based on a Page

The simplest way to create a portlet from a JSF application is to generate a portlet based upon a page.

To create a JSF portlet from an existing application page:

1. Start Oracle JDeveloper.
2. In the Application Navigator, open the application that contains the JSF page to portletize.
3. Right-click the page to portletize and choose **Create Portlet Entry.**
4. In the Create Portlet Entry dialog (Figure 28–1), in the **Portlet Name** field, enter a name for the portlet.

Figure 28–1 The Create Portlet Entry Dialog for a Page


Create Portlet Entry

Enter portlet information to generate a portlet entry. The name must adhere to Oracle naming standards. Other values are for display purposes only.

Portlet Name: PortletBridgeApplication_testPage_jspx

Display Name: PortletBridgeApplication_testPage_jspx

Portlet Title: PortletBridgeApplication_testPage_jspx

Short Title: PortletBridgeApplication_testPage_jspx

Description: PortletBridgeApplication_testPage_jspx

☐ Create navigation parameters for events

5. In the **Display Name** field, enter a descriptive name for your portlet.
6. In the **Portlet Title** field, enter a descriptive title for your portlet.

The portlet title is displayed in the Resource Palette or Application Resources panel, so make the title something to help users decide whether the portlet is useful to them. The portlet title is also displayed on the portlet header when the portlet appears on a page.

7. In the **Short Title** field, enter a shorter title for your portlet. This short title is displayed on the portlet header when the portlet appears on a page on a mobile device.
8. In the **Description** field, enter a description for your portlet
9. Select **Create navigation parameters for events** to create navigation parameters for any events exposed by the page.

   Navigation parameters enable a portlet to communicate with the page on which it resides and with other portlets on that page. If you select this option, navigation parameters are added to the `oracle-portlet.xml` file.
10. Click **OK.**

    The files `portlet.xml` and `oracle-portlet.xml` are created.

    The `portlet.xml` file contains the portlet entry (Example 28–1) and is opened ready for viewing or editing.

Example 28–1 Generated Portlet Entry for a Page

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<portlet-app version="1.0" xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/
  portlet-app_1_0.xsd http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd">
    <portlet id="adf_jsf__MyJSFPage_jspx">
        <description>myApp_myPage_jspx</description>
        <portlet-name>myApp_myPage_jspx</portlet-name>
        <display-name>myApp_myPage_jspx</display-name>
        <portlet-class>
         oracle.portlet.bridge.adf.application.ADFBridgePortlet
        </portlet-class>
        <init-param>
            <name>javax.portlet.faces.defaultViewId.view</name>
            <value>/myPage.jspx</value>
        </init-param>
        <supports>
            <mime-type>text/html</mime-type>
            <portlet-mode>VIEW</portlet-mode>
        </supports>
        <supported-locale>en</supported-locale>
        <portlet-info>
            <title>myApp_myPage_jspx</title>
            <short-title>myApp_myPage_jspx</short-title>
        </portlet-info>
    </portlet>
    <custom-portlet-mode>
        <portlet-mode>about</portlet-mode>
    </custom-portlet-mode>
    <custom-portlet-mode>
        <portlet-mode>config</portlet-mode>
    </custom-portlet-mode>
    <custom-portlet-mode>
        <portlet-mode>edit_defaults</portlet-mode>
    </custom-portlet-mode>
    <custom-portlet-mode>
        <portlet-mode>preview</portlet-mode>
    </custom-portlet-mode>
    <custom-portlet-mode>
        <portlet-mode>print</portlet-mode>
    </custom-portlet-mode>
</portlet-app>
```

The page you selected earlier is used as the entry point for the portlet View mode. This is indicated in the `portlet.xml` file by the `javax.portlet.faces.defaultViewId.view` initialization parameter. You can manually edit the `portlet.xml` file to define the pages for other default portlet modes:

- Edit mode: `javax.portlet.faces.defaultViewId.edit`
- Help mode: `javax.portlet.faces.defaultViewId.help`

Note: The value for the `defaultViewId` is relative to the application context root and should always start with a `/`. For example, in Example 28–1, the value for `defaultViewId.view` is `/myPage.jspx`.

If you add `defaultViewId` for other portlet modes, make sure you also add the mode to the `<supports>` tag. For example, `<portlet-mode>HELP</portlet-mode>`.

The `oracle-portlet.xml` file is an Oracle extension of `portlet.xml` to support WSRP 2.0 features such as navigation parameters used for inter-portlet communication. If you selected **Create navigation parameters for events,** navigation parameters are added to this file to contain the payload for any events exposed by the page.

28.2.2 How to Create a JSF Portlet Based on a Task Flow

An advantage of using Oracle ADF is task flows that provide a modular approach for defining control flow in an application. Instead of representing an application as a single large JSF page flow, you can break it up into a collection of reusable task flows. In each task flow, you identify application activities, the work units that must be performed in order for the application to complete. An activity represents a piece of work that can be performed when running the task flow.

Task flows can be unbounded or bounded:

- An unbounded task flow is a set of activities, control flow rules, and managed beans interacting to allow a user to complete a task. An unbounded task flow consists of all activities and control flows in an application that are not included within any bounded task flow.
- A bounded task flow is a specialized form of task flow, having a single entry point and one or more exit points. It contains its own set of private control flow rules, activities, and managed beans. An Oracle ADF bounded task flow allows reuse, parameters, transaction management, and reentry. It can have zero to many exit points.

A typical application is a combination of an unbounded and one or more bounded task flows. The application can then call bounded task flows from activities within the unbounded task flow. For more detailed information about bounded and unbounded task flows, refer to *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

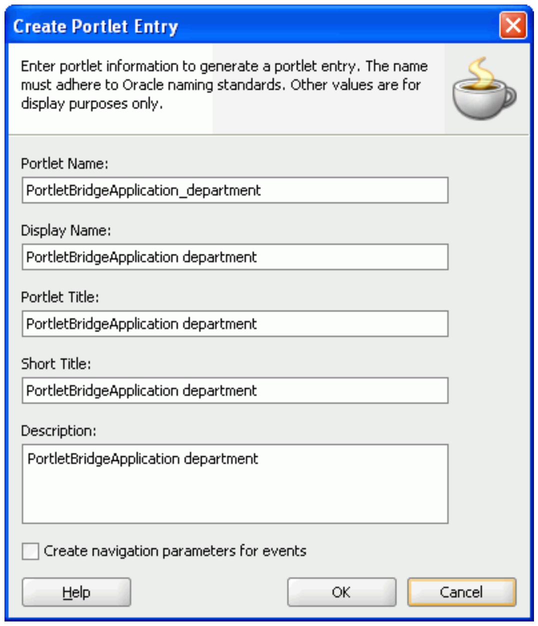
28.2.2.1 Creating a Portlet From a Task Flow Using the Create Portlet Entry Dialog

Use the Create Portlet Entry dialog to create a portlet from a single task flow.

To make a portlet from a task flow using the Create Portlet Entry dialog:

1. Start JDeveloper.
2. In the Application Navigator, open the JSF application that contains the task flow from which you want to make a portlet.
3. Right-click the task flow to portletize and choose **Create Portlet Entry.**
4. In the Create Portlet Entry dialog (Figure 28–2), in the **Portlet Name** field, enter a name for the portlet.

Figure 28–2 The Create Portlet Entry Dialog for a Task Flow



Create Portlet Entry

Enter portlet information to generate a portlet entry. The name must adhere to Oracle naming standards. Other values are for display purposes only.

Portlet Name: PortletBridgeApplication_department

Display Name: PortletBridgeApplication department

Portlet Title: PortletBridgeApplication department

Short Title: PortletBridgeApplication department

Description: PortletBridgeApplication department

☐ Create navigation parameters for events

5. From the Entry Point View list,

   If the task flow is bounded, you do not see the **Entry Point View** list.
6. In the **Display Name** field, enter a descriptive name for your portlet.
7. In the **Portlet Title** field, enter a descriptive title for your portlet.

   The portlet title is displayed in the Resource Palette or Application Resources panel, so make the title something to help users decide whether the portlet is useful to them. The portlet title is also displayed on the portlet header when the portlet appears on a page.
8. In the **Short Title** field, enter a shorter title for your portlet. This short title is displayed on the portlet header when the portlet appears on a page on a mobile device.
9. In the **Description** field, enter a description for your portlet.

10. Select **Create navigation parameters for events** to create navigation parameters for any events exposed by the task flow.

    Navigation parameters enable a portlet to communicate with the page on which it resides and with other portlets on that page. If you select this option, navigation parameters are added to the `oracle-portlet.xml` file.

11. Click **OK.**

    When your portlet has been created, you should receive a message that says:

    ```
    New portlet has been successfully created
    ```

    In addition, the files `portlet.xml` and `oracle-portlet.xml` are created. The `portlet.xml` file contains the portlet entry (Example 28–2) and is opened ready for viewing or editing.

Example 28–2 Generated Portlet Entry for a Task Flow

```
<portlet id="adf_taskflow_task-flow-definition">
  <description>task-flow-definition</description>
  <portlet-name>task-flow-definition</portlet-name>
  <display-name>task-flow-definition</display-name>
  <portlet-class>
    oracle.portlet.bridge.adf.application.ADFBridgePortlet
  </portlet-class>
    <init-param>
     <name>javax.portlet.faces.defaultViewId.view</name>
     <value>/adf.task-flow?_document=
      /WEB-INF/adfp-portlet-bridge-container.xml&amp;
      _id=adfp-portlet-bridge-container&amp;_fragmentTaskFlowDoc=
      /WEB-INF/task-flow-definition.xml&amp;_fragmentTaskFlowId=
      task-flow-definition
     </value>
    </init-param>
    <supports>
      <mime-type>text/html</mime-type>
      <portlet-mode>VIEW</portlet-mode>
    </supports>
    <supported-locale>en</supported-locale>
    <portlet-info>
      <title>task-flow-definition</title>
      <short-title>task-flow-definition</short-title>
    </portlet-info>
</portlet>
```

Note: The portlet.xml file can include multiple portlets and can have a combination of pages and task flows exposed as portlets.

The `oracle-portlet.xml` file is an Oracle extension of `portlet.xml` to support WSRP 2.0 features such as navigation parameters used for inter-portlet communication. If you selected **Create navigation parameters for events,** navigation parameters are added to this file to contain the payload for any events exposed by the task flow. If your task flow includes input parameters, corresponding portlet navigation parameters are created in `oracle-portlet.xml` to be used for inter-portlet communication.

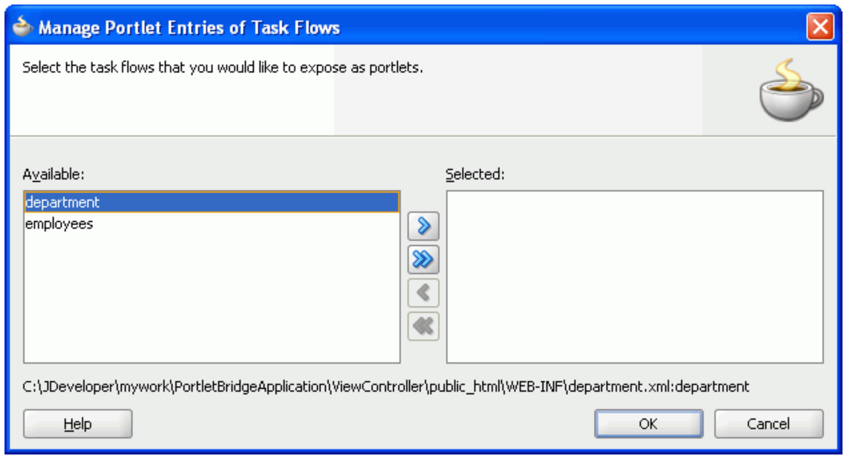
28.2.2.2 Creating a Portlet From a Task Flow Using the Manage Portlet Entries of Task Flows Dialog

If your project includes a lot of task flows, you may find it easier to select the task flows to create as portlets from a list. You can do this using the Manage Portlet Entries of Task Flows dialog. This dialog also lets you create portlets from multiple task flows at the same time, rather than having to create them individually.

To create a portlet from a task flow using the Manage Portlet Entries of Task Flows dialog

1. From the main menu, choose **File > New.**
2. In the New Gallery, expand **Web Tier,** select **Portlets** and then **Manage Portlet Entries of Task Flows,** and click **OK.**
3. In the Manage Portlet Entries of Task Flows dialog, use the shuttle buttons to select which task flows you want to create as portlets.

Figure 28–3 The Manage Portlet Entries of Task Flows Dialog



Select the task flows that you would like to expose as portlets.

department
employees

C:\JDeveloper\mywork\PortletBridgeApplication\ViewController\public_html\WEB-INF\department.xml:department

4. Click **OK** to create portlets for the selected task flows.
5. If the task flow exposes any events, you must manually add navigation parameters to the `oracle-portlet.xml` file to contain the payload for those events.

28.2.3 How to Test a JSF Portlet

When you have created your JSF portlet you can test it using the Integrated WebLogic Server that comes packaged with JDeveloper.

To test a JSF portlet:

1. From the main menu, choose **Run > Start Server Instance.**

   It may take a few moments for the Integrated WLS to start. When the instance has started, you should see a message similar to the following in your Log panel:

   ```
   DefaultServer started
   ```

   For more information, see Section 3.8, "Using Integrated WLS."

2. You are now ready to run or deploy your portlet application to the Integrated WLS. Because your web application and portlet application are one and the same

(the portlet application is your existing web application with additional portlet artifacts), you can run or deploy your web application as you normally would or you can run or deploy your portlet following the instructions in Section 33.2.1, "How to Test JSR 168 Portlets on Integrated WebLogic Server."

Note the distinction between Run and Deploy in the note in that section. Deploy provides a more persistent testing scenario.

3. Once deployed, you can view the Producer Test Page by going to:

   ```
   http://host:port/context-root/info
   ```

   Post deployment you must verify that the application works correctly as a web application before it is consumed as a portlet application. For example, verify that the page you portletized earlier works:

   ```
   http://localhost:7101/myApp-ViewController-context-root/faces/myPage.jspx
   ```

4. Once you have successfully deployed the application containing the portlet, you can register it as a portlet producer with any other application. For more information see Section 9.2.1, "How to Register a WSRP Portlet Producer."

   You can continue to access the application as a regular web application or consume it as a portlet producer.

5. Now that your portlet producer is deployed and registered, you can consume your JSF portlet as you would any other portlet. For more information, see Section 9.3, "Adding Portlets to a Page."

28.2.4 What Happens at Runtime

After having created and tested the JSF portlet you can deploy the application to your production environment.

> **Note:** Ensure that you deploy your application to a Java EE container with the Oracle Portlet Container installed. The Integrated WLS has the container already installed, which is why it is recommended for testing.

After successful deployment, you can access both the application and the portlet producer test page. For example, the application URL would be similar to:

```
http://host:port/appcontextroot/faces/pagename.jspx
```

And the portlet producer test URL would be similar to:

```
http://host:port/appcontextroot/info
```

For more information, see Section 9.2.1, "How to Register a WSRP Portlet Producer."

28.2.5 What You May Need to Know When Creating a JSF Portlet

You must code your JSF pages such that they produce markup that conforms with JSR 168 portlet markup fragment rules. If you have chosen to use Oracle ADF, the markup in your page comes mainly from the Oracle ADF Faces components, most of which naturally render markup in a style that is compatible with JSF portlets.

For those components that might cause problems in a portlet environment, the application developer must take special care. Some components generate markup that conflicts with the portlet environment and hence restricts their use. Other components may allow program control (inputs) that enable developers to introduce values that conflict with the portlet environment. In this latter case, you as the developer must be aware of the potential to publish the page as a portlet and therefore properly encode a value.

28.2.5.1 General Guidelines

The guidelines that follow lay out the issues of which you should be aware as you code Oracle ADF pages that you may later choose to publish as portlets.

- Prior to creating a portlet from your JSF application, the application, including all of its pages and task flows, must run properly after you deploy it to a standalone or Integrated WLS. If it does not run as a regular web application, it will not run as a portlet producer either.
- When writing an application that is supposed to run in both servlet and portlet environments, avoid casting to `HttpServlet` objects (or you get a `ClassCastException` when running as a portlet). Instead, use the abstraction provided by the Faces `ExternalContext` object. For example, instead of:

  ```
  HttpServletRequest request = getRequestFromFacesContext();
  String localContextPath = request.getContextPath();
  ```

  Use the following:

  ```
  String localContextPath=
  FacesContext().getCurrentInstance().getExternalContext().getRequestContextPath(
  );
  ```

  To the extent that `ExternalContext` does not provide that abstraction for you, you can write servlet or portlet specific code. You can use the following method to check if you are running as portlet:

  ```
  public static boolean isPortletRequest() {
      Map<String, Object> m =
          FacesContext.getCurrentInstance().getExternalContext().getRequestMap();
      Object phase = m.get("javax.portlet.faces.phase");
      if (phase != null) {
          return true;
      }
      else {
          return false;
      }
  }
  ```

- When consuming a JSF Portlet, the consumer application automatically renders the portlet content within an iframe. The implication is that any inline popup is then confined within the iframe. You should take this into consideration when specifying the size of the portlet.
- If your application is secured, ensure that you have secured identity propagation as described in Section 24.12, "Securing Identity Propagation Through WSRP Producers with WS-Security."
- Deep links are not directly supported. A by product of the JSR 168 portlet container implementation on WSRP is that session cookie management is proxied by the consuming application rather than the client. Portlets that deep link to their full service application usually rely on shared session state to allow the transition from the current portlet context. As most applications rely on maintaining session context with a cookie, the current architecture prevents such state sharing. A deep

link from the client directly to the producer server to invoke the application does not establish a session cookie between the consumer and the producer, hence a second session is established. Applications wanting to share such state must implement their own schemes for transferring the data between the two contexts. A common implementation is to write this state to a reachable location and pass a reference to this state in the deep link.

- Java EE login is not supported. Java EE applications can be configured with different authentication techniques, such as Basic and Digest. As portlets are fragments incorporated into a consumer's page, it is generally expected that direct authentication occurs between the client and the consumer, not the client and the portlet producer. As a result, these authentication techniques are not supported in JSR 168. In the JSR 168 case, Java EE authentication occurs through WS-Security mechanisms that allow the Web service consumer to be authenticated and verified by the producer, and propagate the user identity for user authentication and authorization. Published Oracle ADF artifacts should not contain login links that trigger Java EE authentication.
- For applications that take a long time to render, consider increasing the time out period when you register a producer that was created by the Oracle JSF Portlet Bridge. Note however that the time out period specified during producer registration is limited by the maximum time out period (`maximumTimeout` element) specified in `adf-config-xml`.

28.2.5.2 Portlet Guidelines

The portlet guidelines are as follows:

- For resources and links, you must specify the location relative to the `web-app-context-root`. Otherwise, your images and other resources cannot be found by the portlet. Do not use relative (../) path notation. Portlets in Oracle WebCenter Framework run remotely and are accessed using a SOAP protocol (WSRP). The latter means that the regular Web application concept of request path is meaningless in a JSR 168 container. The JSR 168 specification reflects this by mandating that all resource URLs either be absolute or context path relative.
- Do not redirect or forward a request within your JSP. JSR 168 only supports `requestDispatcher.include()`. The use of `httpServletResponse.sendRedirect()` or `requestDispatcher.forward()` results in exceptions and errors. To work properly in a portlet environment, you must implement JSF navigation rules in `faces-config.xml` or Oracle ADF task flow control flow rules.
- To minimize overall memory consumption in the application when running as a portlet, you should only store the minimal amount of data in the request scope.
- If you portletize an application that contains downloadable resources, the portlet container may rewrite the file name in such a way that it may be too long for the browser to open or save. In such a case, you can save the resource to a different name and open the file using the appropriate program directly.

28.2.5.3 Security Guidelines

The security guidelines are as follows:

- When you use the Create Portlet Entry or Manage Portlet Entries of Task Flows dialog to portletize a task flow in an Oracle ADF secured application (the security scheme being Authentication and Authorization), you must manually grant permission to the following wrapper task flow in the `jazn-data.xml` file of the producer application:

```
/WEB-INF/adfp-portlet-bridge-container.xml#adfp-portlet-bridge-container
```

If you do not grant the permissions, the portlet does not render.

For example, if you have an application with 2 roles: authenticated-role, with view permission; and testrole, with customize, edit, grant, personalize, and view permissions, you must add the following entries inside the `<permissions>` tag of each role:

– For authenticated-role:

```
<permission>
  <class>oracle.adf.controller.security.TaskFlowPermission</class>
  <name>
  /WEB-INF/adfp-portlet-bridge-container.xml#adfp-portlet-bridge-container
  </name>
  <actions>view</actions>
</permission>
```

– For testrole:

```
<permission>
  <class>oracle.adf.controller.security.TaskFlowPermission</class>
  <name>

/WEB-INF/adfp-portlet-bridge-container.xml#adfp-portlet-bridge-container
  </name>
  <actions>customize,edit,grant,personalize,view</actions>
</permission>
```

- If you are portletizing a task flow or page that has role based authorization (that is, the task flow or page has been granted certain roles), WS-Security is required to propagate the user correctly. If you set up WS-Security, the JSF portlet does not render the content due to lack of permissions.

  For information about setting up WS-Security on the producer, see "Securing a WSRP Producer with WS-Security" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

  When registering the producer with the consuming application, you must ensure that you set the appropriate security properties. For more information, see Section 9.2.1, "How to Register a WSRP Portlet Producer."

28.2.5.4 JSF Guidelines

The JSF guidelines are as follows:

- When using the `h:commandLink` JSF standard HTML component ensure that you set the following `context-param` in `web.xml` so that the JSF Reference Implementation does not generate any external JavaScript resource. This is to workaround an issue in the JSF Reference Implementation where the reference to the JavaScript resource is not properly encoded.

```
<context-param>
    <param-name>com.sun.faces.externalizeJavaScript</param-name>
    <param-value>false</param-value>
</context-param>
```

28.2.5.5 Oracle ADF Guidelines

The Oracle ADF guidelines are as follows:

- If you are portletizing an Oracle ADF task flow, make sure you are able to consume that task flow by dropping it onto a page as a region and running the page. This ensures that the task flow runs correctly before portletizing it.
- ADF Faces Dialog Framework is not supported. If your application includes any buttons or icons that launch secondary browser windows, the contents of the new windows are not displayed properly when the application is run as a portlet. As such, you should avoid using these components if you plan to portletize your application. Examples of components that launch secondary windows are:
  - `<tr:inputDate>`
  - `<tr:inputColor>`
  - `<tr:popup>`
  - the `useWindow` attribute of `<af:commandButton>`
- Portletization of pages that contain Oracle Composer components is not supported.
- Oracle ADF components/code that handle `prepareModel` must be idempotent. Any code executing during the `prepareModel` phase must be rerunnable without effect to the underlying data/business logic. When executing in the portlet environment, `prepareModel` is called twice during the complete JSF life cycle as opposed to being called once when executing within a regular Web application.

  The reason for this difference is that the Oracle JSF Portlet Bridge executes JSF in two requests not one. It is implemented as if every JSF request redirected before the render phase. The consequence of this approach is that the JSF `restoreView` phase is called both when the request is first submitted and then again when request to render is received.
- Do not access or reference request parameters from model code except in page parameters. Besides being a cleaner MVC2 implementation, following this guideline avoids problems when such artifacts are published as portlets. Where regular JSF artifacts run their entire lifecyle in a single request, the Oracle JSF Portlet Bridge executes these artifacts in two requests, as if every JSF request redirected before the render phase.

  This two phase model enables the clearing of submitted parameters before rendering in a manner that allows such clearing to be communicated all the way back to the client, which makes this request something you could bookmark. As a result, request parameters do not exist during the render phase. As described previously, `prepareModel` is invoked again in this rendering phase. Therefore, any references to request parameters in this phase handler fail. You should avoid code like either of the following fragments:

  ```
  <invokeAction id="doExecuteWithParams"
                Binds="ExecuteWithParams"
                Refresh="prepareModel"
                RefreshCondition="${param.id != null}"
  />

  <invokeAction id="doExecuteWithParams"
                Binds="ExecuteWithParams"
                Refresh="renderModel"
                RefreshCondition="${param.id != null}"
  />
  ```
- Do not reference page parameters in the `prepareModel` phase. This issue relates to the same problem just described for request parameters. Generally, page

parameters depend on request parameters and are evaluated during the `restoreView` phase. As this phase is called a second time during portlet rendering and request parameters are not available, such use fail. Instead, move any dependency on a page parameter value into the model before JSF transitions from its execute phases to its render phase.

- If you are portletizing an application that contains only Trinidad components (`<tr: >` tags only), you must manually include the following libraries in your project:
  - `jdeveloper\modules\oracle.adf.view_11.1.1\adf-richclient-api-11.jar`
  - `jdeveloper\modules\oracle.adf.view_11.1.1\adf-richclient-impl-11.jar`

  This is so that the URLs to icons in the style sheet generated for the producer are encoded correctly.

- Because a portlet is a naming container, special consideration should be taken when using ADF Faces client-side APIs to find components. An example component ID:
  - When run as a regular web application: `id="demoTemplate:popup"`
  - When run as a portlet application: `id="__ns12345678:demoTemplate:popup"`

  Things to watch out for specifically are:
  - Avoid using the `AdfPage.PAGE.findComponentByAbsoluteId()` API. Use `getSource()` and `findComponent()` methods instead. For example:

    ```
    <trh:script text="
    function showPopup(event) {
        event.cancel();
        // var popup =
            AdfPage.PAGE.findComponentByAbsoluteId("demoTemplate:popup");
        var source = event.getSource();
        var popup = source.findComponent("popup");
        popup.show({align:"after_end", alignId:"button"});
    }
    "/>
    ```

  - Use a relative path for clientId instead of an absolute path (starting with a '`:`'). For example, use:

    ```
    <af:showPopupBehavior popupId="demoTemplate:iteratorpop"
                triggerType="mouseHover"/>
    ```

    Instead of:

    ```
    <af:showPopupBehavior popupId=":demoTemplate:iteratorpop"
                triggerType="mouseHover"/>
    ```

  - For more information, see "What You May Need to Know About Naming Containers" in the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*.

28.3 Creating JSF Portlets: Example

The following section steps you through an example of creating JSF portlets. In the example, you will:

- Create an application that includes Oracle ADF task flows
- Turn the task flows into JSF portlets
- Create an application to consume the JSF portlets
- Wire the portlets using navigational parameters and ADFm events

28.3.1 How to Create the Example Application

The example application consists of a master task flow for entering a department number, and a detail task flow that displays the employees of the department provided in the master task flow. As you work through the example, remember to periodically save your progress.

To create the example application:

1. Start JDeveloper.
2. From the main menu, choose **File > New.**
3. In the New Gallery, expand **General,** select **Applications** and then **Fusion Web Application (ADF),** and click **OK.**
4. In the Name your application page of the Create Fusion Web Application (ADF) wizard, in the **Application Name** field, enter `PortletBridgeApplication` and then click **Finish.**
5. In the Application Navigator, right-click the **ViewController** project and choose **New.**
6. In the New Gallery, expand **General,** select **Java** and then **Java Class,** and click **OK.**
7. In the Create Java Class dialog, in the **Name** field, enter `Department`.
8. In the **Package** field, enter `hr` and click **OK.**
9. Replace the code for `Department.java` with that in Example 28–3:

Example 28–3 Department.java

```
package hr;
public class Department {
    private int deptno = 0;
    public Department() {
        super();
    }
    public void setDeptno(int deptno) {
        this.deptno = deptno;
    }
    public int getDeptno() {
      return deptno;
    }
}
```

10. Repeat steps 5 through 8 to create another Java class with the name `DepartmentBean`.

11. Replace the code for `DepartmentBean.java` with that in Example 28–4:

Example 28–4 DepartmentBean.java

```
package hr;
import java.util.Map;
import oracle.adf.share.ADFContext;
public class DepartmentBean {
    private static String DEPARTMENT_KEY = "HR_DEPARTMENT";
    private static String DEPARTMENT_DEFAULT_VALUE = "10";
    public DepartmentBean() {
        Department dept = new Department();
        dept.setDeptno(10);
        Map sessionScope = ADFContext.getCurrent().getSessionScope();
        if (sessionScope.get(DEPARTMENT_KEY) == null)
            sessionScope.put(DEPARTMENT_KEY, dept);
    }
    public int getDeptno() {
        Map sessionScope = ADFContext.getCurrent().getSessionScope();
        Department dept = (Department)sessionScope.get(DEPARTMENT_KEY);
        return dept.getDeptno();
    }
    public void setDeptno(int deptno) {
        Map sessionScope = ADFContext.getCurrent().getSessionScope();
        Department dept = (Department)sessionScope.get(DEPARTMENT_KEY);
        dept.setDeptno(deptno);
    }
    public String selectDepartmentString() {
        Map sessionScope = ADFContext.getCurrent().getSessionScope();
        Department dept = (Department)sessionScope.get(DEPARTMENT_KEY);
        String deptno =
            (dept == null) ? DEPARTMENT_DEFAULT_VALUE :
              String.valueOf(dept.getDeptno());
        System.out.println("DepartmentBean.selectDepartmentString:Department number is " + deptno);
        return deptno;
    }
}
```

12. Repeat steps 5 through 8 again to create another Java class with the name `Employee`.

13. Replace the code for `Employee.java` with that in Example 28–5:

Example 28–5 Employee.java

```
package hr;
public class Employee {
    private String firstName;
    private String lastName;
    private String title;
    private int deptno;
    public Employee(String firstName, String lastName, String title,
                    int deptno) {
        setFirstName(firstName);
        setLastName(lastName);
        setTitle(title);
        setDeptno(deptno);
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getFirstName() {
        return firstName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getTitle() {
        return title;
    }
    public void setDeptno(int deptno) {
        this.deptno = deptno;
    }
    public int getDeptno() {
        return deptno;
    }
}
```

14. Repeat steps 5 through 8 again to create another Java class with the name `EmployeesBean`.
15. Replace the code for `EmployeesBean.java` with that in Example 28–6

Example 28–6 EmployeesBean.java

```
package hr;
import java.util.ArrayList;
import oracle.adf.view.rich.context.AdfFacesContext;
public class EmployeesBean {
    private static final String DEPARTMENT_NUMBER_KEY = "DEPTNO";
    private static final int DEPARTMENT_NUMBER_NULL_VALUE = -1;
    private static Employee[] employees =
    { new Employee("Neil", "Russell", "Clerk", 10),
      new Employee("Terrence", "Bennett", "Manager", 10),
      new Employee("Blair", "Palmer", "VP", 10),
      new Employee("Cory", "O'Reilly", "Reporter", 20),
      new Employee("Tony", "McConnell", "Editor", 20),
      new Employee("Jennifer", "Kuffner", "VP", 30)
    };
    public EmployeesBean() {
    }
    public void setDepartmentNumber(String deptno) {
        selectDepartment(deptno);
    }
    public int findDepartmentValue(String defaultValue) {
        AdfFacesContext afContext = AdfFacesContext.getCurrentInstance();
        String deptno =
            (defaultValue == null ?
(String)afContext.getPageFlowScope().get(DEPARTMENT_NUMBER_KEY) :
             defaultValue);
        return (deptno == null ? DEPARTMENT_NUMBER_NULL_VALUE :
                Integer.valueOf(deptno));
    }
    public void selectDepartment(String deptno) {
        AdfFacesContext afContext = AdfFacesContext.getCurrentInstance();
        afContext.getPageFlowScope().put(DEPARTMENT_NUMBER_KEY, deptno);
    }
    public Employee[] getEmployees(String deptno) {
        int filterDeptno = findDepartmentValue(deptno);
        Employee[] filteredEmployees = null;
        if (filterDeptno == DEPARTMENT_NUMBER_NULL_VALUE) {
            filteredEmployees = employees;
        } else {
            ArrayList empsInDept = new ArrayList();
            for (int i = 0; i < employees.length; i++) {
                if (employees[i].getDeptno() == filterDeptno) {
                    empsInDept.add(employees[i]);
                }
            }

            filteredEmployees = new Employee[empsInDept.size()];
            empsInDept.toArray(filteredEmployees);
        }
        return filteredEmployees;
    }
}
```

16. In the Application Navigator, right-click **DepartmentBean.java** and choose **Create Data Control**.
17. Right-click **EmployeesBean.java** and choose **Create Data Control**.

    You should be able to see the two new data controls listed in the Data Controls panel.
18. Right-click the **ViewController** project and choose New.
19. In the New Gallery, expand **Web Tier,** select **JSF** and then **ADF Task Flow,** and click **OK.**
20. In the Create Task Flow dialog, in the **File Name** field, enter `department`.
21. Make sure **Create as Bounded Task Flow** and **Create with Page Fragments** are selected and click **OK.**

    Note: A task flow can raise ADFm events. When a task flow is turned into a portlet, all events that can be raised are exposed through navigational parameters. You can edit the associated `oracle-portlet.xml` file to remove any events that you do not want to be exposed.

22. In the Component Palette, select **View** from the list of activities, drag it onto `department.xml` and name it `department`.
23. Double click the view activity and click **OK** in the Create New JSF Page Fragment dialog to create the corresponding `department.jsff` page.
24. In the Data Controls panel of the Application Navigator, expand the **DepartmentBean** data control.
25. Select the **deptno** element and drag it onto `department.jsff`, choosing **Texts** and then **ADF Input Field w/Label** from the pop up menu.
26. Select the **selectDepartmentString()** method and drag it onto `department.jsff`, choosing **Methods** and then **ADF Button** from the pop up menu.

Note: While ADFm event payloads can be any object type, when you are exposing ADFm events through WSRP, only payloads of type String or a collection of strings can be supported.

27. Right-click **department.jsff** and choose **Go to Page Definition**.

    If you view the source code, you can see the `methodAction` created for the `selectDepartmentString()` method.

28. In the Structure panel of the Application Navigator, right-click **selectDepartmentString** and choose **Insert inside selectDepartmentString** and then **events.**

29. Right-click the **events** tag and chose **Insert inside events** and then **event**.

30. In the Insert event dialog, in the **name** field, enter `DepartmentSelectedEvent` and then click **OK.**

    This event is raised whenever this `methodAction` executes. This can happen through a UI action, such as clicking on the button associated with the `methodAction`, or through an `invokeAction` call to call this `methodAction` directly.

    You can create several event entries for a `methodAction` and all are raised when the `methodAction` fires.

    The payload for the event is the return value for the underlying method. If you examine at the `selectDepartmentString()` method in `DepartmentBean.java` (Example 28–4), you can see that it returns a String representation of the department number.

31. In the Application Navigator, right-click the ViewController project and choose **New.**

32. In the New Gallery, expand **Web Tier,** select **JSF** and then **ADF Task Flow,** and click **OK.**

33. In the Create Task Flow dialog, in the **File Name** field, enter `employees`.

34. Make sure **Create as Bounded Task Flow** and **Create with Page Fragments** are selected and click **OK.**

Note: A detail task flow can contain event consumers to handle ADFm events. However, when a task flow is exposed as a portlet, the only context that can be passed to the task flow is through portlet parameters which are mapped to task flow parameters.

35. In the Component Palette, select **View** from the list of activities, drag it onto `employees.xml` and name it `employees`.

36. Double click the view activity and click **OK** in the Create New JSF Page Fragment dialog to create the corresponding `employees.jsff` page.

37. In the Data Controls panel of the Application Navigator, expand the **EmployeesBean** data control.

38. Select the **getEmployees(String)** method and drag it onto `employees.jsff`, choosing **Methods** and then **ADF Button** from the pop up menu.

Note: We do not actually want the button on the page, but this is a simple way of creating a `methodAction` binding. You can remove the button after you've dropped it.

---

39. In the Edit Action Binding dialog, click **OK.**
40. Right-click **employees.jsff** and choose **Go to Page Definition**.

    If you view the source code, you can see the event consumer `methodAction` binding.

    When a `methodAction` binding receives an event, the underlying method is called with the parameters for the event passed to the parameters for the method.

Note: The `methodAction` binding can be both an event producer and an event consumer. For a `methodAction` to be an event producer, it must have an `event` tag defined in the metadata, like the one you created earlier.

All `methodAction` bindings are event consumers and require no additional metadata.

---

41. In the source code of the `employees.jsff` page, delete the button that you added earlier.

```
<af:commandButton actionListener="#{bindings.getEmployees.execute}"
                  text="getEmployees"
                  disabled="#{!bindings.getEmployees.enabled}" id="cb1"/>
```

    Deleting the button from the source view means the `methodAction` is not deleted from the page definition.

42. In the Data Controls panel of the Application Navigator, expand the **getEmployees(String)** method.
43. Select **Employee** and drag it onto `employees.jsff`, choosing **Tables** and then **ADF Read-only Table** from the pop up menu.
44. In the Edit Table Columns dialog, click **OK.**
45. In `employees.xml`, click the **Source** tab.
46. Create a `managed-bean` entry for the `EmployeesBean` class.

Example 28–7 EmployeesBean Managed Bean

```
<managed-bean>
  <managed-bean-name>contextProvider</managed-bean-name>
  <managed-bean-class>hr.EmployeesBean</managed-bean-class>
  <managed-bean-scope>pageFlow</managed-bean-scope>
</managed-bean>
```

47. Create an `input-parameter-definition entry` and map the value of the parameter to the appropriate `EmployeesBean` method to store this value.

    This is needed so that the task flow can consume WSRP navigational parameters when it is exposed as a portlet. When you turn this task flow into a portlet, the task flow parameter is converted into a portlet parameter.

    The code should now look like that in Example 28–8:

Example 28–8 Employees.xml

```
<?xml version="1.0" encoding="windows-1252" ?>
<adfc-config xmlns="http://xmlns.oracle.com/adf/controller" version="1.2">
  <task-flow-definition id="employees">
    <default-activity>employees</default-activity>
    <input-parameter-definition>
      <description>Main context parameter</description>
      <display-name>Department Number</display-name>
      <name>deptno</name>
      <value>#{pageFlowScope.contextProvider.departmentNumber}</value>
      <class>java.lang.String</class>
    </input-parameter-definition>
    <managed-bean>
      <managed-bean-name>contextProvider</managed-bean-name>
      <managed-bean-class>hr.EmployeesBean</managed-bean-class>
      <managed-bean-scope>pageFlow</managed-bean-scope>
    </managed-bean>
    <view id="employees">
      <page>/employees.jsff</page>
    </view>
    <use-page-fragments/>
  </task-flow-definition>
</adfc-config>
```

48. Before you can turn your task flows into portlets, you must first ensure that they work within the original application. To quickly test this, you can create a page on which you can place the page fragments.

    In the Application Navigator, right-click the **ViewController** project and choose **New.**

49. In the New Gallery, expand **Web Tier,** select **JSF** and then **JSF page,** and click **OK.**
50. In the Create JSF Page dialog, in the **File Name** field, enter `testPage.jspx`.
51. Select **Create as XML Document (*.jspx)** and click **OK.**
52. In the Application Navigator, expand the Page Flows node.
53. Drag the **department** task flow onto the page and select **Region** from the pop up menu.
54. Drag the **employees** task flow onto the page and select **Region** from the pop up menu.
55. In the Edit Task Flow Binding dialog, in the **Value** field for the deptno input parameter, enter `${'10'}` and click **OK**.
56. In the Application Navigator, right-click **testPage.jspx** and choose **Go to Page Definition**.
57. In the Structure panel of the Application Navigator, right-click **testPagePageDef** and choose **Edit Event Map**.
58. In the Event Map Editor dialog, click the Add a New Event Entry icon.
59. In the Add a New EventMap Entry dialog, in the **Producer** field, drill down to select **departmentPageDef.selectDepartmentString**.
60. In the **Event Name** field, make sure **DepartmentSelectedEvent** is selected.
61. In the **Consumer** field, drill down to select **employeesPageDef.getEmployees**.
62. In the Consumer Params section, click the Add Consumer Parameter icon.

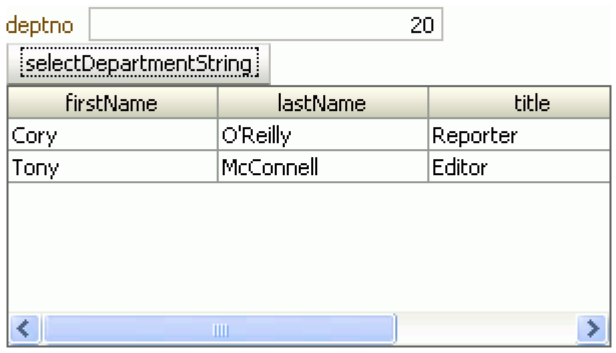
63. In the **Param Name** field, enter `department`.
64. In the **Param Value** field, enter `${payLoad}`.

> **Note:** The parameter name is not actually used. You can call your parameter anything. ADFm eventing just uses the position to pass these parameter values to the input parameters of the underlying `methodAction` method.
>
> `payLoad` is a keyword. If you invoked the expression building when in this field, you can also see `payLoad` as one of the options to choose. In this example, the payload is a scalar value (the department number as a string). If it was an object, you can further dereference it or pass the object as a whole. You can pass any valid EL for the value.

65. Click **OK** to close the Add a New EventMap dialog.
66. Click **OK** to close the Event Map Editor dialog.
67. In the Application Navigator, right-click **testPage.jspx** and choose **Run.**

    You should be able to enter a value of 10, 20, 30 in the **deptno** field of the master department task flow and see the employees task flow update accordingly when you click **getEmployees** (Figure 28–4).

Figure 28–4 Example Application Running as a Web Application



firstName	lastName	title
Cory	O'Reilly	Reporter
Tony	McConnell	Editor

28.3.2 How to Create Portlets from the Task Flows in the Example Application

After creating the example application and verifying that it runs correctly, you can create portlets from the task flows so that they can be consumed in other applications.

To create portlets from task flows:

1. In the Application Navigator, under the Page Flows node, right-click the **department** task flow and choose **Create Portlet Entry**.
2. In the Create Portlet Entry dialog, you can accept the default values for the name, title, and description fields, but ensure that you select the **Create navigation parameters for events** check box before you click **OK**.

   Selecting this check box creates a navigation parameter for the event raised by the task flow. When the event is raised by the task flow, the payload for the event is passed to the navigation parameter.
3. In the generated `portlet.xml` file, you can see the `portlet-class`, which shows that this is an ADFBridgePortlet, and the `init-param` that enables you to

access the portlet directly. Otherwise this file contains the standard WSRP information and nothing about the ADFm event that is in the task flow.

4. In the Application Navigator, right-click **oracle-portlet.xml** and choose **Open**.

   In this file you can see the `navigation-parameter` created for the ADFm event. It is exposed via the bridge as `_adf_event_DepartmentSelectedEvent`. When this event is raised by the task flow, the value of this parameter contains the payload for the event. The portlet binding code automatically raises this ADFm event on the consumer side which it is received.

5. Right-click the **employees** task flow and choose **Create Portlet Entry**.
6. You can accept the default values for the name, title, and description fields, and click **OK.**

   You have now made both of your task flows available as portlets from within this application.

7. Right-click **testPage.jspx** and choose **Run.**

   Running the application has an implicit deployment. When you deploy an application containing `portlet.xml`, WebCenter automatically generates the appropriate WSRP entry point for the application. So, after you run the page, you can access the application through the WSRP entry point and through HTTP.

8. To access the WSRP Producer Test Page for your portlets, when your application comes up in your browser, add `/info` after the context root for your application (typically, this means replacing the end of the URL, starting with `/faces/...`, with `/info`).
9. To check that the WSDL is accessible, click **WSRP v2 WSDL** in the WSRP Producer Test Page.

   Copy the WSDL URL of this page as you will need it later to register the producer.

28.3.3 How to Consume the JSF Portlets from the Example Application

Now that you have created the portlets, you can use them in another application.

To consume JSF portlets:

1. From the main menu, choose **File > New**.
2. In the New Gallery, expand **General,** select **Applications** and then **WebCenter Application,** and click **OK.**
3. In the Name your application page of the Create WebCenter Application wizard, in the **Application Name** field, enter `PortletBridgeConsumerApplication` and then click **Finish.**
4. Right-click the **ViewController** project and click **New**.
5. In the New Gallery, expand **Web Tier**, select **JSF** and then **JSF Page**, and click **OK.**
6. In the Create JSF Page dialog, in the **File Name** field, enter `PortletsRaisingEvents.jspx`.
7. Select **Create as XML Document (*.jspx)** and click **OK**.

   This is the page on which you will place the portlets. Before place portlets on a page, you must register the portlet producer with the application.

8. In the Application Resources panel of the Application Navigator, right-click **Connections**, choose **New Connection** and then choose **WSRP Producer.**

9. In the Specify Producer Name page of the Register WSRP Portlet Producer wizard, in the **Producer Registration Name** field, enter `PortletBridgeTestProducer` and click **Next.**
10. In the Specify Connection Details page, in the **WSDL URL** field, enter the URL you copied at the end of the previous exercise then click **Next.**
11. In the Specify Additional Registration Details page, click **Finish.**
12. In the Application Resources panel, expand the **Connections** node and then **WSRP Producer.**

    You can see the new WSRP producer. Expand this to see the portlets that you created from the department and employees task flows.
13. Select the **department** portlet and drag it onto the PortletsRaisingEvents page.
14. Select the **employees** portlet and drag it onto the page, just below the department portlet.
15. Right-click **PortletsRaisingEvents.jspx** and choose **Go to Page Definition.**

    In the source code you can see an entry for department portlet that includes the event that you defined for the task flow. This event is raised whenever the underlying task flow raises the event.

    You can also see an entry for the employees portlet that includes the parameter that you defined for the task flow. This is used to pass data from anywhere on the page back through the portlet to the task flow.

    The page definition also includes a variable iterator that defines a page variable that was created because a portlet with navigational parameters was placed on the page. Portlets or components on the page that do not create ADFm events can pass context to portlets using this page variable.
16. Add a default value for the page variable as shown in Example 28–9

Example 28–9 Default Value for Deptno Page Variable

```
<variableIterator id="variables">
  <variable Name="PortletBridgeApplicationemployees1_1_deptno"
            Type="java.lang.Object
            DefaultValue="${'20'}"/>
</variableIterator>
```

17. In the Structure panel, right-click **PortletsRaisingEventsPageDef** and choose **Edit Event Map.**
18. In the Event Map Editor dialog, click the Add a New Event Entry icon.
19. In the Add New EventMap Entry dialog, in the **Producer** field, drill down to select **PortletBridgeApplicationdepartment1_1**.
20. In the **Event Name** field, make sure **DepartmentSelectedEvent** is selected.
21. In the **Consumer** field, drill down to select **PortletBridgeApplicationemployees1_1.**
22. In the Consumer Params section, click the Add Consumer Parameter icon.
23. In the **Param Name** field, enter `deptno`.
24. In the **Param Value** field, enter `${payLoad}`.

Note: No payload or a null payload is propagated across the wire from the event producer portlet to the event consumer as an empty string. If the consumer must differentiate between an empty string and null payload, you can encode the null value in the payload in the event producer. The event consumer must also look for this custom encoding to detect the null payload.

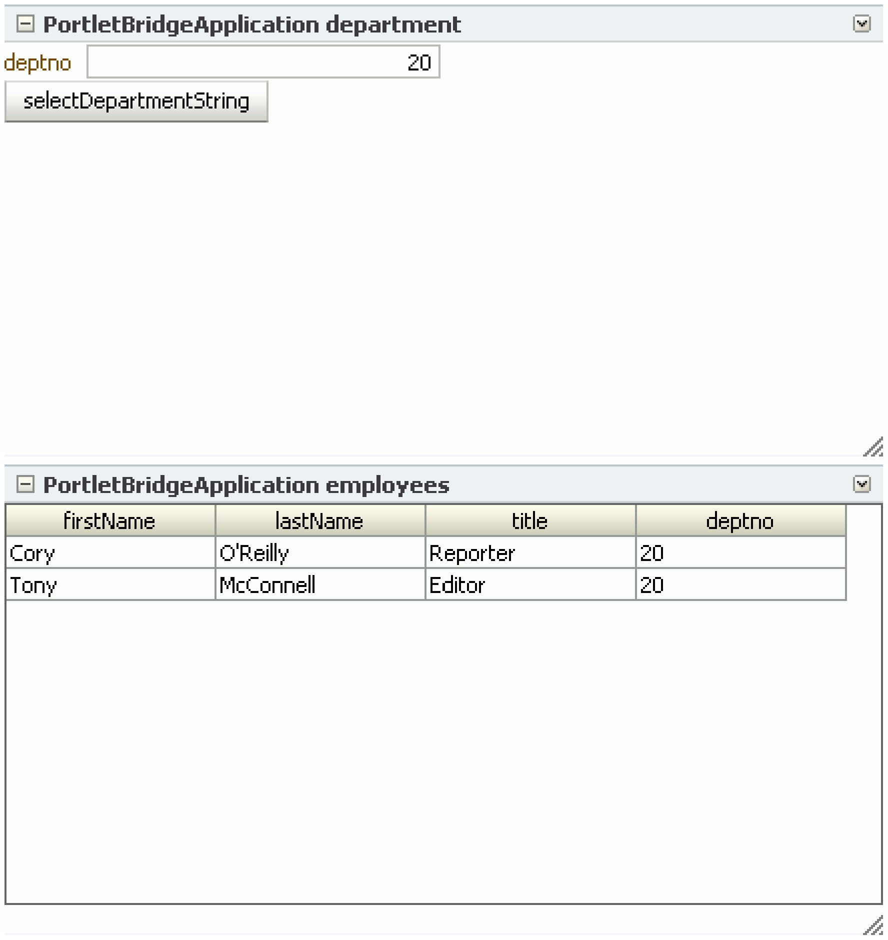
25. Click **OK** to close the Add New EventMap Entry dialog.
26. Click **OK** to close the Event Map Editor dialog.

    You should now be able to see a new event map entry in the source code of the page definition.
27. In the PortletsRaisingEvents page, select the employees portlet.
28. In the Property Inspector, expand the **Common** tab if necessary, and in the **PartialTriggers** field, enter `portlet1` (the ID of the department portlet).

    This makes sure that the employees portlet refreshes to reflect any changes made in the department portlet (portlet1).
29. In the Application Navigator, right-click **PortletsRaisingEvents.jspx** and choose **Run.**
30. In the **deptno** field of the department portlet, enter `20` and click **selectDepartmentString**.

    The employees portlet refreshes to show the employee details for department 20 (Figure 28–5).

Figure 28–5 Portletized Task Flows in an Application


PortletBridgeApplication department
deptno
selectDepartmentString
PortletBridgeApplication employees

deptno: 20

firstName	lastName	title	deptno
Cory	O'Reilly	Reporter	20
Tony	McConnell	Editor	20

29 Creating Portlets with the Portlet Wizard

This chapter explains how to create Java portlets based on the Java Portlet Specification (JSR 168) or the Oracle Portal Developer Kit-Java (PDK-Java) in a WebCenter portlet application, using the Create JSR 168 Java Portlet and Create Oracle PDK-Java Portlet wizards.

This chapter includes the following sections:

- Section 29.1, "Introduction to Java Portlets"
- Section 29.2, "Creating Java Portlets"

29.1 Introduction to Java Portlets

This section includes the following sections:

- Section 29.1.1, "Introduction to Standards-Based Java Portlets"
- Section 29.1.2, "Introduction to PDK-Java Portlets"

29.1.1 Introduction to Standards-Based Java Portlets

Organizations engaged in custom WebCenter application 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 changed with the introduction of the following standards:

- Web Services for Remote Portlets (WSRP)
- Java Specification Request (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 custom WebCenter applications.

WSRP is a Web services standard that enables 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 the following:

- Web Services Definition Language (WSDL) interface for the invocation of WSRP services
- Markup fragment rules for markup emitted by WSRP services
- The methods to publish, find, and bind WSRP services and metadata

JSR 168 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. JSR 168 defines container services which provide the following:

- 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 JSR 168.

Note: HTML forms can be submitted using either the `get` or `post` method, but the WSRP standard only requires the consumer (custom WebCenter application) to use the `post` method. Support of the `get` method is optional according to the standard. Since application consumers are not required to support the `get` method, Oracle recommends that you use the `post` method when developing your portlets.

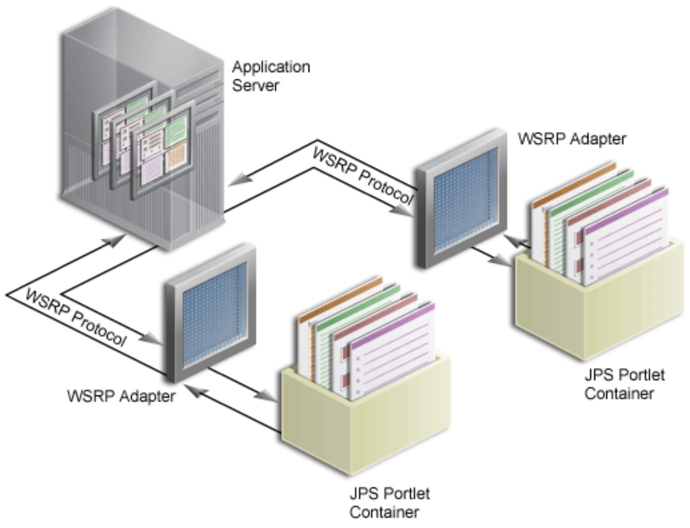
The Relationship Between WSRP and JSR 168

WSRP is a communication protocol between custom WebCenter application servers and portlet containers, while JSR 168 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 pages becomes as simple as selecting portlets from the JDeveloper Component Palette.

Figure 29–1 shows the architecture of the WSRP specification.

Note: Figure 29–1 illustrates the use of JSR 168 portlets with WSRP, but it should be noted that WSRP can also work with non-JSR 168 portlets.

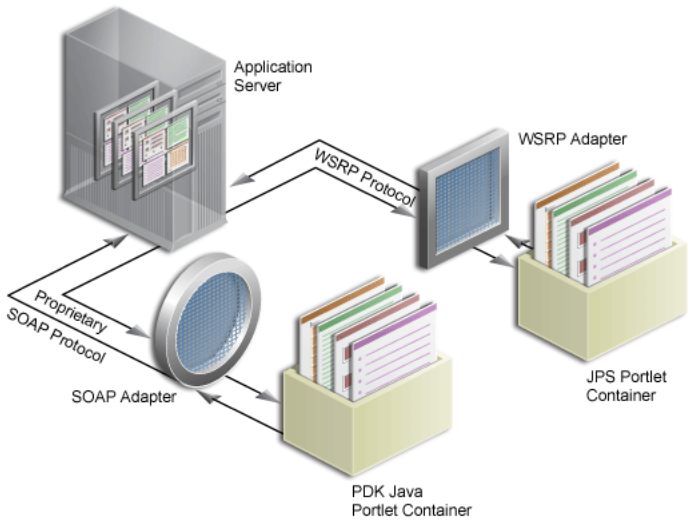
Figure 29–1 WSRP Specification Architecture



Oracle WebCenter Framework can support communication between the custom WebCenter application and both the new Java Portlet APIs and our existing APIs (PDK-Java).

Figure 29–2 shows the architecture of the WSRP support. Notice that the JSR 168-compliant portlet container uses the WSRP protocol for communication and the PDK-Java portlet container uses Oracle's proprietary SOAP protocol for communication.

Figure 29–2 Oracle WebCenter Portlet Architecture



For a description of how JSR 168 security concepts are exposed through WSRP, see Section 24.12, "Securing Identity Propagation Through WSRP Producers with WS-Security."

29.1.2 Introduction to PDK-Java Portlets

PDK-Java gives you a framework to simplify the development of Java portlets by providing commonly required utilities and enabling you to leverage existing development skills and application components such as JSPs, servlets, and static HTML pages. PDK-Java also enables you to create portlets without having to deal directly with the complexity of communications between Oracle WebCenter Framework and producers.

The PDK-Java framework is divided into the following areas:

- The **Producer Adapter** insulates the developer from the HTTP syntax defined by Oracle WebCenter Framework for communication with Web producers. It translates the information passed between Oracle WebCenter Framework and your Java Web producer. Without an adapter, your producer would not only manage portlets, but it would also have to communicate this information directly to Oracle WebCenter Framework in the expected language. The adapter eliminates the need for your Web producer to understand the portal language and vice-versa.
- The **Producer Interface** defines the APIs (functions) required by your Java implementation to integrate with the Producer Adapter. The Producer Adapter receives messages from the custom WebCenter application, translates them into calls to the Producer Interface, and translates the producer's response into a format that the application can understand. The Producer Interface contains a set of Java

classes that define the methods your producer implements and, in often, provides a standard implementation. Some of the primary classes are as follows:

- `ProviderDefinition` (`oracle.portal.provider.v2.ProviderDefinition`)
- `ProviderInstance` (`oracle.portal.provider.v2.ProviderInstance`)
- `PortletDefinition` (`oracle.portal.provider.v2.PortletDefinition`)
- `PortletInstance` (`oracle.portal.provider.v2.PortletInstance`)
- `ParameterDefinition` (`oracle.portal.provider.v2.ParameterDefinition`)
- `EventDefinition` (`oracle.portal.provider.v2.EventDefinition`)

- The **Producer Runtime** provides a base implementation that follows the specification of the Producer Interface. The Producer Runtime includes a set of default classes that implement each of the Producer Interfaces and enables you to leverage the rendering, personalization, and security frameworks provided with PDK-Java. These classes and the associated frameworks simplify the development of a producer by implementing common functions for Oracle WebCenter Framework requests and providing a declarative mechanism for configuring the producer. Using the Producer Runtime, you can focus your development efforts on the portlets themselves rather than the infrastructure needed to communicate with the custom WebCenter application. If the standard behavior of the Producer Runtime does not meet your requirements, then you can easily extend or override specific behaviors. Some of the primary classes are as follows:
  - `DefaultProviderDefinition` (`oracle.portal.provider.v2.DefaultProviderDefinition`)
  - `DefaultProviderInstance` (`oracle.portal.provider.v2.DefaultProviderInstance`)
  - `DefaultPortletDefinition` (`oracle.portal.provider.v2.DefaultPortletDefinition`)
  - `DefaultPortletInstance` (`oracle.portal.provider.v2.DefaultPortletInstance`)
  - `PortletRenderer` (`oracle.portal.provider.v2.render.PortletRenderer`)
  - `PortletPersonalizationManager` (`oracle.portal.provider.v2.personalize.PortletPersonalizationManager`)
  - `PortletSecurityManager` (`oracle.portal.provider.v1.http.DefaultSecurityManager`)
- The **Producer Utilities** provide methods for simplifying the rendering of portlets. The utilities include methods for constructing valid links (`hrefs`), rendering the portlet's container (including the header), rendering HTML forms that work within a page, and supporting portlet caching.

You can find the JavaDoc reference for the PDK-Java on OTN at:

`http://www.oracle.com/technology/products/webcenter/portlet_download.html`

29.2 Creating Java Portlets

Before you begin you should make sure:

- You are familiar with portlet terminology such as portlet modes. For more information, see Chapter 27, "Overview of Portlets" and Section 29.2.5, "What You May Need to Know When Creating Java Portlets."
- You are familiar with Oracle JDeveloper and know how to build and deploy Java components using it.
- The application in which you are going to create your JSR 168 and PDK-Java portlets is either a Portlet Producer Application or any application scoped for portlet creation (any application except for WebCenter Application).

Note: The figures in this section were taken with the default Look and Feel and Theme settings in JDeveloper (Oracle and Fusion Blue). If you have changed these settings, then what you see on your screen may vary slightly, but the content and functionality remains the same. To change the Look and Feel and Theme settings, select **Preferences** from the **Tools** menu, and then select **Environment.**

29.2.1 How to Create a JSR 168 Java Portlet

Using the Create JSR 168 Java Portlet wizard in JDeveloper you can expose your portlet over WSRP 2.0 quickly and easily. This wizard supports both WSRP 1.0 and WSRP 2.0. If you choose to create a WSRP 2.0 portlet, then an additional page appears in the wizard for the WSRP 2.0 enhancements.

In the Create JSR 168 Java Portlet wizard, you can choose which portlet modes you want to implement and the implementation method (JSP, HTTP servlet, Java class, or HTML) to use for each mode. The wizard then creates a simple implementation for each of the selected modes.

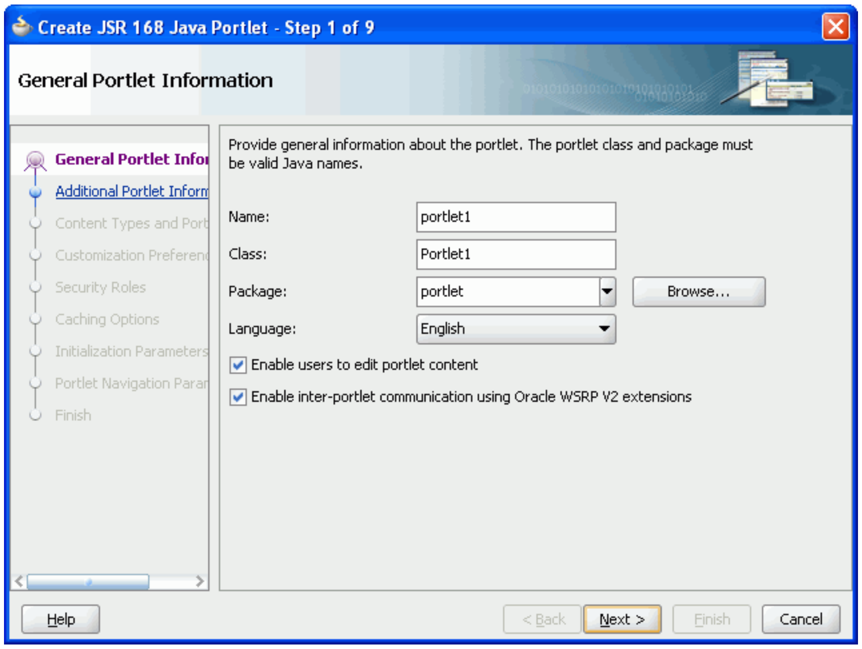
To create a JSR 168 Java portlet using the JDeveloper wizard:

1. Start JDeveloper.
2. In the Application Navigator, open the application under which you want to create your portlet.

   The application should be scoped for portlet creation. An easy way to achieve this is to use a Portlet Producer Application. WebCenter Applications are not scoped for portlet creation.
3. Right-click the project under which you want to create your portlet, and choose **New.**
4. In the New Gallery, expand **Web Tier,** select **Portlets** and then **Standards-based Java Portlet (JSR 168),** and click **OK.**

   Note: Selecting **Standards-based Java Portlet** opens the wizard for creating JSR 168-compliant portlets. Selecting **Oracle PDK-Java Portlet** opens the wizard for creating PDK-Java portlets.
5. In the General Portlet Information page of the Create JSR 168 Java Portlet wizard (Figure 29–3), replace the default name provided in the **Name** field with one that better describes the purpose of the portlet.

Figure 29–3 The General Portlet Information Page


Create JSR 168 Java Portlet - Step 1 of 9

General Portlet Information

Provide general information about the portlet. The portlet class and package must be valid Java names.

Name: portlet1
Class: Portlet1
Package: portlet
Language: English
[✓] Enable users to edit portlet content
[✓] Enable inter-portlet communication using Oracle WSRP V2 extensions

6. In the **Class** field, enter a name for the class for the portlet. You can accept the default name provided or supply your own. If you supply your own name, it must be a valid Java name.

7. From the **Package** list, select the package in which to create the class.

   Click the **Browse** button to find packages within the project, if required. If you do not select a specific package, the wizard uses the default package of the project.

8. From the **Default Language** list, select the default language that your portlet supports. The wizard uses English by default.

9. Select **Enable users to edit portlet content** if you want your portlet to support Edit mode. In the wizard, this option is selected by default.

   If you select this option, you can specify the details for the Edit mode later on in the wizard.

10. Select **Enable inter-portlet communication using Oracle WSRP V2 extensions** to create a portlet that supports Oracle WSRP 2.0 extensions and then click **Next.**

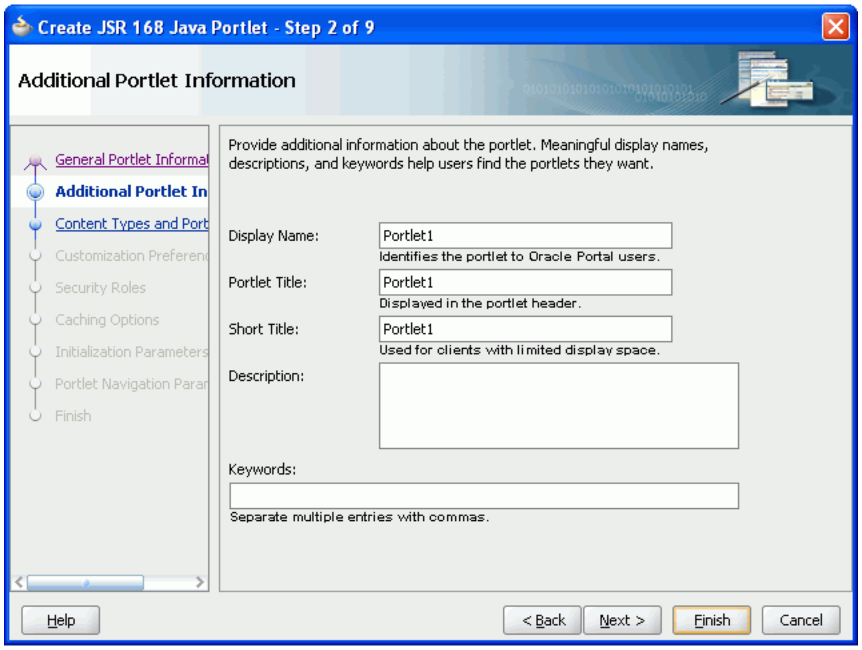
    Selecting this option creates the `oracle-portlet.xml` file that is used for WSRP 2.0 features, such as navigation parameters. The WSRP 2.0 standard extends WSRP 1.0 by including support for inter-portlet communication (through navigation parameters) and export and import of portlet customizations.

    Note: JSR 168 portlets built with the Oracle WSRP V2 extensions can be consumed by any consumer that supports WSRP 2.0.

11. In the Additional Portlet Information page (Figure 29–4), in the **Portlet Title** field, enter a descriptive title for your portlet.

    The portlet title is displayed in the Resource Palette or Application Resources panel, so make the title something to help users decide whether the portlet is useful to them. The portlet title is also displayed on the portlet header when the portlet appears on a page.

Figure 29–4 The Additional Portlet Information Page



Provide additional information about the portlet. Meaningful display names, descriptions, and keywords help users find the portlets they want.

Display Name: Portlet1
Identifies the portlet to Oracle Portal users.

Portlet Title: Portlet1
Displayed in the portlet header.

Short Title: Portlet1
Used for clients with limited display space.

Description:

Keywords:
Separate multiple entries with commas.

12. The display name, short title, description, and keyword attributes are not implemented in WebCenter applications. You do not need to enter any values for these fields unless your portlet is likely to be consumed by other applications. For example, Oracle Portal uses the portlet display name in the Portlet Repository.

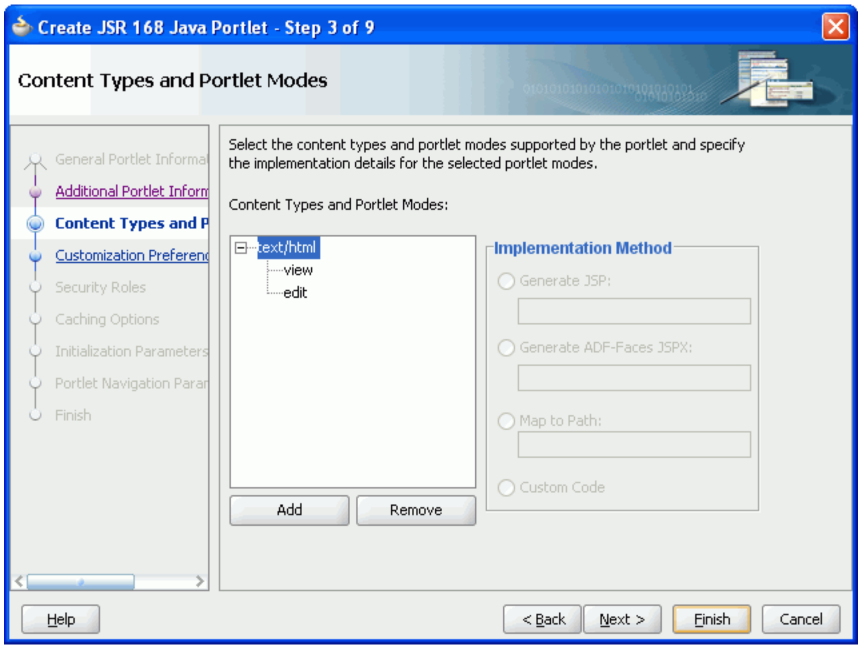
13. At this point in the wizard, you can click **Finish** to create the portlet immediately, using the default values for all remaining settings.

    To provide additional details for your portlet, click **Next** and follow the remaining steps.

14. In the Content Types and Portlet Modes page (Figure 29–5), to add a content type, select an existing content type (for example, **text/html**) and then click **Add.** In the Content Types dialog move the desired content types to the **Selected** list and then click **OK.**

    By default, your portlet displays its content using the `text/html` content type.

Figure 29–5 The Content Types and Portlet Modes Page



Select the content types and portlet modes supported by the portlet and specify the implementation details for the selected portlet modes.

Content Types and Portlet Modes:

- text/html
  - view
  - edit

15. To add another portlet mode to your portlet, select an existing mode (for example, **view**) under the appropriate content type and then click **Add.** In the Portlet Modes dialog, move the desired modes to the **Selected** list and then click **OK.**

By default, your portlet includes View mode. If you selected **Enable users to edit portlet content** on the General Portlet Properties page, your portlet also includes Edit mode. For more information about portlet modes, see Section 29.2.5.1, "Guidelines for Portlet Modes."

16. Select each of the portlet modes and specify the implementation method for rendering content.

The wizard generates a portlet dispatcher Java class and any other required files for each portlet mode. The exact nature of this class and any other generated files depends on the chosen implementation method:

- Select **Generate JSP** to generate a skeleton JSP file for the portlet mode. Enter a name for the JSP in the corresponding field, or accept the default.

  When you complete the wizard, the generated JSP displays in the Application Navigator where you can select it for further development. This is the default selection for all portlet display modes.

- Select **Generate ADF-Faces JSPX** to generate a page to which you can add ADF-Faces components. Enter a name for the JSPX in the corresponding field, or accept the default.

  This applies to all portlet modes. If any portlet mode uses this selection, the wizard generates a portlet application which uses the Oracle Portlet Bridge.

- Select **Map to Path** to map the portlet mode to a web resource in the portlet application, such as a page. The resource is not generated by the wizard. Enter the name and path in the corresponding field. You can create this resource after completing the wizard if it does not already exist.

  With this selection, you must write the targeted resource or file yourself. The target could be, for example, a JSP, a servlet, or an HTML file. This selection enters code in the generated portlet java class that routes requests for the given mode to the specified target.

- Select **Custom Code** to implement the portlet mode though a custom coded object. You must create this object later. This selection generates a skeleton method to render content (`private void do<MODE_NAME><CONTENT_TYPE>`) in the generated portlet java class. You must update this code to render useful content.

17. Click **Next.**

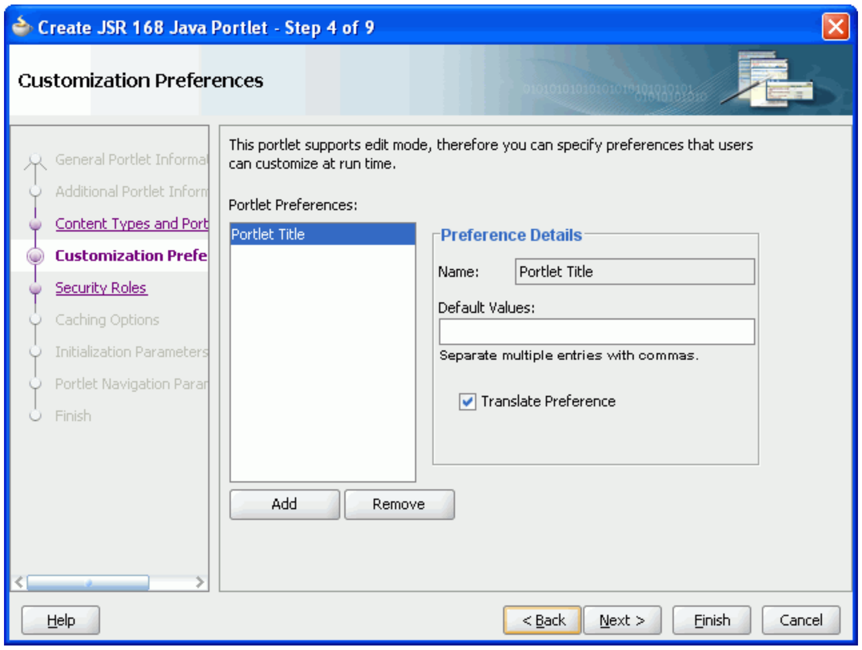
    If you selected **Enable users to edit portlet content** on the General Portlet Information page earlier in the wizard, then the Customization Preferences page appears where you can create additional customization preferences. Go to step 18.

    If you did not select this option, the Security Roles page appears. Go to step 23.

18. In the Customization Preferences page (Figure 29–6), click **Add** to add a new customization preference to the portlet.

    Customization preferences enable users of the portlet to specify values for the portlet at runtime. By default, the wizard includes a preference to enable users to customize the portlet title.

Figure 29–6 The Customization Preferences Page


Create JSR 168 Java Portlet - Step 4 of 9

Customization Preferences

This portlet supports edit mode, therefore you can specify preferences that users can customize at run time.

Portlet Preferences:

Portlet Title

Preference Details

Name: Portlet Title

Default Values:

Separate multiple entries with commas.

☑ Translate Preference

19. In the Add New Preference dialog, in the **Name** field, enter a name for the new preference.

    The name must be unique in the portlet. Use only letters, numbers, and the underscore character.

20. In the **Default Values** field, enter one or more default values for the new preference. Separate multiple values with commas.

21. To make the preferences translatable, select the **Translate** check box and then click **OK.**

JDeveloper generates a resource bundle class for translatable preferences, with strings for which you can obtain translations. At runtime, the portlet references the resource bundle entries.

Note: The **Name** is always translated, but there is not always a need to translate the **Default Values.** For example, if the value is an integer, then no translation is needed.

---

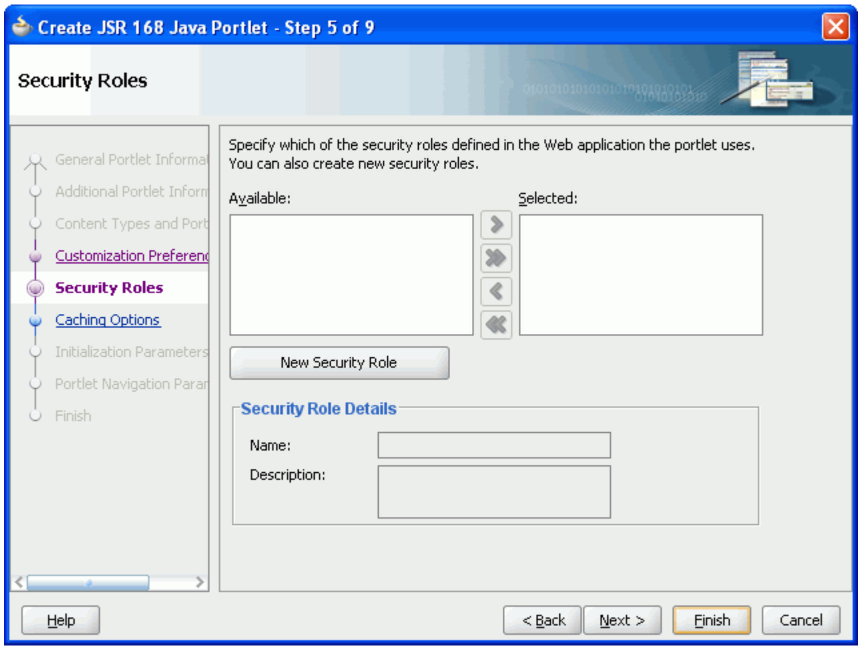
22. Repeat the preceding steps to add more preferences. When you are done click **Next.**

23. In the Security Roles page (Figure 29–7), to add an existing security role to your portlet, select the security role and move it to the **Selected** list.

    Security roles enable you to set tiered levels of access to the portlet. For example, a *View* user can view the portlet but cannot edit it; a *Customize* user can customize portlet settings; a *Manage* user can perform all available functions associated with the portlet.

    The **Available** list displays the security roles defined for the application in which you are creating the portlet. Moving a security role to the **Selected** list creates a reference of the security role in the application's portlet deployment file (`portlet.xml`) that refers to the security role in the application's web deployment file (`web.xml`).

Figure 29–7 The Security Roles Page



Specify which of the security roles defined in the Web application the portlet uses. You can also create new security roles.

24. To define a new security role, click **New Security Role.** In the Create New Security Role dialog, specify a name and description for the role and then click **OK** to add the role to the list of available roles.

    The name should be unique within the application. The description should provide details about the access privileges and restrictions this role has on the portlet.

You can also manually create security roles. For more information, see Section 24.3, "Setting Up Security for Your Application."

25. Click **Next.**

26. In the Caching Options page (Figure 29–8), to enable caching for your portlet, select **Cache Portlet**.

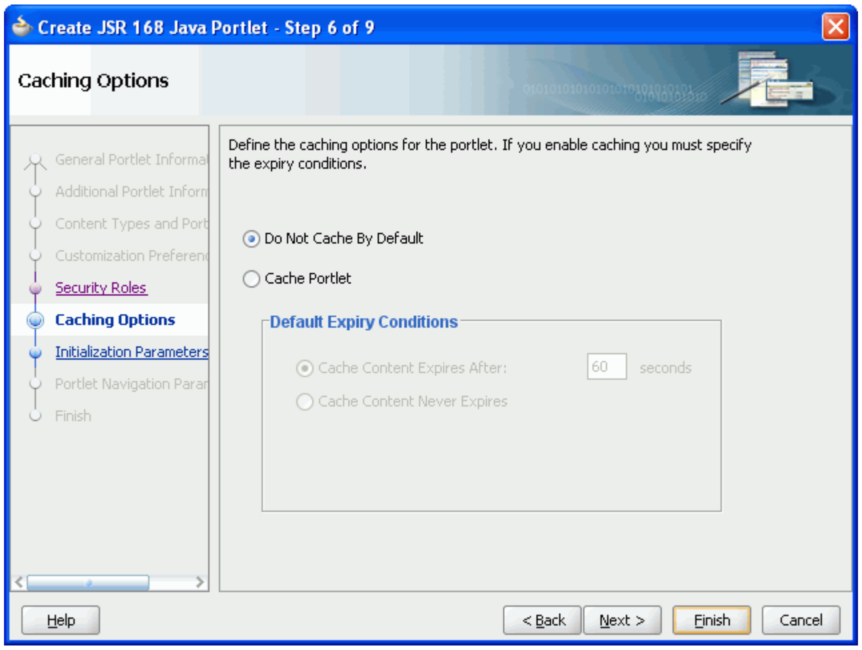
    Selecting this option indicates that portlet caching is managed by the portlet container. The portlet itself may choose to cache content for any given response. The settings on this page apply only when the portlet itself does not specify a caching condition for a response.

    If you do not want any default caching for this portlet, choose **Do Not Cache By Default.** In this case, the wizard actually sets a cache duration of 0 seconds. As stated earlier, 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 to implement default caching for the portlet, then you can change the cache duration value in the `portlet.xml` file, which is generated by the wizard, to a number greater than zero.

    For more information, se e Section 27.2.4, "Caching Style."

Figure 29–8 The Caching Options Page



Define the caching options for the portlet. 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

27. If you chose to cache the portlet, in the **Default Expiry Conditions** section, select:

    - **Cache Content Expires After** to expire the cached portlet content after a certain amount of time. Specify the time limit in the adjacent field.
    - **Cache Content Never Expires** to never expire the cached portlet content. You may want to select this option if the portlet contains static content that is unlikely to change.

28. Click **Next.**

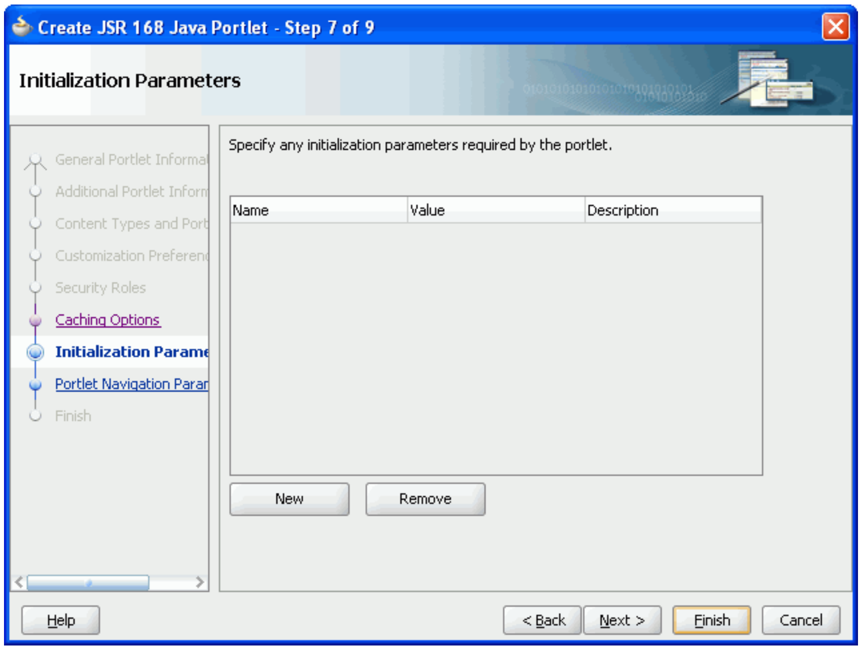
29. In the Initialization Parameters page (Figure 29–9), click **New** to add an initialization parameter to the portlet.

This adds a new row to the table of parameters. Double-click each field in the row to provide a name, default value and description for the parameter.

Repeat this step to add more initialization parameters. When you are done, click **Next.**

Initialization parameters provide the 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 application (for example, servlets and portlets) in a compatible way. These initialization parameters are added to the `portlet.xml` file.

Figure 29–9 The Initialization Parameters Page



Specify any initialization parameters required by the portlet.

Name	Value	Description

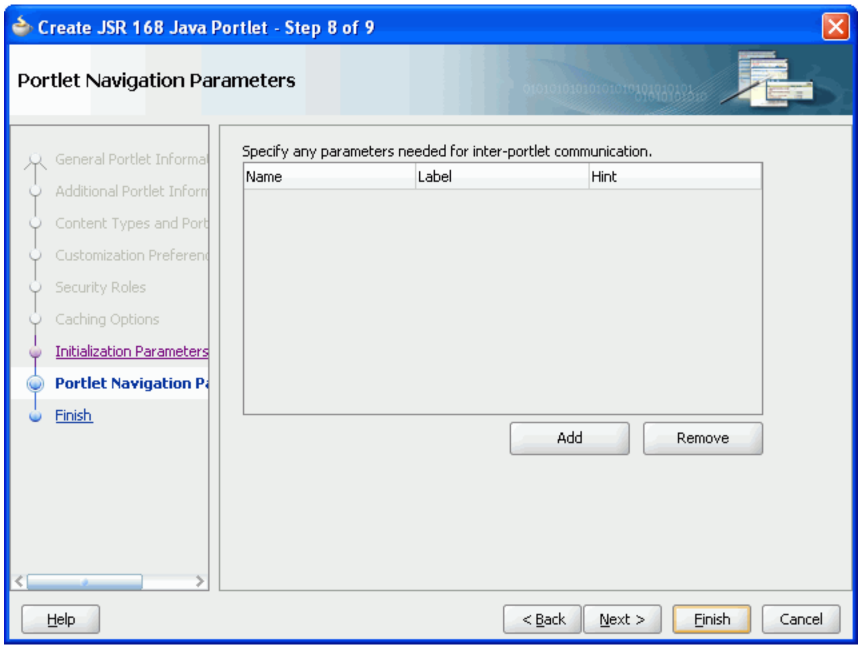
30. In the Portlet Navigation Parameters page (Figure 29–10), click **Add** to add a navigation parameter to the portlet.

    This adds a new row to the table of parameters. Double-click each field in the row to provide a name, label and hint for the parameter.

    Repeat this step to add more navigation parameters. When you are done, click **Finish.**

    Navigation parameters enable a portlet to communicate with the page on which it resides and with other portlets on that page. For more information, see Section 30.1.2, "How to Implement Navigational Parameters (WSRP 2.0)" and Section 9.7, "Contextually Linking WSRP 2.0 Portlets."

Figure 29–10 The Portlet Navigation Parameters Page



Specify any parameters needed for inter-portlet communication.

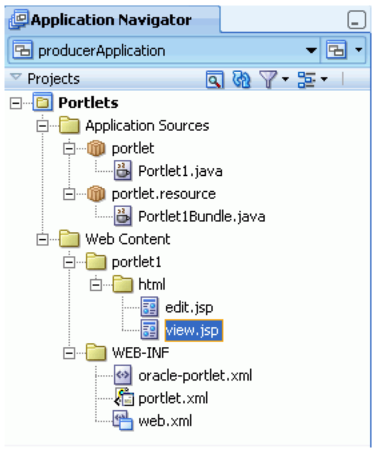
29.2.2 What Happens When You Create a JSR 168 Java Portlet Using the JDeveloper Wizard

When you use the JDeveloper wizard to create a JSR 168 Java portlet, JDeveloper generates a default implementation of the portlet. Specifically, the following files are created:

- Two Java classes:
  - `portletName.java` is invoked by the portlet container and contains all the methods required by the portlet standards.
  - `portletnameBundle.java` contains all the translation strings for the portlet.
- `oracle-portlet.xml` is generated if you selected the Oracle WSRP V2 extensions option.
- `portlet.xml` is the portlet deployment file for the application.
- `web.xml` is the web deployment file for the application.
- Files for each portlet mode you selected for the portlet:
  - If you selected **Generate JSP** for the portlet mode, a JSP page is created for the mode, for example, `view.jsp`.
  - If you selected **Generate ADF-Faces JSPX**, a JSPX page is created for the mode, for example, `view.jspx`. You can add Faces components to this page.
  - If you selected **Map to Path,** no additional files are created as the code for the portlet mode resides in an existing resource. Code is added to the portlet's Java class to route requests to the specified target.
  - If you selected **Custom Code**, no additional files ar e created, but the code for the portlet mode resides in the portlet's Java class.

You can see all these files in the Application Navigation, as shown in Figure 29–11.

Figure 29–11 Files Generated for a JSR 168 Java Portlet



The next step is to extend the sample code with your own business logic to implement the desired functionality and features for your portlet. For more information, see the JSR 168. For a specific example, see "Step 3: Create the Business Logic for the Standards-Based Portlet" in the *Oracle Fusion Middleware Tutorial for Oracle WebCenter Developers*.

29.2.3 How to Create a PDK-Java Portlet

Using the Create Oracle PDK-Java Portlet wizard in JDeveloper you can quickly and easily create PDK-Java portlets. You can choose which portlet modes you want to implement and the implementation method (JSP, HTTP servlet, Java class, or HTML) to use for each mode. The wizard then creates a simple implementation for each of the selected modes.

To create a PDK-Java portlet using the JDeveloper wizard:

1. Start JDeveloper.
2. In the Application Navigator, open the application under which you want to create your portlet.

   The application should be scoped for portlet creation. An easy way to achieve this is to use a Portlet Producer Application. WebCenter Applications are not scoped for portlet creation.
3. Right-click the project under which you want to create your portlet, and choose **New.**

   Note: To create the portlet in an existing producer, right-click the producer's `provider.xml` file and choose **Add Portlet.** This takes you directly to the General Portlet Information page of the Create Oracle PDK-Java Portlet wizard (step 9).

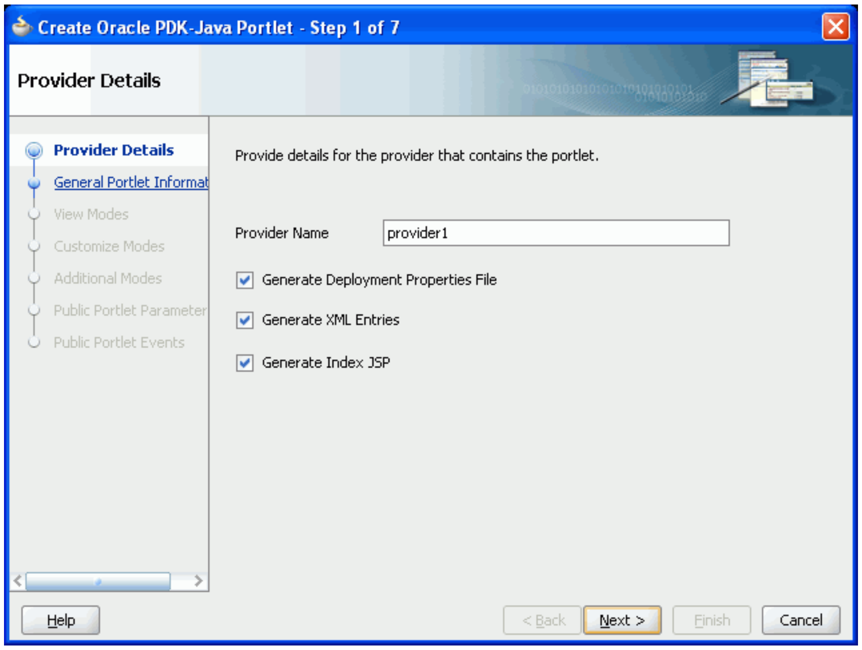
4. In the New Gallery, expand **Web Tier,** select **Portlets** and then **Oracle PDK-Java Portlet,** and click **OK.**

   Note: Selecting **Oracle PDK-Java Portlet** opens the wizard for creating PDK-Java portlets. Selecting **Standards-based Java Portlet** opens the wizard for creating JSR 168-compliant portlets.

5. In the Provider Details page of the Create Oracle PDK-Java Portlet wizard (Figure 29–12), enter a name for the new producer to contain your portlet. This name must be unique within the project.

   In the PDK-Java, the term provider is used instead of producer. A provider is the same thing as a producer.

Figure 29–12 The Provider Details Page



Provide details for the provider that contains the portlet.

Provider Name: provider1

☑ Generate Deployment Properties File

☑ Generate XML Entries

☑ Generate Index JSP

6. Select **Generate Deployment Properties File.**

   This automatically generates two `.properties` files:

   - *`serviceID`*`.properties` defines properties for a producer with that service ID. The service ID has the same value as the producer name.
   - `_default.properties` is a default properties file. A producer application may have multiple producers, each with its own service ID. On registration, if no service ID is defined, then the default properties file is used.

7. Select **Generate XML Entries.**

   This automatically generates a producer definition file (`provider.xml`) for the producer that contains details of the portlets belonging to the producer, including those generated by the wizard.

8. Select **Generate Index JSP** and then click **Next.**

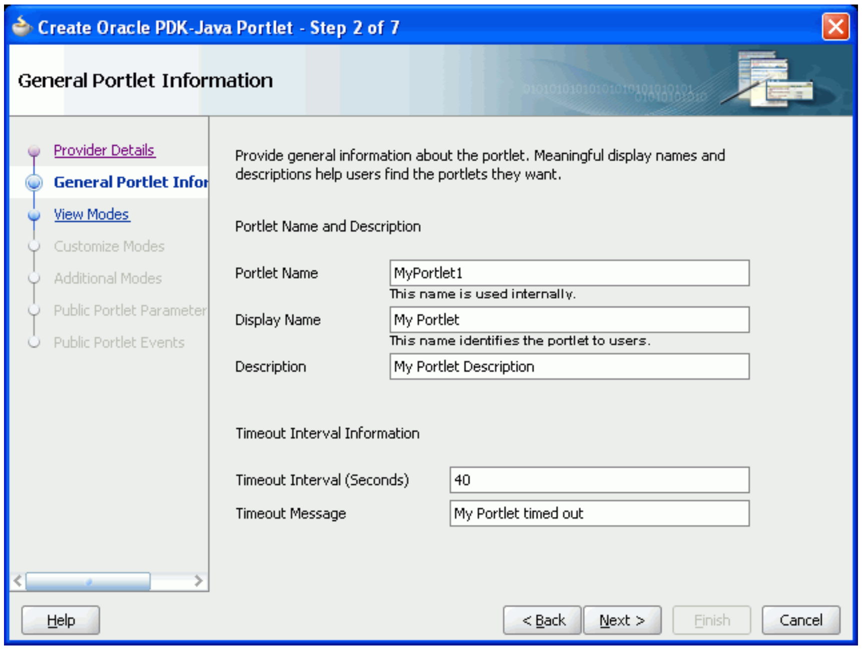
   This automatically generates an `index.jsp` file that lists all the producers that reside in the application with hyperlinks that enable easy access to producer test pages.

9. In the General Portlet Information page (Figure 29–13), enter a name and display name for your portlet.

   The name is used internally and is not exposed to users. The display name is displayed to users in portlet selection lists, such as the Component Palette.

The description is not implemented in WebCenter applications so you do not need to enter a value in this field unless your portlet is likely to be consumed by other applications, such as Oracle Portal.

Figure 29–13 The General Portlet Information Page



Provide general information about the portlet. Meaningful display names and descriptions help users find the portlets they want.

Portlet Name and Description

Portlet Name: MyPortlet1
This name is used internally.

Display Name: My Portlet
This name identifies the portlet to users.

Description: My Portlet Description

Timeout Interval Information

Timeout Interval (Seconds): 40

Timeout Message: My Portlet timed out

10. In the **Timeout Interval (Seconds)** field, enter the number of seconds to allow for rendering the portlet.
11. In the **Timeout Message** field, enter a message to display if the rendering of the portlet exceeds the timeout interval specified and then click **Next.**
12. In the View Modes page (Figure 29–14), under Show Page, from the **Implementation Style** list, select the implementation style to use for the portlet's Shared Screen mode.

    - Select **JSP** to implement the portlet's Shared Screen mode as a JavaServer Page. In the **File Name** field, enter the name of the file to be generated by the wizard.
    - Select **HTTP Servlet** to implement the portlet's Shared Screen mode as an HTTP servlet. In the **Package Name** field, enter the name of the package that contains the HTTP servlet. In the **Class Name** field, enter the Java class to be referenced with the portlet's Shared Screen mode.
    - Select **HTML File** to implement the portlet's Shared Screen mode as an HTML file. In the **File Name** field, enter the name of the file to be generated by the wizard. Note that, when you choose **HTML File,** the following code is added inside the `<renderer>` element of your `provider.xml` file:

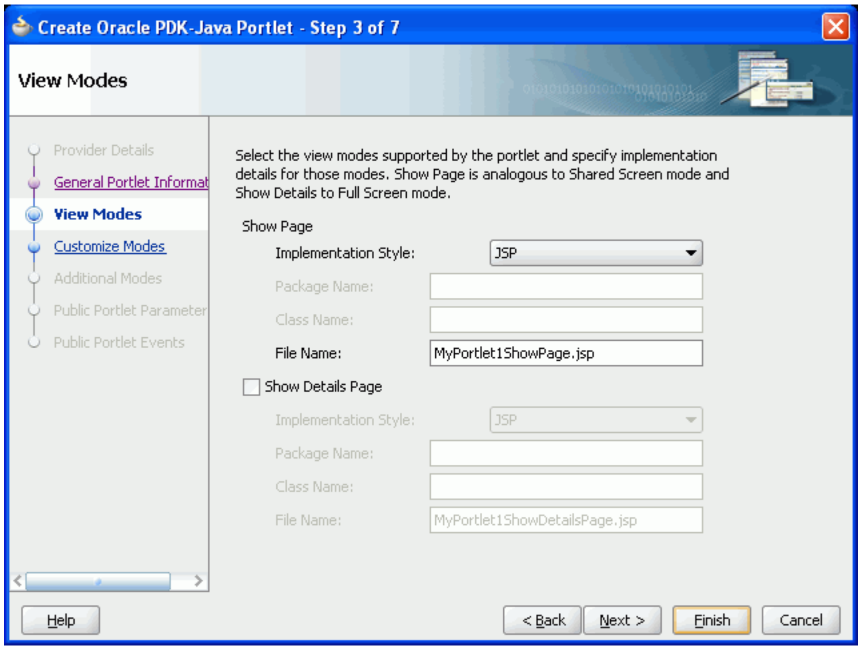
```
<showPage class="oracle.portal.provider.v2.render.http.ResourceRenderer">
    <resourcePath>/hub_inside/index.html</resourcePath>
    <contentType>text/html</contentType>
    <charSet>UTF-8</charSet>
 </showPage>
```

      `<charSet>` tells the producer what character set to use to encode the HTML page. The default character set specified by the wizard is UTF-8. If you require a different character set, you must update this element of `provider.xml` accordingly.

- Select **Java Class** to implement the portlet's Shared Screen mode as a Java class. In the **Package Name** field, enter the name of the package that contains the Java class. In the **Class Name** field, enter the name of the Java class.

For more information about Shared Screen mode, see Section 29.2.5.1.1, "Shared Screen Mode (View Mode for JSR 168)."

Figure 29–14 The View Modes Page



Select the view modes supported by the portlet and specify implementation details for those modes. Show Page is analogous to Shared Screen mode and Show Details to Full Screen mode.

Show Page
Implementation Style: JSP
Package Name:
Class Name:
File Name: MyPortlet1ShowPage.jsp

☐ Show Details Page
Implementation Style: JSP
Package Name:
Class Name:
File Name: MyPortlet1ShowDetailsPage.jsp

13. To implement Full Screen mode for your portlet, select **Show Details Page** and then select an implementation style as described for Full Screen mode in step 12.

    For more information about Full Screen mode, see Section 29.2.5.1.5, "Full Screen Mode (PDK-Java)."

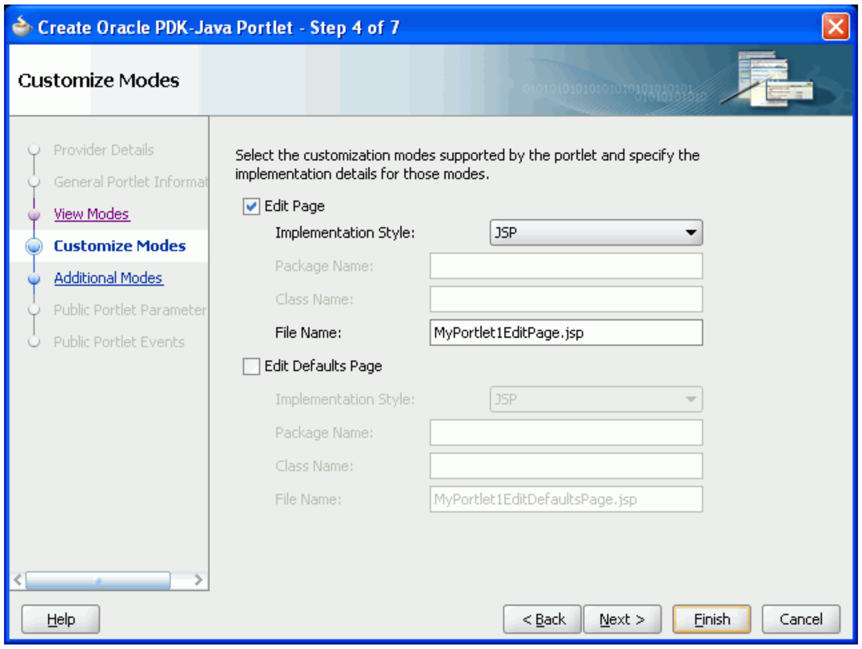
14. At this point in the wizard, you can click **Finish** to create the portlet immediately, using the default values for all remaining settings.

    To provide additional details for your portlet, click **Next** and follow the remaining steps.

15. In the Customize Modes page (Figure 29–15), **Edit Page** is selected by default. To implement Edit mode for your portlet, select an implementation style as described for Shared Screen mode in step 12. If you do not want to implement Edit mode, then deselect **Edit Page.**

    For more information about Edit mode, see Section 29.2.5.1.2, "Edit Mode (JSR 168 and PDK-Java)."

Figure 29–15 The Customize Modes Page


Create Oracle PDK-Java Portlet - Step 4 of 7

Customize Modes

Select the customization modes supported by the portlet and specify the implementation details for those modes.

[✓] Edit Page

Implementation Style: JSP

Package Name:

Class Name:

File Name: MyPortlet1EditPage.jsp

[ ] Edit Defaults Page

Implementation Style: JSP

Package Name:

Class Name:

File Name: MyPortlet1EditDefaultsPage.jsp

16. To implement Edit Defaults mode for your portlet, select **Edit Defaults Page**, and then select an implementation style as described for Shared Screen mode in step 12.

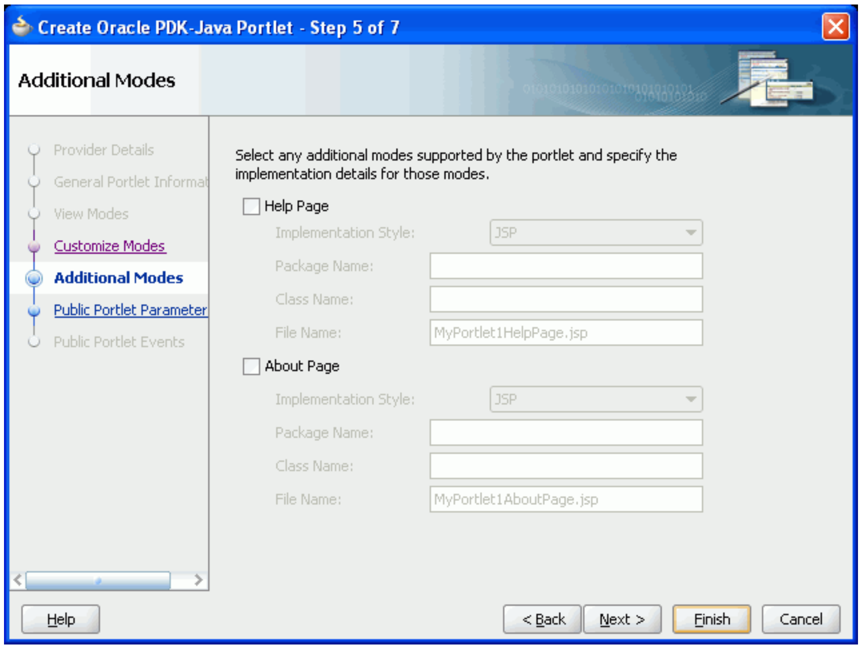
    For more information about Edit Defaults mode, see Section 29.2.5.1.3, "Edit Defaults Mode (JSR 168 and PDK-Java)."

17. Click **Next.**

18. In the Additional Modes page (Figure 29–16), to implement Help mode for your portlet, select **Help Page**, and then select an implementation style as described for Shared Screen mode in step 12.

    For more information about Help mode, see Section 29.2.5.1.6, "Help Mode (JSR 168 and PDK-Java)."

Figure 29–16 The Additional Modes Page



Select any additional modes supported by the portlet and specify the implementation details for those modes.

☐ Help Page

Implementation Style:	JSP
Package Name:	
Class Name:	
File Name:	MyPortlet1HelpPage.jsp

☐ About Page

Implementation Style:	JSP
Package Name:	
Class Name:	
File Name:	MyPortlet1AboutPage.jsp

19. To implement About mode for your portlet, select **About Page**, and then select an implementation style as described for Shared Screen mode in step 12.

For more information about About mode, see Section 29.2.5.1.7, "About Mode (JSR 168 and PDK-Java)."

20. Click **Next.**

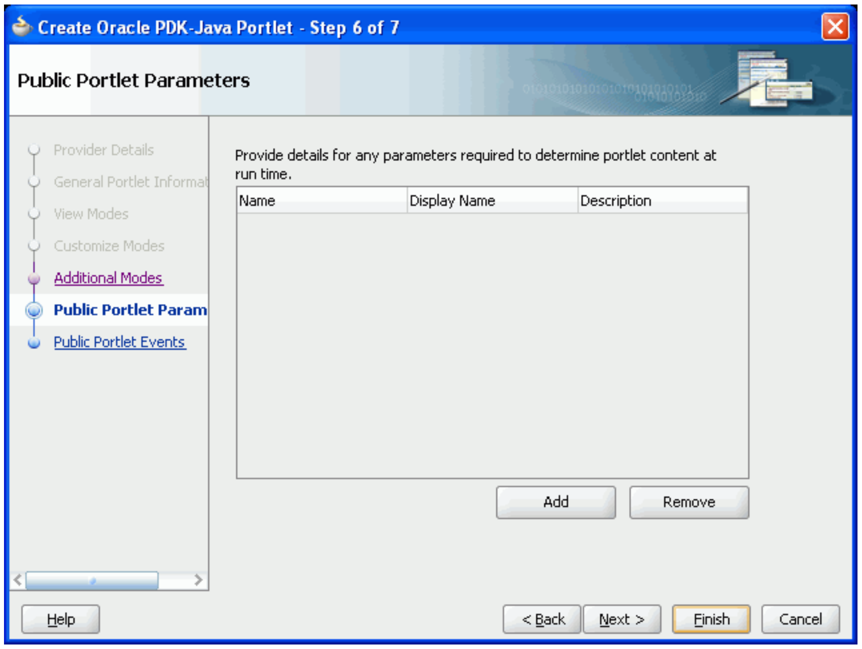
21. In the Public Portlet Parameters page (Figure 29–17), click **Add** to add a public parameter to your portlet.

This adds a new row to the table of parameters. Double-click each field in the row to provide a name, display name, and description for the parameter.

Repeat this step to add more public parameters. When you are done, click **Next.**

Public portlet parameters enable a portlet to communicate with the page on which it resides and with other portlets on that page. For more information, see Section 30.2.2, "How to Implement Public Parameters."

Figure 29–17 The Public Portlet Parameters Page


Create Oracle PDK-Java Portlet - Step 6 of 7

Public Portlet Parameters

Provide details for any parameters required to determine portlet content at run time.

Name	Display Name	Description

22. Oracle PDK-Java events are not supported in WebCenter applications, so in the Public Portlets Events page, click **Finish.**

    For more information about events, see the *Oracle Fusion Middleware Developer's Guide for Oracle Portal*.

29.2.4 What Happens When You Create a PDK-Java Portlet

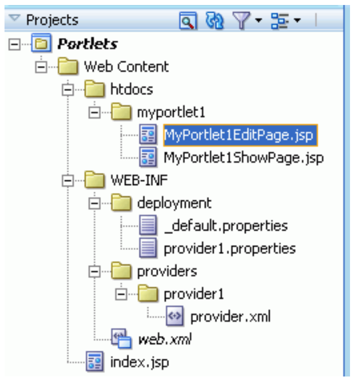
When you use the JDeveloper wizard to create a PDK-Java portlet, JDeveloper generates a default implementation of the portlet. Specifically, the following files are created:

- Files for each portlet mode you selected, for example `*portletname*ShowPage.jsp`.
- `provider.xml` is the producer definition file that contains details of the portlets belonging to the producer.
- `web.xml` is the web deployment file for the application.
- `index.jsp` is used by JDeveloper for testing purposes
- `_default.properties` is the default properties file.
- `*serviceID*.properties` is the properties file for the producer identified by *serviceID*.

All these files are required to deploy and run the portlet successfully, except for `index.jsp`.

You can see all these files in the Application Navigator, as shown in Figure 29–18.

Figure 29–18 Files Generated for a PDK-Java Portlet



The next step is to extend the sample code with your own business logic to implement the desired functionality and features for your portlet. For more information, see the PDK-Java JavaDoc.

29.2.5 What You May Need to Know When Creating Java Portlets

When you write your portlets in Java for either JSR 168 or PDK-Java, you should follow the best practices described in this section, which are as follows:

- Section 29.2.5.1, "Guidelines for Portlet Modes"
- Section 29.2.5.2, "Guidelines for Navigation within a Portlet"
- Section 29.2.5.3, "Guidelines for JavaScript"
- Section 29.2.5.4, "Guidelines for PDK-Java Portlets"

29.2.5.1 Guidelines for Portlet Modes

Portlet mode exhibits the runtime portlet functionality seen by users. You can provide an extended set of modes in addition to the standard ones provided with Oracle WebCenter and Oracle Portal. Different portal products support extended portlet modes as follows:

- Oracle WebCenter and Portal have defined an extended set of modes that these two products support. Different modes are available for both PDK-Java and JSR 168 portlets. PDK-Java offers some modes not offered by JSR 168. If you are coding portlets to JSR 168, then you can declare custom portlet modes in the `portlet.xml` file that map to the extra modes offered by PDK-Java, or to accommodate any other functionality you may want to provide. For example, the JSR 168 Java Portlet wizard for JSR 168 portlets includes a custom mode called print, which you can use to provide a printer friendly version of the portlet.

  Defining custom modes is especially useful if the portlet must interoperate with portal implementations from other vendors. You can offer any of these modes to users. In fact, it is recommended that some modes like Edit Defaults are offered.

- Third party portal products may have their own set of extended modes (JSR 168 custom modes) that producers can offer. In WebCenter applications and Oracle Portal, chrome UI for portlets only shows the custom modes that are defined in Oracle WebCenter Framework. Arbitrary custom modes that a third party or custom portlet producer offers are ignored and therefore not supported.

A portlet may have the following portlet modes, each with its own visualization and behavior:

- Shared Screen Mode (View Mode for JSR 168)
- Edit Mode (JSR 168 and PDK-Java)
- Edit Defaults Mode (JSR 168 and PDK-Java)
- Preview Mode (JSR 168 and PDK-Java)
- Full Screen Mode (PDK-Java)
- Help Mode (JSR 168 and PDK-Java)
- About Mode (JSR 168 and PDK-Java)

29.2.5.1.1 Shared Screen Mode (View Mode for JSR 168) A portlet uses Shared Screen mode (known as View mode in JSR 168) to appear on a page with other portlets. This is the mode most people think about when they envision a portlet. A JSR 168 portlet must have a view mode, the rest are optional.

When developing portlets, you must consider all of the factors that may influence the portlet's appearance on the page, such as the portlet's containing object and the other portlets with which your portlet shares the page. For example, suppose you choose to place your portlet inside of an HTML table cell. This means the portlet should display only content that can be rendered within a table cell. Furthermore, the actual size of the table cell may vary depending on user settings, the browser width, and the amount and style of content in the portlet.

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, if 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 and HTML Tidy can help detect and fix hanging and unnecessary tags.
- **Consider restrictions imposed by container objects.** If your portlet is contained inside of an HTML element, such as a table cell, then you must ensure that your portlet can be rendered within that container. For example, if you place a portlet in a table cell, then you could not use frames in the portlet because they 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 may end up with portlet content too small or cramped for smaller monitors. Full screen content is best viewed in Full Screen mode of PDK-Java.
- **Do not create fixed-width HTML tables in portlets.** You have no way to tell how wide a column your portlet will have on a user's page. If your portlet requires more room than given, then 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 want and they can change it at any time. Your portlet should handle these situations gracefully.

The HTML you use also affects 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 consider the following:

- **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, then it can greatly slow the rendering of the page.

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 page. The portlets access these settings for their fonts and colors, either directly or using the Application Programming Interface (API).

While different browsers have implemented varying levels of the full CSS specification, Oracle WebCenter Framework 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, then 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 style sheet definition is available at the top of 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 style sheet changes in the future.
- **Avoid using CSS for absolute positioning.** Since users can personalize their pages, you cannot guarantee that your portlet can appear in a particular spot.
- **Follow Accessibility Standards.** You should ensure that you code your style sheets according to existing accessibility standards (more information available at `http://www.w3.org/TR/WCAG10-CSS-TECHS/`).

29.2.5.1.2 Edit Mode (JSR 168 and PDK-Java) A portlet uses Edit mode to enable users to personalize the behavior of the portlet. Edit mode provides a list of settings that the user can change. These 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.

Users typically access a portlet's Edit mode by choosing **Personalize** from the portlet's dropdown list of options. When users choose **Personalize**, a new page appears in the same browser window. The portlet typically creates a Web page representing a dialog to choose the portlet's settings. After applying the settings, users automatically return to the original page.

Guidelines for Edit Mode Operations

The following guidelines should govern what you expose to users in Edit mode:

- **Enable users to personalize the title of the portlet.** The same portlet may be added to the same page several times. Enabling the user to personalize the title helps alleviate confusion.
- **If using caching, invalidate the content.** If personalizations cause a change in portlet display or content, then 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 producer settings or do other administrative tasks, then you should create secured portlets specifically for those tasks.
- **Only show mobile options when applicable.** The portlet can interrogate whether it should display the Edit mode page for mobile or desktop. Furthermore, if the page might serve both mobile and desktop users, you should consider delineating between mobile options and desktop options. Refer to Section 15.1.4.5, "Tailor Personalization Pages" for additional tips.

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 personalizations and returns the portlet to view mode.
- **Apply** saves the user personalizations and reloads the current page.
- **Cancel** returns the portlet to view mode without saving changes.

Guidelines for Rendering Personalization Values

When you show the forms used to change personalization settings, you should default the values such that the user does not have to constantly reenter settings. When rendering the personalization values, use the following sequence to provide consistent behavior:

1. **User preference:** Query and display this user's personalizations, if available.
2. **Instance defaults:** If no user personalizations are found, then query and display system defaults for the portlet instance. These are set in Edit Defaults mode and apply only to this portlet instance.
3. **Portlet defaults:** If no system default personalizations are found, then 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 enables the personalizations to be presented in a predictable way, consistent with the other portlets in the WebCenter portlet application.

29.2.5.1.3 Edit Defaults Mode (JSR 168 and PDK-Java) A portlet uses the Edit Defaults mode to enable administrators to customize the default behavior of a particular portlet instance. Edit Defaults mode provides a list of settings that the application developer can change. These 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 personalization 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.

Administrators access Edit Defaults mode, when editing a page, by choosing **Customize** from the portlet's dropdown list.

When users click the Customize icon, the portlet displays in the same browser window. The portlet typically creates a Web page representing a dialog to personalize the portlet instance settings. After applying the settings, users are automatically returned to the original page.

Guideline for Edit Defaults Mode Options

The following guideline should govern what you expose to page designers in Edit Defaults mode:

- **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 producer settings or do other administrative tasks, then you should create secured portlets specifically for those tasks.
- **Only show mobile options when applicable.** The portlet can interrogate whether it should display the Edit mode page for mobile or desktop. Furthermore, if the page might serve both mobile and desktop users, you should consider delineating between mobile options and desktop options. Refer to Section 15.1.4.5, "Tailor Personalization Pages" for additional tips.

Guidelines for Buttons in Edit Defaults Mode

For consistency and user convenience, Edit Defaults mode should implement the following buttons in the following order:

- **OK** saves the user personalizations and returns the portlet to view mode.
- **Apply** saves the user personalizations and reloads the current page.
- **Cancel** returns the portlet to view mode without saving changes.

Guidelines for Rendering Personalization Values

When you show the forms used to change personalization settings, you should default the values so that the application developer does not have to constantly reenter settings. When rendering personalization values, use the following sequence to provide consistent behavior:

1. **Instance preferences:** Query and display system defaults for the portlet instance.
2. **Portlet defaults**: If no system default personalizations are found, then 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 enables the personalizations to be presented in a predictable way, consistent with the other portlets in the custom WebCenter application.

29.2.5.1.4 Preview Mode (JSR 168 and PDK-Java) A portlet uses Preview mode to show the user how the portlet looks before adding it to a page. Preview mode visually represents what the portlet can do. Not all portlet consumers call this mode. For example, Oracle Portal uses this mode but Oracle WebCenter Framework does not.

Note: Oracle Portal calls this mode when the user clicks the Preview icon from the Add Portlet page. A window then displays the preview of the chosen portlet. The user has the option to add that portlet to the page. This mode has no particular application in WebCenter portlet application, but used in Oracle Portal's Portlet Repository.

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 work as expected when rendered in Preview mode. Hyperlinks can be simulated using the underline font.
- **Do not use active form buttons.** Forms may not work as you expect them to when rendered in Preview mode. If you decide to render form elements, then do not link them to anything.

29.2.5.1.5 Full Screen Mode (PDK-Java) Portlets use Full Screen mode to provide a larger version of the portlet for displaying additional details. Full Screen mode lets a portlet have the entire window to itself. Not all portlet consumers call this mode. For example, Oracle Portal uses this mode but Oracle WebCenter Framework does not. In Oracle Portal, users access a portlet's Full Screen mode by clicking the title of the portlet.

For example, if a portlet displays expense information, then 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, then it could link to the entire expense application from Full Screen mode.

Technically, JSR 168 portlets do not have Full Screen mode. However, you can implement the equivalent of Full Screen mode for a JSR 168 portlet with View mode (Shared Screen mode) and a maximized state for the window.

29.2.5.1.6 Help Mode (JSR 168 and PDK-Java) 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.

Users access a portlet's Help mode by choosing the Help action in the portlet.

Guideline for Help Mode

The following guideline 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.

29.2.5.1.7 About Mode (JSR 168 and PDK-Java) 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.

Users access a portlet's About mode by choosing **About** from the dropdown list in the portlet's chrome. 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.

Guideline for About Mode

The following guideline 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.

29.2.5.2 Guidelines for Navigation within a Portlet

In some ways, navigation between different sections or pages of a single portlet is identical to navigation between standard Web pages. Users can submit forms and click links. In typical, simple Web pages, both of these actions involve sending a message directly to the server responsible for rendering the new content, which is then returned to the client. In portlets, which comprise only part of a page, the form submission or link rendered within the portlet does not directly target the portlet. It passes information to the portlet through the custom WebCenter application. If a link or form within a portlet does not refer back to the application, then following that link takes the user away from the application, which is not typically the desired behavior.

The component developer does not need to know the detailed mechanics of how the parameters of a form or link get passed around between the user, application, and portlet. However, they must understand that they cannot write links in a portlet the same way they do for typical, simple Web pages.

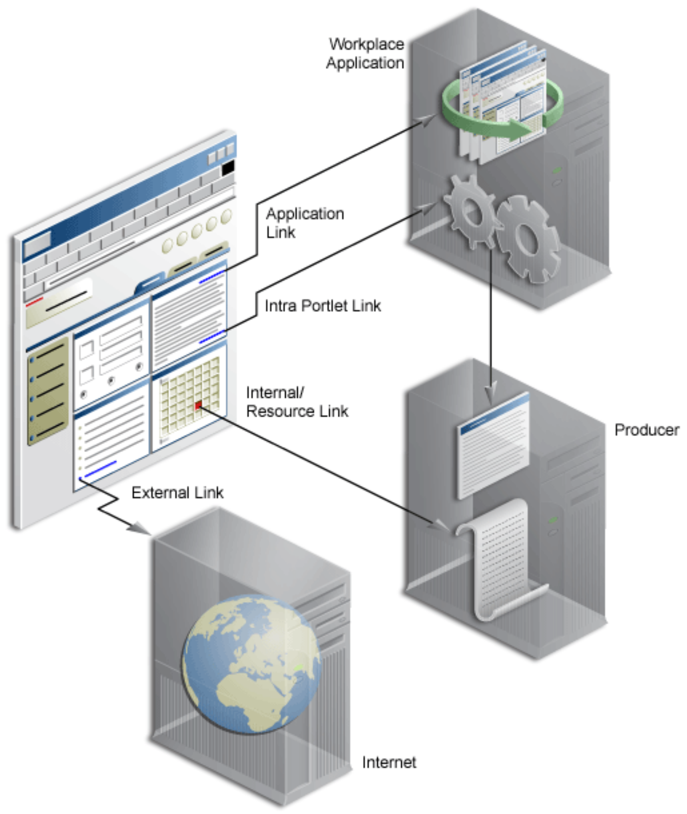
Types of Links for Portlets

A portlet may render links of four classes, as follows:

- **Intraportlet links** require the portlet to be aware of the address of the custom WebCenter application because they actually refer to it in some way.
- **Application links**, like intraportlet links, must be aware of the address of the custom WebCenter application for the same reason.
- **External links** make no reference to the custom WebCenter application and work in portlets as they would do in a normal Web page.
- **Internal/Resource links**, like external links, also make no reference to the custom WebCenter application.

Figure 29–19 contains a summary of these link types. The arrows indicate how the links reference the resources to which they logically refer.

Figure 29–19 Custom WebCenter Application Link Types



29.2.5.2.1 Intraportlet Links Intraportlet links go to different sections or pages within a given portlet. Strictly speaking, they refer to the page containing the portlet, but they contain parameters that cause the portlet to render a different section or page within that page when it is requested by the user.

As a direct consequence, a portlet cannot expect to render links to different sections or pages of itself using relative links or absolute links based on its own server context. Intraportlet link are useful for intraportlet navigation, either as links or form submission targets.

29.2.5.2.2 Application Links Application links refer to significant pages within the custom WebCenter application, such as the user's home page.

29.2.5.2.3 External Links External links refer neither to the portlet (through a page) nor to any part of the custom WebCenter application. If selected, these links take the user away from the application, for example, www.oracle.com.

29.2.5.2.4 Internal/Resource Links Internal/Resource links refer to internal (to the portlet) resources. Sometimes they are exclusively used internally during portlet rendering, for example as a server side include. On other occasions, they may be used externally to reference portlet resources like images. In this latter case, you can use the PDK-Java `constructResourceURL` method in the `UrlUtils` class to retrieve images from behind a firewall using resource proxy. Note that in order for resource proxying to work, you must first set the JNDI variable, `oracle/portal/provider/sample/resourceUrlKey`, for the producer. For more information about setting JNDI variables, see Section 30.2.4.2, "Setting JNDI Variable Values."

For example, `lottery.jsp` of the lottery sample, which is available with PDK-Java, contains resource proxy requests for images.

```
<%@ page contentType="text/html;charset=UTF-8" %>
<%@ page session="false" import="oracle.portal.provider.v2.render.*" %>
<%@ page import="oracle.portal.provider.v2.render.http.HttpPortletRendererUtil" %>
<%@ page import="oracle.portal.provider.v2.url.UrlUtils" %>
<%@ page import="oracle.portal.sample.v2.devguide.lottery.*" %>
<%
  LottoPicker picker = new LottoPicker();
  picker.setIdentity(request.getRemoteAddr() ); %>
<% PortletRenderRequest portletRequest = (PortletRenderRequest)
request.getAttribute("oracle.portal.PortletRenderRequest"); %>
<% String name = portletRequest.getUser().getName(); %>
<P class="PortletHeading1" ALIGN="CENTER">Hi <%= name %>, Your Specially
  Picked</P>
<P ALIGN="CENTER"><IMG SRC="<%= UrlUtils.constructResourceURL(portletRequest,
  HttpPortletRendererUtil.absoluteLink(request, "images/winningnumbers.gif")) %>"
  WIDTH="450" HEIGHT="69" ALIGN="BOTTOM" BORDER="0"></P>
<P>
<P ALIGN="CENTER">
<TABLE ALIGN="CENTER" BORDER="0" CELLPADDING="0" CELLSPACING="0">
<TR>
<%
     int [] picks = picker.getPicks();
     for (int i = 0; i < picks.length; i++) {
%>
             <TD>
    <IMG SRC="<%= UrlUtils.constructResourceURL(portletRequest,
      HttpPortletRendererUtil.absoluteLink(request, "images/ball" + picks[i])) %>
      .gif" WIDTH="68" HEIGHT="76" ALIGN="BOTTOM" BORDER="0">
     </TD>
<%
     }
```

For session-based producers, any cookies returned from the original `initSession` call to the producer are sent with the request back to the producer to maintain the right session context.

29.2.5.3 Guidelines for JavaScript

JavaScript can often be useful within a portlet, but bear in mind the following guidelines within your portlets:

- You should never use JavaScript to redirect the page in which the portlet is rendered. If you need to direct users elsewhere, then you should do so in your portlet action handling code or open a new window in the browser.
- Ensure that identifiers in your JavaScript are qualified.By qualifying your identifiers, you ensure that they are unique and do not clash with any JavaScript on the page.

29.2.5.4 Guidelines for PDK-Java Portlets

In Oracle WebCenter Framework, PDK-Java portlets work somewhat differently than they did in Oracle Portal. As a result, you must be aware of the following new design considerations when you build PDK-Java portlets in Oracle WebCenter Framework:

- Your portlet must not contain any code that relies upon the URL format or parameters in the request that were not explicitly added by your portlet

- You should never assume that your portlet is the only one on a page, regardless of the portlet mode. For example, even if your are in Edit mode, you should not assume that yours is the only portlet on the page.
- Never write a portlet mode that simply redirects. A redirect can only be issued while processing a post to your portlet or following a link generated by your portlet.

30 Coding Portlets

This chapter explains how you can enhance the Java portlets you created with the Oracle JDeveloper Portlet Wizard.

This chapter includes the following sections:

Before You Begin

Before you begin looking through this chapter ensure that:

- You are familiar with portlet terminology such as portlet modes. For more information, see Chapter 27, "Overview of Portlets" and Section 29.2.5, "What You May Need to Know When Creating Java Portlets.".
- You have access to WebLogic Server (WLS) managed servlet that is configured for use as a portlet container, such as the WLS_Portlet server.
- You are familiar with JDeveloper and know how to build and deploy Java components using it. You can download JDeveloper from OTN:

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

30.1 Enhancing JSR 168 Java Portlets

When you have built your initial portlet in the Portlet Wizard as described in Section 29.2.1, "How to Create a JSR 168 Java Portlet," you will want to enhance it. Because JSR 168 portlets adhere to the Java standards, you can find substantial information about enhancing them from many different sources, such as third-party books and Web pages. Some of the more important enhancements that you might want to perform are as follows:

- Adding personalization. For more information, see Section 30.1.1, "How to Add Personalization."
- Adding navigational parameters. For more information, see Section 30.1.2, "How to Implement Navigational Parameters (WSRP 2.0)."
- Exporting and importing of customizations. For more information, see Section 30.1.3, "How to Implement Export/Import of Customizations (WSRP 2.0)."

- Rewriting URLs for a resource proxy. For more information, see Section 30.1.4, "How to Implement Rewritten URLs for Resource Proxy."
- Implementing stateless resource proxying. For more information, see Section 30.1.5, "How to Implement Stateless Resource Proxying."
- Disabling caching. For more information, see Section 30.1.6, "How to Disable Java Object Cache for Preference Store Access."
- Adding security. For more information, see Section 30.1.7, "How to Implement Security for JSR 168 Portlets."

30.1.1 How to Add Personalization

As a quick example of adding personalization, you can enhance the portlet you created in Section 29.2.1, "How to Create a JSR 168 Java Portlet" with some code that enables a user in Edit or Edit Defaults mode to paste HTML into a field for the portlet to render. You can then easily redeploy the portlet and then test it.

Before You Begin

The steps that follow assume that you have:

- Built a portlet using the Create JSR 168 Java Portlet wizard in JDeveloper. For more information, see Section 29.2.1, "How to Create a JSR 168 Java Portlet."
- Added a preference called `portletContent` in the Customization Preferences page of the wizard.
- Registered the producer with a custom WebCenter application and added the portlet to a page. For more information, see Section 9.2.1, "How to Register a WSRP Portlet Producer" and Section 9.3.1, "How to Add a Portlet to a Page."
- Enabled Oracle ADF security for the application. For more information, see Section 24.3, "Setting Up Security for Your Application."

  The user must be logged in to the application as an authenticated user to access the personalization feature.

To implement simple personalization:

1. Start JDeveloper.
2. In the Application Navigator, open the application that contains the portlet.
3. Expand the project that contains the portlet.
4. Right-click `view.jsp` and choose **Open.**
5. In the visual editor, click the **Source** tab and add the code indicated in bold in Example 30–1 to display the `portletContent` customization preference:

Example 30–1 view.jsp Sample Code

```
<%@ page contentType="text/html"
    import="javax.portlet.*,java.util.*,Portlets.Portlet1,
    Portlets.resource.Portlet1Bundle"%>
<%@ taglib uri="http://java.sun.com/portlet" prefix="portlet"%>

<portlet:defineObjects/>
<%
String[] str = {"Portlet Content"};
PortletPreferences prefs = renderRequest.getPreferences();
str = prefs.getValues("portletContent",str);
```

```
for (int i=0; i<str.length; i++)
{
%><%=(i<str.length-1)?str[i]+", ":str[i]%><%}%>
```

6. In the Application Navigator, right-click `edit.jsp` and choose **Open.**

   Notice that the JSP consists of a form field, a form input field, and two form button fields.

7. In the visual editor, click the **Source** tab and add the code indicated in bold in Example 30–2 to implement a form field for users to enter a value for the `portletContent` customization preference:

Example 30–2 edit.jsp Sample Code

```
<%@ page contentType = "text/html; charset=windows-1252"
         pageEncoding = "windows-1252"
         import = "javax.portlet.*, java.util.*,
Portletenhance.Portletenhance,
Portletenhance.resource.PortletenhanceBundle"%>
@ <%@ taglib uri = "http://java.sun.com/portlet" prefix="portlet"%>
<portlet:defineObjects/>
<%
    PortletPreferences prefs = renderRequest.getPreferences();
    ResourceBundle res =
        portletConfig.getResourceBundle(renderRequest.getLocale());
%>
<form action="<portlet:actionURL/>" method="POST">
  <table border="0">
    <tr>
      <td width="20%">
        <p class="portlet-form-field" align="right">
          <%=  res.getString(PortletenhanceBundle.PORTLETCONTENT) %>
        </p>
      </td>
      <td width="80%">
        <input class="portlet-form-input-field"
               type="TEXT"
               name="<%= Portletenhance.PORTLETCONTENT_KEY %>"
               value="<%= Portletenhance.buildValue(prefs,
Portletenhance.PORTLETCONTENT_KEY) %>"
               size="20">
      </td>
    </tr>    <tr>
      <td width="20%">
        <p class="portlet-form-field" align="right">
          <%=  res.getString(PortletenhanceBundle.PORTLETTITLE) %>
        </p>
      </td>
      <td width="80%">
        <input class="portlet-form-input-field"
               type="TEXT"
               name="<%= Portletenhance.PORTLETTITLE_KEY %>"
               value="<%= prefs.getValue(Portletenhance.PORTLETTITLE_KEY,
res.getString("javax.portlet.title")) %>"
               size="20">
      </td>
    </tr>
    <%
    String[] str = {"Portlet Content"};
    str = prefs.getValues("portletContent",str);
```

```
%>
<tr><td width="20%">
<p class="portlet-form-field" align="right">
Content
</p>
</td><td width="80%">
<textarea rows="10" cols="60" class="portlet-form-input-field"
  name="portletContent"><%
for (int i=0; i<str.length; i++)
{%><%= (i<str.length-1) ? str[i]+", " : str[i] %><%}%>
</textarea>
</td></tr>
    <tr>
      <td colspan="2" align="center">
        <input class="portlet-form-button" type="submit"

name="<%=Portletenhance.OK_ACTION%>"

value="<%=res.getString(PortletenhanceBundle.OK_LABEL)%>">
        <input class="portlet-form-button" type="submit"

name="<%=Portletenhance.APPLY_ACTION%>"

value="<%=res.getString(PortletenhanceBundle.APPLY_LABEL)%>">
      </td>
    </tr>
  </table>
</form>
```

8. In the Application Navigator, right-click *portletName*.java and choose **Open.**
9. In the visual editor, click the **Source** tab and add the following two lines of code (indicated in bold) to the processAction method:

```
// Save the preferences.
PortletPreferences prefs = request.getPreferences();
String param = request.getParameter(PORTLETTITLE_KEY);
prefs.setValues(PORTLETTITLE_KEY, buildValueArray(param));
String contentParam = request.getParameter("portletContent");
if (contentParam != null)
{
  prefs.setValues("portletContent", buildValueArray(contentParam));
}
prefs.store();
```

10. Redeploy the portlet.

    Notice that JDeveloper automatically saves and compiles the code before deploying the portlet. For a reminder of how to perform this step, see Chapter 33, "Testing and Deploying Your Portlets."
11. Reload the page that contains the portlet and you can see that the portlet now displays the text Portlet Content, which was one of the changes you made.
12. Click the **Personalize** link to see the form field that you added. Enter some text in this field and close the dialog. You can see the new text displayed in the portlet.

30.1.2 How to Implement Navigational Parameters (WSRP 2.0)

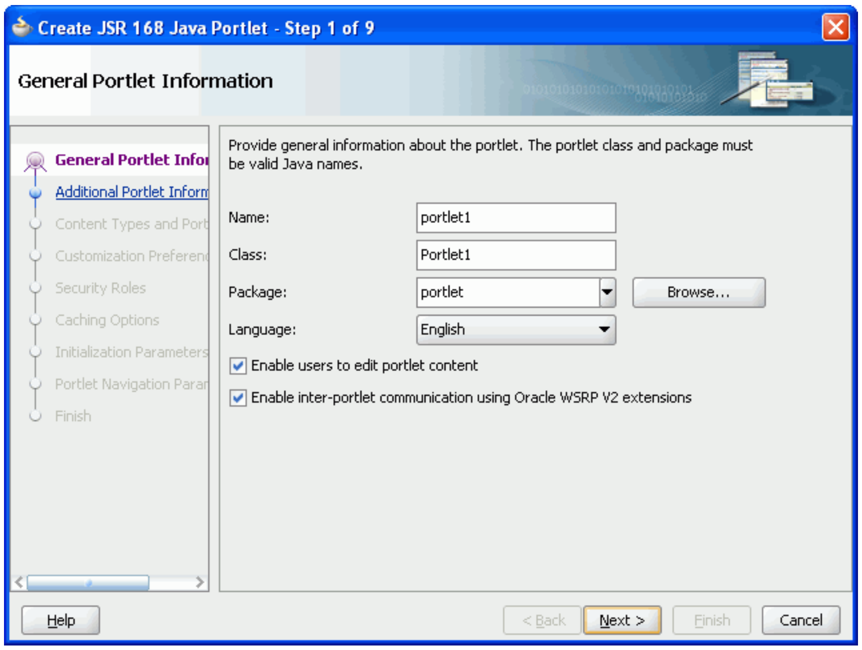
While JSR 168 and WSRP 1.0 do not address public portlet parameters, WSRP 2.0 introduces navigational parameters to enable inter-portlet communication. With the release of the new portlet Application Programming Interface (API) standard, JSR 286, the need for vendor-specific API extensions will not be necessary any more. Until then, you are required to use the Oracle-specific portlet container and API extensions.

Using the Create JSR 168 Java Portlet wizard, you can easily create a portlet with navigational parameters. When you register the producer and drop the portlet on a page, the portlet's parameters are automatically linked to page variables.

To add public parameters to JSR 168 portlets using WSRP 2.0 navigational parameters:

1. Start JDeveloper.
2. In the Application Navigator, open the application under which you want to create the portlet.
3. Right-click the project under which you want to create the portlet and choose **New.**
4. In the New Gallery, expand **Web Tier,** select **Portlets** and then **Standards-based Java Portlet (JSR 168),** and click **OK.**
5. In the General Portlet Information page of the Create JSR 168 Java Portlet wizard, select Enable inter-portlet communication using Oracle WSRP V2 extensions as shown in Figure 30–1.

Figure 30–1 Creating a Portlet with Navigational Parameters



Provide general information about the portlet. The portlet class and package must be valid Java names.

Name: portlet1

Class: Portlet1

Package: portlet

Language: English

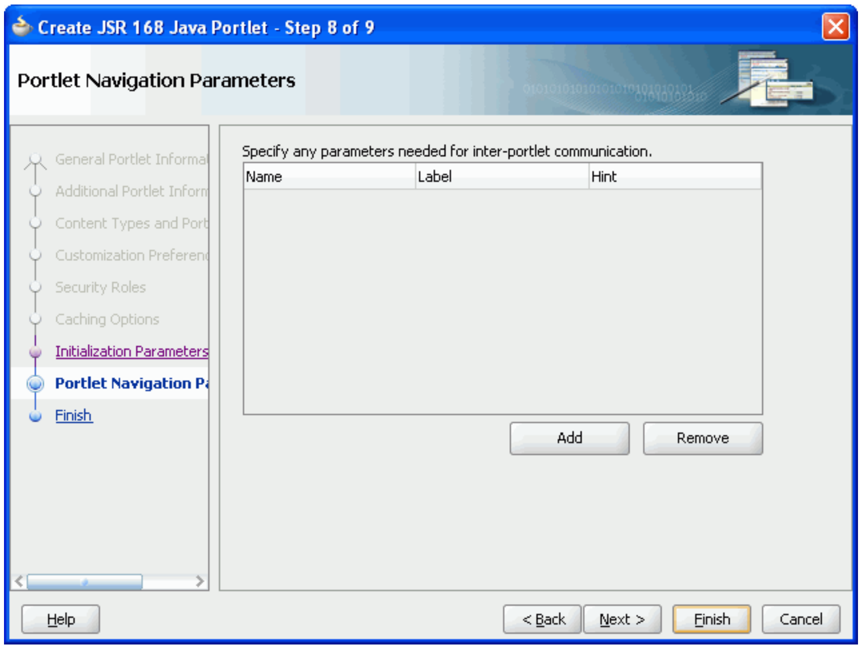
[✓] Enable users to edit portlet content

[✓] Enable inter-portlet communication using Oracle WSRP V2 extensions

6. Proceed through the wizard until you reach the Portlet Navigation Parameters page. For basic information about going through the wizard, see Section 29.2.1, "How to Create a JSR 168 Java Portlet."
7. In the Portlet Navigation Parameters page (Figure 30–2), click **Add.**

   This adds a new row to the table of parameters.

Figure 30–2 Portlet Navigation Parameters Page of Portlet Wizard



Specify any parameters needed for inter-portlet communication.

8. Replace the default values in the **Name, Label,** and **Hint** fields with something more meaningful for your parameter.

9. Add more parameters as required and then click **Finish.**

10. In the Applications Navigator, right-click **oracle-portlet.xml** and choose **Open.**

    You should see entries for the parameters that you added when you created the portlet, as shown in Example 30–3.

Example 30–3 oracle-portlet.xml Sample, Navigational Parameters

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<portlet-app-extension
 xsi:schemaLocation="http://xmlns.oracle.com/portlet/oracle-portlet-app"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://xmlns.oracle.com/portlet/oracle-portlet-app">
    <portlet-extension>
        <portlet-name>portlet1</portlet-name>
        <navigation-parameters>
            <name>Parameter_01</name>
            <type>xsi:string</type>
            <label xml:lang="en">Parameter 1</label>
            <hint xml:lang="en">First parameter.</hint>
        </navigation-parameters>
        <navigation-parameters>
            <name>Parameter_02</name>
            <type>xsi:string</type>
            <label xml:lang="en">Parameter 2</label>
            <hint xml:lang="en">Second parameter.</hint>
        </navigation-parameters>
        <portlet-id>1164232649525</portlet-id>
    </portlet-extension>
</portlet-app-extension>
```

11. Add code to the portlet that utilizes the parameters.

When the parameters are included in the `oracle-portlet.xml` file, you can access and set them as you would any normal render parameter, and the values are automatically exposed by the consumer.

For an example of a portlet used to set the values of navigation parameters, see the Parameter Form portlet, provided with the sample WSRP producer.

12. You can use the following APIs to read parameters passed to the portlet.

    ```
    String param1 = request.getParameter("Parameter_01");
    String param2 = request.getParameter("Parameter_02");
    ```

    For an example of a portlet used to read and display the values of navigation parameters, see the Parameter Display portlet, provided with the sample WSRP producer.

13. Application developers can add these portlets to a page and link them using the navigational parameters. For more information, see Section 9.7, "Contextually Linking WSRP 2.0 Portlets."

30.1.3 How to Implement Export/Import of Customizations (WSRP 2.0)

Another new feature that arrives with WSRP 2.0 is the ability to keep customizations with portlets when moving them from one deployment to another. Customizations are portlet preferences that are set in edit defaults mode. For example, suppose that you create a portlet and then customize its title within your development environment. If you have enabled export and import for that portlet and its producer, then the customized title is transported along with the portlet when you deploy it in production environment. If you do not enable export and import, then all customizations are lost when you transport the portlet from one deployment environment to another.

To implement export or import for a portlet and its producer:

1. Start JDeveloper.
2. In the Application Navigator, open the application that contains the portlet.
3. Expand the project that contains the portlet.
4. Right-click the `oracle-portlet.xml` file for the portlet's producer and choose **Open**.
5. Ensure that the `allow-export` and `allow-import` tags are included for each portlet and the producer as shown in Example 30–4, adding them if necessary.

Example 30–4 oracle-portlet.xml Sample, Export/Import

```
<portlet-app-extension xsi:schemaLocation="./oracle-portlet-app.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="./oracle-portlet-app.xsd">
    <portlet-extension>
        <portlet-name>portlet1</portlet-name>
        <navigation-parameters>
            <name>Parameter_01</name>
            <type>xsi:string</type>
            <label xml:lang="en">First Parameter</label>
            <hint xml:lang="en">hint0</hint>
        </navigation-parameters>
        <navigation-parameters>
            <name>Parameter_02</name>
            <type>xsi:string</type>
```

```
                <label xml:lang="en">Second Parameter</label>
                <hint xml:lang="en">hint1</hint>
            </navigation-parameters>
            <navigation-parameters>
                <name>Parameter_03</name>
                <type>xsi:string</type>
                <label xml:lang="en">Third Parameter</label>
                <hint xml:lang="en">hint2</hint>
            </navigation-parameters>
            <portlet-id>1</portlet-id>
            <allow-export>true</allow-export>
            <allow-import>true</allow-import>
            <portlet-name>portlet2</portlet-name>
            <navigation-parameters>
                <name>Parameter_01</name>
                <type>xsi:string</type>
                <label xml:lang="en">First Parameter</label>
                <hint xml:lang="en">hint0</hint>
            </navigation-parameters>
            <navigation-parameters>
                <name>Parameter_02</name>
                <type>xsi:string</type>
                <label xml:lang="en">Second Parameter</label>
                <hint xml:lang="en">hint1</hint>
            </navigation-parameters>
            <navigation-parameters>
                <name>Parameter_03</name>
                <type>xsi:string</type>
                <label xml:lang="en">Third Parameter</label>
                <hint xml:lang="en">hint2</hint>
            </navigation-parameters>
            <portlet-id>2</portlet-id>
            <allow-export>true</allow-export>
            <allow-import>true</allow-import>
        </portlet-extension>
        <allow-export>true</allow-export>
        <allow-import>true</allow-import>
    </portlet-app-extension>
```

30.1.4 How to Implement Rewritten URLs for Resource Proxy

Resource proxying is the standard way to retrieve resources with WSRP. To avoid problems with URLs within your portlet, you can set a flag to rewrite all of the URLs within a specified resource. For example, if have an HTML fragment that contains URLs, then you could set this flag to rewrite its URLs taking into account the WSRP resource proxying.

To indicate that URLs should be rewritten, set the `PortletRequest` attribute, `oracle.portlet.server.resourceRequiresRewriting`, to `true`. For example, you might include code similar to the excerpt in Example 30–5 to use resource proxying for a URL that you are encoding. Note that typically you would want to encapsulate this code within a method to avoid repeating it for every URL individually.

Example 30–5   Resource Proxy for WSRP

```
request.setAttribute("oracle.portlet.server.resourceRequiresRewriting",
     Boolean.TRUE);
String url = response.encodeURL(pathToResourceForRewriting);
request.removeAttribute("oracle.portlet.server.resourceRequiresRewriting");
```

If you do not specifically set `oracle.portlet.server.resourceRequiresRewriting`, then it defaults to `false`, meaning that URLs are not rewritten. You must explicitly activate the feature by setting this attribute to `true`.

30.1.5 How to Implement Stateless Resource Proxying

If you have out of protocol resources that do not require rewriting, you may want to use stateless resource proxying. Stateless resource proxying means that the URLs returned to the browser do not need to include portlet IDs or any other contextual information. This increases the cache hit ratio for such resources. You might find stateless resource proxying useful for functionality such as static JavaScript files, static images, and so on.

To indicate that stateless proxying should be used, set the `PortletRequest` attribute `oracle.portlet.server.useStatelessProxying` to `true`. For example, you might include code similar to the excerpt in Example 30–6 to use stateless proxying for a URL that you are encoding. Note that typically you would want to encapsulate this code within a method to avoid repeating it for every URL individually.

Example 30–6 Stateless Resource Proxying

```
request.setAttribute("oracle.portlet.server.useStatelessProxying", Boolean.TRUE);
String url = response.encodeURL(pathToResource);
request.removeAttribute("oracle.portlet.server.useStatelessProxying");
```

You can also use the following constant:

```
oracle.portlet.server.containerimpl.PortletRequestImpl.USE_STATELESS_PROXYING
```

This is shown in Example 30–7.

Example 30–7 Stateless Resource Proxying Using Constant

```
request.setAttribute(PortletRequestImpl.USE_STATELESS_PROXYING, Boolean.TRUE);
String url = response.encodeURL(pathToResource);
request.removeAttribute(PortletRequestImpl.USE_STATELESS_PROXYING);
```

If you do not specifically set `oracle.portlet.server.useStatelessProxying`, it defaults to `false`. You must explicitly activate the feature by setting this attribute to `true`.

30.1.6 How to Disable Java Object Cache for Preference Store Access

In some cases, you may prefer to avoid the use of the Java Object Cache by your WSRP preference store. You can configure the use of caching by the WSRP preference store with the following JNDI variable:

```
oracle/portal/wsrp/server/disableJavaObjectCache
```

By default, this variable is set to `false`. You can set the variable yourself in the `web.xml` file for you portlet application as follows:

```
<env-entry>
 <env-entry-name>oracle/portal/wsrp/server/disableJavaObjectCache</env-entry-name>
 <env-entry-type>java.lang.Boolean</env-entry-type>
 <env-entry-value>true</env-entry-value>
</env-entry>
```

For more information about setting JNDI variables, see Section 30.2.4.2, "Setting JNDI Variable Values."

30.1.7 How to Implement Security for JSR 168 Portlets

You can secure JSR 168 portlets that are deployed to a WSRP producer by configuring security at the WSRP producer end and the client end. For information about securing a JSR 168 portlet through its WSRP producer, see Section 24.12, "Securing Identity Propagation Through WSRP Producers with WS-Security."

30.2 Enhancing PDK-Java Portlets

When you have built your initial portlet in the Portlet Wizard as described in Section 29.2.3, "How to Create a PDK-Java Portlet," you will want to enhance it. You can find the JavaDoc reference for the PDK-Java on OTN at:

`http://www.oracle.com/technology/products/webcenter/portlet_download.html`

Some of the more important enhancements that you might want to perform are as follows:

- Adding portlet modes. For more information, see Section 30.2.1, "How to Add Portlet Modes."
- Adding parameters. For more information, see Section 30.2.2, "How to Implement Public Parameters."
- Using JNDI variables. For more information, see Section 30.2.4, "How to Use JNDI Variables."
- Accessing session information. For more information, see Section 30.2.5, "How to Access Session Information."
- Enhancing portlet performance. For more information, see Section 30.2.6, "How to Enhance Portlet Performance with Caching."

The source code for many of the examples referenced in this section is available as part of the Portlet Developer's Kit (PDK).

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

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

30.2.1 How to Add Portlet Modes

In the Portlet Wizard, you add portlet modes by checking boxes on the wizard pages. For more information about using the wizard, see Section 29.2.3, "How to Create a PDK-Java Portlet." For each portlet mode that you select in the wizard, a basic skeleton is created. If you want to add a portlet mode after creating the portlet, you can do that manually by updating `provider.xml` and HTML or JSPs in JDeveloper.

The principles of implementing portlet modes using `RenderManager` are the same for all modes.

More On OTN

For more detailed information about the PDK runtime classes used in this section, see the JavaDoc on OTN:

`http://www.oracle.com/technology/products/webcenter/portlet_download.html`

For more information about the syntax of `provider.xml`, see the provider JavaDoc, also available on OTN.

Before You Begin

The steps that follow assume that you have:

- Built a portlet using the wizard, successfully registered the producer, and added the portlet to the page.

To add a portlet mode:

1. Start JDeveloper.
2. In the Application Navigator, open the application that contains the portlet.
3. Expand the project that contains the portlet.
4. Expand the **Web Content** node and then the **htdocs** node and then the node for the producer.
5. Right-click the node for the portlet and choose **New.**

   The node for the portlet is located under **Web Content > htdocs > *provider***.
6. In the New Gallery, expand **Web Tier**, select **HTML** or **JSP**, and click **OK**.

   You must create an HTML file or a JSP for each mode to add to your portlet. For example, to implement Help mode, you would create an HTML file to provide the help content.
7. In the resulting dialog, enter a file name for the HTML file or JSP and click **OK**.
8. In the visual editor, edit the content of the page to implement the desired functionality.

   For example, for Help mode, you could add the following HTML:

   ```
   <p>This is the <i>Help</i> mode of your portlet!</p>
   ```
9. In the Application Navigator, right-click the `provider.xml` file for the provider that owns the portlet and choose **Open**.

   The `provider.xml` file is located under **Web Content > WEB-INF > providers > *provider***.
10. Change the value of the appropriate tag for the mode you are adding to `true`.

    For example, if you are adding Help mode, change the `hasHelp` tag as follows:

    ```
    <hasHelp>true</hasHelp>
    ```

    This indicates to the PDK Framework that a link or icon to that mode should be rendered.
11. Add the code to point to the HTML page or JSP that you created earlier for the mode.

    For example, for the Help page, add the following code:

    ```
    <helpPage>/htdocs/myprovider/myportlet/myHelpPage.html</helpPage>
    ```
12. Save your changes.
13. Redeploy your portlet. For more information, see Chapter 33, "Testing and Deploying Your Portlets."

When you redeploy, JDeveloper automatically saves and compiles the code before deploying the portlet.

14. Copy the HTML file or JSP you created and the updated `provider.xml` file to the WLS instance where you plan to deploy the portlet.
15. Refresh the producer.
16. Refresh the page containing your portlet.

    You should now be able to access the new mode. For example, if you added Help mode, you should be able to click the Help link.

30.2.2 How to Implement Public Parameters

PDK-Java and Oracle WebCenter Framework provide public and private portlet parameters to enable portlet developers to easily write reusable, complex portlets. The Portlet Wizard in JDeveloper creates portlets that are set up to use parameters. 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, see Section 27.2.11, "Public Portlet Parameter Support" and Section 27.2.12, "Private Portlet Parameter Support."

Before You Begin

The steps that follow assume that you have:

- Followed through and understood Section 29.2.3, "How to Create a PDK-Java Portlet."
- Built a portlet using the wizard and successfully added it to a page.

> **Note:** Each portlet is limited to 4K of data. The lengths of parameter and event names, display names, and descriptions all contribute toward this 4K limit. Hence, you should not use too many parameters and events for each portlet, or give them lengthy names and descriptions.

Using the Portlet Wizard, you can easily create a portlet with public parameters. When you register the producer and drop the portlet on a page, the portlet's parameters are automatically linked to page variables.

To create a portlet with public parameters to your portlet:

1. Start JDeveloper.
2. In the Application Navigator, open the application that contains the portlet.
3. Right-click the project under which you want to create your portlet, and choose **New.**

> **Note:** To create the portlet in an existing producer, right-click the producer's `provider.xml` file and choose **Add Portlet.** This takes you directly to the General Portlet Information page of the Create Oracle PDK-Java Portlet wizard.

4. In the New Gallery, expand **Web Tier,** select **Portlets** and then **Oracle PDK-Java Portlet,** and click **OK.**

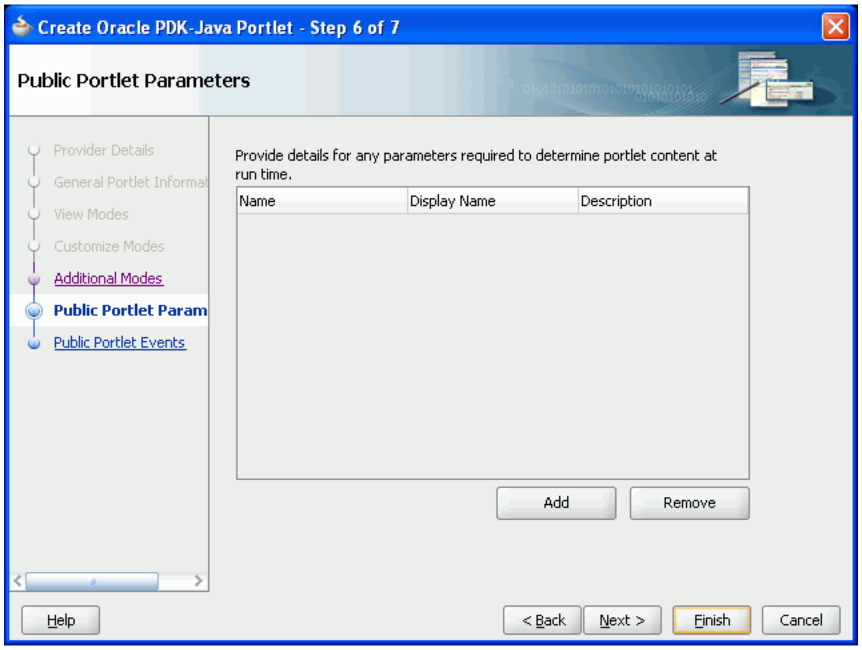
5. Proceed through the Create Oracle PDK-Java Portlet wizard as you normally would until you reach the Public Portlet Parameters page.

   See Section 29.2.3, "How to Create a PDK-Java Portlet" for basic information about going through the wizard.

6. In the Public Portlet Parameters page (Figure 30–3), click **Add.**

   This adds a new row to the table of parameters.

Figure 30–3 Public Portlet Parameters Page of Portlet Wizard



Provide details for any parameters required to determine portlet content at run time.

7. Replace the default values in the **Name, Display Name,** and **Description** fields with something more meaningful for your parameter.

8. Add more parameters as required and then click **Finish.**

9. In the Application Navigator, right-click the `provider.xml` file for the provider that owns the portlet and choose **Open.**

   The `provider.xml` file is located under **Web Content > WEB-INF > providers >** ***provider.***

   You should see entries for the parameters that you added on the Public Portlet Parameters page in the wizard, for example, as shown in Example 30–8.

Example 30–8 provider.xml Sample, Public Parameters

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<?providerDefinition version="3.1"?>
<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>MyPortlet</name>
        <title>My Portlet</title>
        <description>My Portlet Description</description>
        <timeout>40</timeout>
        <showEditToPublic>false</showEditToPublic>
        <hasAbout>false</hasAbout>
        <showEdit>true</showEdit>
        <hasHelp>false</hasHelp>
        <showEditDefault>false</showEditDefault>
        <showDetails>false</showDetails>
        <inputParameter class=
          "oracle.portal.provider.v2.DefaultParameterDefinition">
         <name>Parameter_01</name>
         <displayName>Parameter_01</displayName>
         <description>My first parameter</description>
        </inputParameter>
        <inputParameter class=
          "oracle.portal.provider.v2.DefaultParameterDefinition">
         <name>Parameter_02</name>
         <displayName>Parameter_02</displayName>
         <description>My second parameter</description>
        </inputParameter>
        <inputParameter class=
          "oracle.portal.provider.v2.DefaultParameterDefinition">
         <name>Parameter_03</name>
         <displayName>Parameter_03</displayName>
        </inputParameter>
        <renderer class="oracle.portal.provider.v2.render.RenderManager">
           <renderContainer>true</renderContainer>
           <renderCustomize>true</renderCustomize>
           <autoRedirect>true</autoRedirect>
           <contentType>text/html</contentType>
           <showPage>/htdocs/myportlet/MyPortletShowPage.jsp</showPage>
           <editPage>/htdocs/myportlet/MyPortletEditPage.jsp</editPage>
        </renderer>
        <personalizationManager class=
          "oracle.portal.provider.v2.personalize.PrefStorePersonalizationManager">
         <dataClass>
          oracle.portal.provider.v2.personalize.NameValuePersonalizationObject
         </dataClass>
        </personalizationManager>
   </portlet>
</provider>
```

10. In the Application Navigator, right-click the `portletnameShowPage.jsp` for your portlet and choose **Open.**

    The file is located under **Web Content > htdocs >** ***provider*** **>** ***portlet.***

    The portlet includes logic for retrieving the parameters, for example, as shown in Figure 30–9.

Example 30–9 ShowPage.jsp Sample

```
<%@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"
%>
<%
```

```
   PortletRenderRequest pReq = (PortletRenderRequest)
      request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
%>
<P>Hello <%= pReq.getUser().getName() %>.</P>
<P>This is the <b><i>Show</i></b> render mode!</P>
<%
   ParameterDefinition[] params =
       pReq.getPortletDefinition().getInputParameters();
%>
<p>This portlet's input parameters are...</p>
<table align="left" width="50%" ><tr><td><span
class="PortletHeading1">Name</span></td><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 < values.length; j++)
           {
               temp.append(values[j]);
               if (j + 1 != values.length)
               {
                   temp.append(", ");
               }
           }
           value = temp.toString();
       }
       else
       {
           value = "No values have been submitted yet.";
       }
%>
<tr>
  <td><span class="PortletText2"><%= name %></span></td>
  <td><span class="PortletText2"><%= value %></span></td>
</tr>
<%
   }
%>
</table>
```

11. Add logic to your portlet that allows it to submit parameter values entered by users.
12. Create a second portlet using the same steps that simply displays parameter values that it retrieves.
13. Register the producer with a WebCenter application. For more information, see Section 9.2.3, "How to Register an Oracle PDK-Java Portlet Producer."
14. Add the two portlets to a page in the application. For more information, see Section 9.3, "Adding Portlets to a Page."

15. In the Structure panel of the Application Navigator, right-click an element of the page and choose **Go to Page Definition.**

    The page definition should look similar to Example 30–10. Notice the variables at the page level and the parameters at the portlet level (indicated in bold).

Example 30–10 Page Definition File Sample

```
<?xml version="1.0" encoding="UTF-8"?>
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
                version="10.1.3.38.90" id="untitled1PageDef"
                Package="view.pageDefs">
  <executables>
    <variableIterator id="variables">
      <variable Name="portlet1_Parameter_01" Type="java.lang.Object"/>
      <variable Name="portlet1_Parameter_02" Type="java.lang.Object"/>
      <variable Name="portlet1_Parameter_03" Type="java.lang.Object"/>
    </variableIterator>
    <portlet id="portlet1"
       portletInstance="/oracle/adf/portlet/PdkPortletProducer1_1153936627784
          /applicationPortlets/Portlet1_abfc5a10_010c_1000_8003_82235f50d831"
        class="oracle.adf.model.portlet.binding.PortletBinding"
        xmlns="http://xmlns.oracle.com/portlet/bindings">
      <parameters>
        <parameter name="Parameter_01" pageVariable="portlet1_Parameter_01"/>
        <parameter name="Parameter_02" pageVariable="portlet1_Parameter_02"/>
        <parameter name="Parameter_03" pageVariable="portlet1_Parameter_03"/>
      </parameters>
    </portlet>
  </executables>
</pageDefinition>
```

16. Run the WebCenter application.

17. Enter values in the first portlet and the same values should be displayed in the second portlet.

30.2.3 How to Implement Private Parameters

In some cases, you might need a parameter that is known only to the portlet instance. These parameters are known as private parameters because they have no connection to the page and are known only to the portlet. Private parameters often come in handy when you are building navigation for your portlet. For example, if you have a portlet made up of multiple pages, then you can use these private parameters to jump to another resource of the portlet.

30.2.3.1 About Private Parameters

Private parameters are used in classic Web applications to pass information from links or forms in the browser back to the server. The server in turn takes actions and returns the appropriate content. For example, if the user of a dictionary Web site asks for information about hedgehogs, then the URL submitted to the server might append a private parameter as follows:

```
http://dictionary.reference.com/search?q=Hedgehog
```

If the server is responsible for rendering the whole page and the client communicates directly with the server, then this form of URL works well. In a custom WebCenter application, the client does not communicate directly with portlets. Instead, Oracle WebCenter Framework mediates between the client and the portlet. Moreover, because most pages have multiple portlets, Oracle WebCenter Framework communicates with multiple portlets.

For example, suppose a page contains two portlets, a thesaurus portlet and a dictionary portlet. Both portlets use `q` as a parameter to record the search queries made by the user. If the user queries the thesaurus portlet, then the URL used to rerequest the page with the updated thesaurus portlet must contain the thesaurus portlet's parameter, `q`. The thesaurus parameter must also be distinguished from dictionary portlet parameter 1, which performs the same function for that portlet. An example URL with the properly qualified thesaurus parameter might look something like the following:

```
http://host/portal/page?_pageid=33,1&_dad=portal&_schema=PORTAL
   &_piref33_38279_33_1_1.q=Hedgehog
```

Notice the fully qualified parameter name, `_piref33_38279_33_1_1.q`. It identifies the parameter and distinguishes it from other parameters on the page. Further, notice that the URL contains some parameters unrelated to any portlet. These parameters are untouched by the portlet because it does not own them.

You must ensure that the portlet meets the following criteria:

- It properly qualifies its own parameters when they are built into links and forms.
- It leaves unchanged any parameters that do not belong to it.

The following API call transforms an unqualified parameter name into a qualified parameter name:

```
HttpPortletRendererUtil.portletParameter(HttpServletRequest request, String
param);
```

`HttpPortletRendererUtil` is in the package `oracle.portal.provider.v2.render.http`.

For example:

```
qualParamQ = HttpPortletRendererUtil.portletParameter(r, "q");
```

To fetch the value of a portlet parameter from the incoming request, you can use the following API:

> **Note:** The API converts the parameter name into the qualified parameter name before fetching the value from the incoming request. Hence, you need not perform this step.

```
PortletRenderRequest.getQualifiedParameter(String name)
```

`PortletRenderRequest` is in the package `oracle.portal.provider.v2.render`.

For example:

```
valueQ = r.getQualifiedParameter("q");
```

The other aspect of a portlet's responsibilities with private parameters is to not disturb the parameters on the URL that it does not own. The utilities you may use to ensure adherence to this rule are discussed in Section 30.2.3.3, "Building Links with the Portlet URL Types" and Section 30.2.3.4, "Building Forms with the Portlet URL Types."

30.2.3.2 About Portlet URL Types

When a portlet renders itself, Oracle WebCenter Framework passes it various URLs, which the portlet can then use to render links. You can fetch and manipulate these URLs to simplify the task of creating links. The following is a list of the URLs provided to portlets:

- **PAGE_LINK** is a URL to the page upon which the portlet instance resides. You use this URL as the basis for all intraportlet links. If the portlet renders a link that navigates the user to another section of the same portlet, then this navigation must be encoded as a set of parameters using the PAGE_LINK.
- **DESIGN_LINK** is a URL to the portlet's personalization (Edit mode) page. A portlet's Edit and Edit Defaults modes are not rendered on the same page as the portlet. The Edit and Edit Defaults modes take over the entire browser window. The portlet's Edit and Edit Defaults modes are not necessarily accessible to every user. It represents a minimal, static framework in which the portlet is free to render its personalization options. This URL is only of use when rendering Personalize links.
- **BACK_LINK** is a URL to a useful return point from the current page where the portlet renders itself. For example, when the portlet is rendering it's personalization page (Edit mode), this link refers to the page on which the portlet resides and from which the user navigated to the personalization page. Consequently, it is the link you would encode in the buttons that accept or cancel the pending action. This URL is only useful for the desktop rendering of portlets (usually in Edit or Edit Defaults mode).

30.2.3.3 Building Links with the Portlet URL Types

To build links with Portlet URL types, you must access them and use them when writing portlet rendering code. To fetch the URL for a link, you call the following APIs in PDK-Java:

```
portletRenderRequest.getRenderContext().getPageURL()
portletRenderRequest.getRenderContext().getEventURL()
portletRenderRequest.getRenderContext().getDesignURL()
portletRenderRequest.getRenderContext().getLoginServerURL()
portletRenderRequest.getRenderContext().getBackURL()
```

In portlet navigation, you must add (or update) your portlet's parameters in the page URL. To perform this task, you can use the following API to build a suitable URL:

```
UrlUtils.constructLink(
    PortletRenderRequest pr,
    int linkType, -- UrlUtils.PAGE_LINK in this case
    NameValue[] params,
    boolean encodeParams,
    boolean replaceParams)
```

`UrlUtils` resides in the package called `oracle.portal.provider.v2.url`. Notice that you do not actually fetch the page URL yourself. Rather you use the supplied portlet URL type, `UrlUtils.PAGE_LINK`.

The parameter names in the `params` argument should be fully qualified. Moreover, if you properly qualify the parameters, `UrlUtils.constructLink` with the appropriate `linkType` does not disturb other URL parameters that are not owned by the portlet.

An alternative version of `UrlUtils.contructLink` accepts a URL as the basis for the returned URL. If you require an HTML link, then you can use `UrlUtils.constructHTMLLink` to produce a complete anchor element.

The following example portlet, `ThesaurusLink.jsp`, uses the parameter `q` to identify the word for which to search the thesaurus. It then creates links on the found, related words that the user may follow to get the thesaurus to operate on that new word. To see the initial submission form that sets the value of `q`, see the example in Section 30.2.3.4, "Building Forms with the Portlet URL Types."

```
<%
    String paramNameQ = "q";
    String qualParamNameQ =
    HttpPortletRendererUtil.portletParameter(paramNameQ);
    PortletRenderRequest pRequest = (PortletRenderRequest)
        request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
    String paramValueQ = pRequest.getQualifiedParameter(paramNameQ);
%>
<!-- Output the HTML content -->
<center>
    Words similar to <%= paramValueQ %>
    <br>
    Click the link to search for words related to that word.
    <br>
    <ul>
<%
        String[] relatedWords = Thesaurus.getRelatedWords(paramValueQ);
        NameValue[] linkParams = new NameValue[1];
        for (int i=0; i<=relatedWords.length; i++)
        {
            linkParams[0] = new NameValue(
                qualParamNameQ, relatedWords[i]);
%>
            <li>
            <b> <%= relatedWords[i] %> </b>
            <%= UrlUtils.constructHTMLLink(
                pRequest,
                UrlUtils.PAGE_LINK,
                "(words related to " + relatedWords[i] + ")",
                "",
                linkParams,
                true,
                true)%>
           </li>
<%
        }
%>
   </ul>
</center>
```

30.2.3.4 Building Forms with the Portlet URL Types

Use of portlet parameters in forms is little different from links. The following two fundamental rules continue to apply:

- Qualify the portlet's parameter names.
- Do not manipulate or remove the other parameters on the incoming URL.

In terms of markup and behavior, forms and links differ quite considerably. However, just as with links, PDK-Java contains utilities for complying with these two basic rules.

The code for properly qualifying the portlet's parameter name is the same as that described in Section 30.2.3.3, "Building Links with the Portlet URL Types." After all, a parameter name is just a string, whether it be a link on a page or the name of a form element.

Forms differ from links in the way you ensure that the other parameters in the URL remain untouched. Once you open the form in the markup, you can make use of the following APIs:

```
UrlUtils.htmlFormHiddenFields(pRequest,UrlUtils.PAGE_LINK, formName);
UrlUtils.htmlFormHiddenFields(someURL);
```

where `formName = UrlUtils.htmlFormName(pRequest,null).`

Note: Just as parameters in URLs and element names in forms require qualification to avoid clashing with other portlets on the page, form names must be fully qualified because any given page might have several forms on it.

The `htmlFormHiddenFields` utility writes HTML hidden form elements into the form, one form element for each parameter on the specified URL that is not owned by the portlet.

```
<INPUT TYPE="hidden" name="paramName" value="paramValue">
```

Thus, you need only to add their portlet's parameters to the form.

The other item of which you should be aware is how to derive the submission target of your form. In most cases, the submission target is the current page:

```
formTarget = UrlUtils.htmlFormActionLink(pRequest,UrlUtils.PAGE_LINK)
```

The value of `formTarget` can be the action attribute in an HTML form or the target attribute in a `SimpleForm`. Even though the method name includes HTML, it actually just returns a URL and thus you can use it in mobile portlets, too.

The following example form renders the thesaurus portlet's submission form. For the portlet that results from the submission of this form, see the example in Section 30.2.3.3, "Building Links with the Portlet URL Types."

```
<%
    String paramNameSubmit = "submit";
    String paramNameQ = "q";
    String qualParamNameQ =
        HttpPortletRendererUtil.portletParameter(paramNameQ);
    String qualParamNameSubmit =
    HttpPortletRendererUtil.portletParameter(paramNameSubmit);
    PortletRenderRequest pRequest = (PortletRenderRequest)
        request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
    String formName = UrlUtils.htmlFormName(pRequest,"query_form");
%>
<!-- Output the HTML content -->
<center>
    <b>Thesaurus</b>
    Enter the word you want to search for
    <form name="<%= formName %>" method="POST"
        action="<%= UrlUtils.htmlFormActionLink(pRequest,UrlUtils.PAGE_LINK) %>">
        <%= UrlUtils.htmlFormHiddenFields(pRequest,UrlUtils.PAGE_LINK, formName)%>
        <table><tr><td>
            Word of interest:
        </td><td>
            <input
                type="text"
                size="20"
                name="<%= qualParamNameQ %>"
                value="">
        </td></tr></table>
        <input type=submit name="<%= qualParamNameSubmit %>" Value="Search">
    </form>
</center>
```

30.2.3.5 Implementing Navigation within a Portlet

You can implement navigation within a portlet in one of three ways:

- Pass navigation information in rendered URLs using private portlet parameters. Branching logic within the portlet code then determines which section of the portlet to render based on the URL. This option represents a small extension to the thesaurus example presented in Section 30.2.3.3, "Building Links with the Portlet URL Types" and Section 30.2.3.4, "Building Forms with the Portlet URL Types." Basically, instead of performing thesaurus search operations using the value of parameter `q`, the portlet branches based on the parameter value and renders different content accordingly.
- Pass navigation information as described in the previous item but use PDK-Java to interpret the parameter and thus branch on its value. This option requires some further changes to the thesaurus example and is more fully explained later in this section.
- Use session storage to record the portlet state and private parameters to represent actions rather than explicit navigation. This method provides the only way that you can restore the portlet to it's previous state when the user navigates off the page containing the portlet. Once the user leaves the page, all private portlet parameters are lost and you can only restore the state from session storage, assuming you previously stored it there. This option requires that you understand and implement session storage. For more information about implementing session storage, see Section 30.2.5.1, "Implementing Session Storage."

The following portlet code comes from the multipage example in the sample producer of PDK-Java:

```
<portlet>
    <id>11</id>
    <name>Multipage</name>
    <title>MultiPage Sample</title>
    <shortTitle>MultiPage</shortTitle>
    <description>
        This portlet depicts switching between two screens all
        in an application page.
    </description>
    <timeout>40</timeout>
    <timeoutMessage>MultiPage Sample timed out</timeoutMessage>
    <renderer class="oracle.portal.provider.v2.render.RenderManager">
        <contentType>text/html</contentType>
        <showPage>/htdocs/multipage/first.jsp</showPage>
        <pageParameterName>next_page</pageParameterName>
    </renderer>
```

```
</portlet>
```

Notice that the value of `pageParameterName` is the name of a portlet parameter, `next_page`, that the PDK-Java framework intercepts and interprets as an override to the value of the `showPage` parameter. If the PDK-Java framework encounters the qualified version of the parameter when the multipage portlet is requested, then it renders the resource identified by `next_page` rather than `first.jsp`. Note that PDK-Java does not render the parameter within the portlet, that responsibility falls to the portlet.

You can modify the thesaurus example to operate with the use of this parameter. Specifically, you can use the form submission portlet to be the input for the thesaurus (the first page of the portlet), then navigate the user to the results page, which contains links to drill further into the thesaurus. The following examples illustrate these changes.

> **Note:** The example that follows is most useful for relatively simple cases, such as this thesaurus example. If your requirements are more complex (for example, you want to build a wizard experience), then you should consider using an MVC framework such as Struts. For information about how to build portlets from struts applications, see Section 30.4, "Building Struts Portlets."

`ThesaurusForm.jsp`:

```
<%
    PortletRenderRequest pRequest = (PortletRenderRequest)
        request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
    String paramNameSubmit = "submit";
    String paramNameQ = "q";
    String qualParamNameQ =
        HttpPortletRendererUtil.portletParameter(pRequest, paramNameQ);
    String qualParamNameSubmit =
    HttpPortletRendererUtil.portletParameter(pRequest, paramNameSubmit);
    String formName = UrlUtils.htmlFormName(pRequest,"query_form");
%>
<!-- Output the HTML content -->
<center>
    <b>Thesaurus</b>
    Enter the word you want to search for
    <form name="<%= formName %>" method="POST"
        action="<%= UrlUtils.htmlFormActionLink(pRequest,UrlUtils.PAGE_LINK) %>">
        <%= UrlUtils.htmlFormHiddenFields(pRequest,UrlUtils.PAGE_LINK, formName)
%>
        <%= UrlUtils.emitHiddenField(
                HttpPortletRendererUtil.portletParameter(request, "next_page"),
                "htdocs/path/ThesaurusLink.jsp" ) %>
        <table><tr><td>
            Word of interest:
        </td><td>
            <input
                type="text"
                size="20"
                name="<%= qualParamNameQ %>"
                value="">
        </td></tr></table>
        <input type=submit name="<%= qualParamNameSubmit %>" Value="Search">
    </form>
```

```
</center>
```

Notice how `next_page` must be explicitly set to point to `ThesaurusLink.jsp`. If you do not explicitly set `next_page` in this way, then it defaults to the resource registered in `provider.xml`, which is `ThesaurusForm.jsp`.

`ThesaurusLink.jsp`:

```
<%
    PortletRenderRequest pRequest = (PortletRenderRequest)
        request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
    String paramNameQ = "q";
    String paramNameNextPage = "next_page";
    String qualParamNameQ =
    HttpPortletRendererUtil.portletParameter(pRequest, paramNameQ);
    String qualParamNameNextPage =
        HttpPortletRendererUtil.portletParameter(pRequest, paramNameNextPage);
    String paramValueQ = pRequest.getQualifiedParameter(paramNameQ);
%>
<!-- Output the HTML content -->
<center>
    Words similar to <%= paramValueQ %>
    <br>
    Click the link to search for words related to that word.
    <br>
    <ul>
<%
        Thesaurus t = new Thesaurus();
        String[] relatedWords = t.getRelatedWords(paramValueQ);
        NameValue[] linkParams = new NameValue[2];
        linkParams[0] = new NameValue(
            qualParamNameNextPage, "htdocs/path/ThesaurusLink.jsp");
        for (int i=0; i<relatedWords.length; i++)
        {
            linkParams[1] = new NameValue(
                qualParamNameQ, relatedWords[i]);
%>
            <li>
            <b> <%= relatedWords[i] %> </b>
            <%= UrlUtils.constructHTMLLink(
                pRequest,
                UrlUtils.PAGE_LINK,
                "(words related to " + relatedWords[i] + ")",
                "",
                linkParams,
                true,
                true)%>
           </li>
<%
        }
%>
   </ul>
   <a href="<%=XMLUtil.escapeXMLAttribute
               (pRequest.getRenderContext().getPageURL())%>">
      Reset Portlet
   </a>
</center>
```

30.2.3.6 Restricting Navigation to Resources

One danger of implementing navigation with private parameters is that users could potentially navigate to portlet resources that you prefer to restrict. To control the resources to which the user may navigate, you can create a whitelist of acceptable resources to which a user may navigate. If you do not construct a whitelist to restrict navigation, then your portlet's resources are accessible according to the following default rules:

- Any path immediately beneath the servlet root context is navigable. For example, `/index.jsp` is accessible but `/WEB-INF/web.xml` is not.
- Any path under the `htdocs` directory is navigable. For example, both `/htdocs/multipage/first.jsp` and `/htdocs/lottery/lotto.jsp` are accessible.

To change this default behavior, you can add allowable path values to the provider definition file, `provider.xml`. For example, suppose you have a portlet where a JSP is used as a controller to forward requests to other pages depending on the `pageParameterName` private parameter. The XML excerpt in Example 30–11 would only allow resources under `/htdocs/multiportlet` to be shown. All other resources would be restricted.

Example 30–11 Whitelist Excerpt from the provider.xml File

```
<portlet class="oracle.portal.provider.v2.DefaultPortletDefinition">
   <id>1</id>
   <name>Multipage</name>
   <title>A MultiPage Portlet</title>
   ...
   <renderer class="oracle.portal.provider.v2.render.RenderManager">
          <contentType>text/html</contentType>
          <showPage>/htdocs/multiportlet/controller.jsp</showPage>
          <pageParameterName>show_page</pageParameterName>
          <allowedPath>/htdocs/multiportlet/*</allowedPath>
   </renderer>
</portlet>
```

The pattern matching rules for this feature are similar to URL pattern matching in `web.xml` files. The rules are as follows:

- To match the defined patterns, the resource path must exactly match unless wildcards are used.
- The first wildcard is for path matching and consists of a string beginning with `/` and ending with `/*`. Any resource whose path starts with this string is matched. For an `<allowedPath>` value of `/htdocs/sub1/*`, valid values of the private parameter would include `/htdocs/sub1/file.jsp` and `/htdocs/sub1/sub2/file2.jsp`.
- The second wildcard is for file type matching and consists of a string starting with `*.` and ending with a file extension. Valid values for the page parameter end with that file extension. For an `<allowedPathvalue>` of `*.jsp`, valid values of the private parameter would include `/htdocs/sub1/file.jsp` and `/htdocs/sub1/file2.jsp`.

30.2.4 How to Use JNDI Variables

When writing Java portlets, you may set deployment specific properties through the JNDI service such that their values may be retrieved from your producer code. In this way, you can specify any property in a producer deployment and then easily access it anywhere in your producer code.

You can use JNDI variables to change producer property values after the producer has been deployed. The environment entry must be declared in `web.xml`. It can then be updated on deployment using a deployment plan.

PDK-Java provides utilities to enable the retrieval of both producer and non-producer JNDI variables within a Java EE container.

30.2.4.1 Declaring JNDI Variables

You declare JNDI variables in the `web.xml` file for your producer. The format for declaring a JNDI variable is as follows:

```
<env-entry>
    <env-entry-name>variableName</env-entry-name>
    <env-entry-type>variableType</env-entry-type>
    <env-entry-value>variableValue</env-entry-value>
</env-entry>
```

The `env-entry-name` element contains the name by which you want identify the variable. `env-entry-type` contains the fully qualified Java type of the variable. `env-entry-value` contains the variable's default value.

30.2.4.1.1 Variable Types In the `env-entry-type` element, you should supply the fully qualified Java type of the variable, which is expected by your Java code. The Java types you may use in your JNDI variables are as follows:

- `java.lang.Boolean`
- `java.lang.String`
- `java.lang.Integer`
- `java.lang.Double`
- `java.lang.Float`

The Java EE container uses these type declarations to automatically construct an object of the specified type and gives it the specified value when you retrieve that variable in your code.

30.2.4.1.2 Variable Naming Conventions The PDK-Java defines environment variables that can be set at the individual producer service level or at the Web application level. To avoid naming conflicts between different producer services or different application components packaged in the same Web application, Oracle recommends you devise some naming convention.

Note: If you use the `EnvLookup` method, then you must use `oracle/portal/provider/service/property`. You cannot substitute your own company name or component in this case.

For example:

- Producer service specific names should be of the form:

  ```
  {company}/{component name}/{producer name}/{variable name}
  ```

- Shared names should be of the form:

```
{company}/{component name}/{producer name}/global
```

where:

- `{company}` is the name of the company owning the application.
- `{component name}` is the name of the application or component with which the producer is associated.
- `{producer name}` is the service name of the producer.
- `{variable name}` is the name of the variable itself.

As you can see, these naming conventions are similar to those used for Java packages. This approach minimizes the chance of name collisions between applications or application components. PDK-Java provides utilities that enable you to retrieve variables in this form without hard coding the service name of the producer into your servlets or JSPs. The service name need only be defined in the producer's WAR file. For more information about retrieving JNDI variables, see Section 30.2.4.3, "Retrieving JNDI Variables."

30.2.4.1.3 Examples The following examples illustrate producer variable names:

```
oracle/portal/myProvider/myDeploymentProperty
oracle/portal/myprovider/myProperties/myProperty
```

The following example illustrates non-producer variable names:

```
oracle/portal/myOtherProperty
```

30.2.4.2 Setting JNDI Variable Values

In your producer deployment, you may want to set a new value for some or all of your JNDI variables. You can perform this task by setting the values manually within a WLS deployment plan. Deployment plans can be created for the producer deployment through the WLS console.

To set variable values manually within a deployment plan:

1. Go to the producer deployment with the WLS console and create a new deployment plan if one does not already exist.
2. Edit the deployment plan XML file. For each deployment property you want to set, add the following variable definition directly under the <deployment-plan> tag:

   ```
   <variable-definition>
     <variable>
       <name>jndi_var_def</name>
       <value>false</value>
     </variable>
   </variable-definition>
   ```

3. To tie this variable definition to the actual JNDI variable, add the following for each property under the WEB-INF/web.xml module descriptor (oracle/portal/sample/rootDirectory is used as an example):

   ```
   <module-descriptor external="false">
     <root-element>web-app</root-element>
     <uri>WEB-INF/web.xml</uri>
     <variable-assignment>
       <name>jndi_var_def</name>

   <xpath>/web-app/env-entry/[env-entry-name="oracle/portal/sample/rootDirectory"]
   ```

```
/env-entry-value</xpath>
   </variable-assignment>
</module-descriptor>
```

4. Save and close the file.
5. Select Update on the producer deployment to apply the deployment plan for the new settings to take effect.

30.2.4.3 Retrieving JNDI Variables

JNDI is a standard Java EE technology. As such, you can access JNDI variables through Java EE APIs. For example:

```
String myVarName = "oracle/portal/myProvider/myVar"
String myVar = null;
try
{
   InitialContext ic = new InitialContext();
   myVar = (String)ic.lookup("java:env/" + myVarName);
}
catch(NamingException ne)
{
   exception handling logic
}
```

In addition to the basic Java EE APIs, PDK-Java includes a simple utility class for retrieving the values of variables defined and used by the PDK itself. These variables all conform to the naming convention described in Section 30.2.4.1.2, "Variable Naming Conventions" and are of the form:

```
oracle/portal/provider_service_name/variable_name
oracle/portal/variable_name
```

To use these APIs, you need only provide the *`provider_service_name`* and the *`variable_name`*. The utilities construct the full JNDI variable name, based on the information you provide, and look up the variable using code similar to that shown earlier and return the value of the variable.

The `EnvLookup` class (`oracle.portal.utils.EnvLookup`) provides two `lookup()` methods. One retrieves producer variables and the other retrieves non-producer variables. Both methods return a `java.lang.Object`, which can be cast to the Java type you are expecting.

The following code example illustrates the retrieval of a producer variable:

```
EnvLookup el = new EnvLookup();
String s = (String)el.lookup(myProviderName, myVariableName);
```

`myProviderName` represents the service name for your producer, which makes up part of the variable name. `myVariableName` represents the portion of the variable name that would come after the producer's service name. The example assumes the variable being retrieved is of type `java.lang.String`.

To retrieve a non-producer variable, you use the same code, you pass only one parameter, the variable name, to the `lookup()`, again excluding the `oracle/portal` prefix.

```
EnvLookup el = new EnvLookup();
Object o = el.lookup(myVariableName);
```

Table 30–1 shows the JNDI variables provided by default with PDK-Java. If you do not declare these variables, then PDK-Java looks for their values in their original locations (`web.xml` and the deployment properties file).

Table 30–1 PDK-Java JNDI Variables

Variable	Description
`oracle/portal/provider/`*`provider_name`*`/autoReload`	Boolean auto reload flag. Defaults to true.
`oracle/portal/provider/`*`provider_name`*`/definition`	Location of producer's definition file.
`oracle/portal/provider/global/log/logLevel`	Log setting (0 through 8). 0 being no logging and 8 the most possible logging.
`oracle/portal/provider/`*`provider_name`*`/maxTimeDifference`	Producer's HMAC time difference.
`oracle/portal/provider/<service_name>/resourceUrlKey`	Authentication key for resource proxying through the Parallel Page Engine. See *Oracle Fusion Middleware Administrator's Guide for Oracle Portal* for more information.
`oracle/portal/provider/`*`provider_name`*`/rootDirectory`	Location for producer personalizations. No default value.
`oracle/portal/provider/`*`provider_name`*`/sharedKey`	HMAC shared key. No default value.
`oracle/portal/provider/`*`provider_name`*`/showTestPage`	(non-producer) A Boolean flag that determines if a producer's test page is accessible. Defaults to true.
`oracle/portal/provider/global/transportEnabled`	A Boolean flag that determines whether Edit Defaults personalizations may be exported and imported.

30.2.5 How to Access Session Information

When a user accesses a page, it initiates a public, unauthenticated session and tracks information about the session across requests. If the user logs in, then this session becomes an authenticated session of the logged-in user. This session terminates when any of the following occur:

- 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.

As part of the metadata generation, all of the producers that contribute portlets to the page are contacted, if they specified during registration that they be called for some special processing. This call allows producers to do processing based on the user session, log the user in the producer's application if needed, and establish producer sessions. For producers, this call is referred to as `initSession`. As most Web-enabled applications track sessions using cookies, this API call enables the producer of the application to return cookies.

You can use 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 store only 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, then you may want to store it in the preference store instead. Some common applications of the session store are as follows:

- 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 producers 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 producers, you create a stateful application that tracks 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, then a stateful application may cause you problems. Stateful applications can affect the load-balancing and failover mechanism for your configuration. Even though you may deploy multiple middle-tiers, 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, affecting 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 the example in this section, session storage is used to count the number of times your portlet has rendered in Shared Screen mode.

Before You Begin

The steps that follow assume that you have:

- Followed through and understood Section 29.2.3, "How to Create a PDK-Java Portlet."
- Built a portlet using the wizard and successfully added it to a page.

30.2.5.1 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, you must 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 producer session may become invalid for the following reasons:

- The session times out.
- The `invalidate` method on `ProviderSession` is called.

- The 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. You can use the returned instance-specific name to write portlet instance data into the session.

More On OTN

For more detailed information about the PDK Framework classes, see the JavaDoc on OTN:

http://www.oracle.com/technology/products/webcenter/portlet_download.html

To implement session storage:

- Import `ProviderSession`, `PortletRendererUtil`, and `HttpPortletRendererUtil`.
- Retrieve the producer session.
- Read and write the session by accessing it from within your Java portlet.
- Set the session to true in `provider.xml`.
- Register the producer 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 must 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, then you display the value, increment the count, and store the new value. If the session is valid but no previously stored value exists, then 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, then 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 must update this flag in order for the producer to receive session information from the portal. You should only set this tag to true if you are using session information in your producer or portlets. By setting this flag to true, extra load is added to the producer calls.

   ```
   <provider class="oracle.portal.provider.v2.DefaultProviderDefinition">
   <session>true</session>
   ```

   More On OTN

   For more information about the syntax of `provider.xml`, see the provider JavaDoc on OTN:

   `http://www.oracle.com/technology/products/webcenter/portlet_download.html`

4. Register your producer with session support. For a reminder on how to register your portlet, see Section 33.4, "Registering and Viewing Your Portlet."

30.2.5.2 Viewing the Portlet

If you have not already added your Java portlet to a page, then do so now. Ensure that you perform the following tasks:

- Refresh the producer to accept the new changes.
- Re-login in case your session is no longer valid.

30.2.6 How to Enhance 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, as follows:

- If a valid version of the requested content exists in the cache, then 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, then the agent forwards the request to its destination and awaits the return of the content. The agent returns the content to the requester 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.

Producers generate dynamic content (that is, portlets) and they reside remotely from the custom WebCenter application instance on which they are deployed. As such, caching might improve their performance. The architecture lends itself well to caching. You can cache the portlets rendered by your producer and reuse the cached copies to handle subsequent requests, minimizing the overhead your producer imposes on page assembly.

The producer 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. Following are the three caching methods:

- **Expiry-based Caching**: When a producer 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. For more information, see Section 30.2.6.1, "Activating Caching" and Section 30.2.6.2, "Adding Expiry-Based Caching."
- **Validation-based Caching**: When a producer 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 producer a render request that includes the version identifier of the cached content. The producer determines whether the version identifier remains valid. If the version identifier is still valid, then the producer immediately sends a lightweight response to the PPE without any content, which indicates the cached version can be used. Otherwise, the producer 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 producer whether the content is up to date. The validity of the cached copy is determined by some logic in the producer. The advantage of this approach is that the producer controls the use of the cached content rather than relying on a fixed period. For more information, see Section 30.2.6.1, "Activating Caching" and Section 30.2.6.3, "Adding Validation-Based Caching."

Before You Begin

The steps that follow assume that you have:

- Followed through and understood Section 29.2.3, "How to Create a PDK-Java Portlet."
- Built a portlet using the wizard and successfully added it to a page.

30.2.6.1 Activating Caching

To use the caching features in your producers, 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 producers, over HTTP.

Usually, your administrator is responsible for the configuration details of caching.

30.2.6.2 Adding Expiry-Based Caching

Expiry-based caching is a simple caching scheme to implement, and can be activated declaratively in your XML producer definition. You can set an expiry time for the output of any `ManagedRenderer` you use by setting its `pageExpires` property to the number of minutes you want the output to be cached for. For example, suppose you want portlet output to be cached for one minute.

To add expiry-based caching:

1. After you have used the Portlet Wizard to build a portlet as described in Section 29.2.3, "How to Create a PDK-Java Portlet," edit the `provider.xml` file and set the `pageExpires` property tag of showPage to 1. This sets an expires entry of 1 minute for the portlet.

   By default the wizard generates a standard and compressed tag for `showPage`. 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>
   ```

   More On OTN

   For more information about the syntax of `provider.xml`, see the provider JavaDoc on OTN:

   `http://www.oracle.com/technology/products/webcenter/portlet_download.html`

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.

30.2.6.3 Adding Validation-Based Caching

Adding validation-based caching requires slightly more effort, but gives you explicit control over exactly which requests to your producer 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 must 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()`, 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, then 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 is 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, then the portlet shows the new content. If the content has not changed, then a cached version of the portlet is displayed.

30.3 Testing Portlet Personalization

If you have implemented personalization for your portlet, then the Personalize link only appears on the portlet for authenticated users. Hence, to test the personalization of a portlet (the Personalize link), you must have some form of security implemented for the application consuming the portlet. For testing purposes, you may prefer to just configure the most basic authentication possible. For more information, see Section 24.8, "Configuring Basic Authentication for Testing Portlet Personalization."

30.4 Building Struts Portlets

This section describes the framework for building Struts portlets with Oracle JDeveloper for use in a custom WebCenter application. You will learn how to build a Struts portlet from an existing application by adding the Struts Tag Library from the Oracle Portal Developer Kit (version 9.0.4.0.2 or later) to JDeveloper, then use the Oracle PDK Java Portlet wizard to create the Java portlet itself.

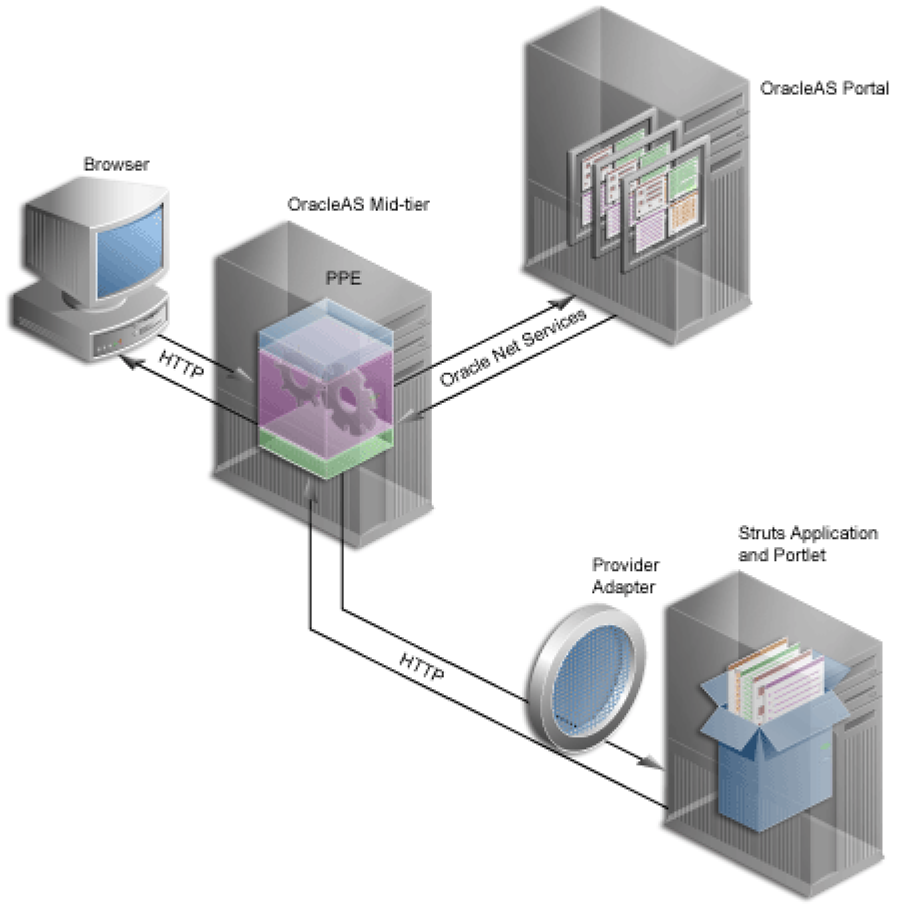
30.4.1 Creating a Struts Portlet

Oracle PDK contains 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. The PDK-Java extensions described in this section rely on Apache Struts 1.1.

The Oracle PDK-Java contains numerous examples and documents regarding the usage of the 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.

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 Java EE container. In WebCenter, the Struts application is called by the PPE, and not by the browser as compared to a standalone Struts application. When a portlet show call is made, the page engine sends a request to the Struts portlet renderer, which then forwards the request to the Apache Struts Controller servlet, as shown in Figure 30–4.

Figure 30–4 Integrating Struts Applications with Oracle Portal



The following code shows a portion of the producer 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>
...
```

More On OTN

For more information about the syntax of `provider.xml`, see the provider JavaDoc on OTN:

`http://www.oracle.com/technology/products/webcenter/portlet_download.html`

The `showPage` tag defines the business logic that is executed in the portlet mode of the portlet. The `showPage` of the Struts portlet contains two important components, which are as follows:

- 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.
- The `defaultAction` tag, which defines the Struts action that is used by default when the portlet is called for the first time.

The PDK-Java 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 enables the end user to use the application within the portal.

To create a Struts portlet, you must use the Oracle 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. Also, as 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 publish a part of an existing Struts application as a portlet, Oracle recommends that you first create a new view to serve as the Portal View of your application. This view uses existing objects (`Actions`, `ActionForm`, and so on) with a new mapping and new JSPs.

Note: Although Oracle recommends that you create a Portal View of your application, you could alternatively replace your application's struts tags with PDK-Java struts tags. This approach enables your application to run both as a standalone struts application and a portlet.

In this example, you will create a portlet that enables you to add a new entry to a blog. Figure 30–5 and Figure 30–6 show how you submit a blog and save a blog entry.

Figure 30–5 Submitting a Blog

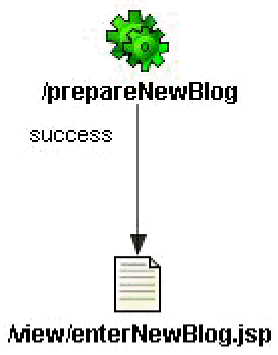
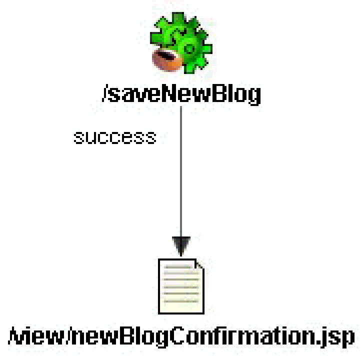


Figure 30–6 Saving a Blog Entry



`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>
```

30.4.1.1 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 redirects 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.

30.4.1.2 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 ensure the portlet meets the following criteria:

- Uses styles that enforce a consistent look and feel with the rest of the page.
- Contains HTML code that is allowed in HTML table cells (that is, no `<html>`, `<body>`, and `<frame>` tags).
- Renders application-aware links and forms. This is necessary to ensure that your Struts portlet renders its content inline, thus keeping your users within the page by rendering the requested content within the same portlet container.

To achieve inline rendering in your Struts portlet, you must use Oracle PDK tags:

```
<pdk-struts-html:form action="/portal/saveNewBlog.do">
...
...
</pdk-struts-html:form>
```

During the rendering of the portlet, a JSP tag (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
- `text`: renders fields on the form.
- `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 producer. To use this tag in Internet Web sites that have firewalls, you must make sure the producer is directly accessible from the Internet. If it is not possible, then you can deploy the images to the 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 Oracle 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.

30.4.1.3 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 use the wizard to build your Struts portlet.

To create a portlet:

1. In Oracle JDeveloper, open the Java Portlet Wizard to create an Oracle PDK Java Portlet.

   Note: The Java Portlet and Oracle PDK Java Portlet options are used to create JSR 168-compliant portlets and PDK-Java portlets respectively. Clicking **Java Portlet** or **Oracle PDK Java Portlet** opens the Java Portlet Wizard. For more information about opening the wizard, see Section 29.2.3, "How to Create a PDK-Java Portlet."

   ---

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`. This generates the skeleton of the portlet renderer class, `StrutsRenderer`.
5. As 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 from the **File** menu, select **Erase from Disk** in Oracle JDeveloper.
6. Edit the `provider.xml` and change the following properties:

   At the producer level, perform the following:

   - If you want users to always return to the same portlet state as when they left the container page, then you can configure the struts renderer to save the struts action in the struts context:

     ```
     <actionInSession>true</actionInSession>
     ```

     If you prefer that users always start from the beginning of the portlet when they return from outside the container page, then you should not save the struts action:

     ```
     <actionInSession>false</actionInSession>
     ```

     Note that this setting is the default behavior.

   - If the Struts application uses sessions (for example, the form sysnchronizer token mechanism is used or `<actionInSession>` is set to true), then enable session handling:

     ```
     <session>true</session>
     ```

   At the portlet level, perform the following:

   - 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>
     ```

   More On OTN

   For more information about the syntax of `provider.xml`, see the provider JavaDoc on OTN:

   `http://www.oracle.com/technology/products/webcenter/portlet_download.html`

30.4.1.4 Extending the Portlet to Add Business Logic

In your application, you should add code specific to your environment, such as the user's information, personalization, and localization. To do so, you can create a new `Action` class that is only called in context, and handles all business logic.

30.4.1.5 Registering the Producer

Now that your portlet is ready to be used by consumers, you must make it accessible by registering it. For information about how to register your PDK-Java portlet, see Section 33.4, "Registering and Viewing Your Portlet".

31 Creating Portlets with OmniPortlet

OmniPortlet is a data publishing portlet that you add to your application at design time, and customize at runtime. It provides a runtime, wizard-based experience to allow page designers to publish data from a variety of data sources, including SQL, XML, web services, spreadsheets, and web pages to a number of different layouts, such as customizable charts and tables.

This chapter covers the information you need to know to use OmniPortlet in your Oracle JDeveloper environment. For information on how to use the OmniPortlet wizard, refer to *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

This chapter includes the following sections:

- Section 31.1, "Introduction to OmniPortlet"
- Section 31.2, "Adding OmniPortlet to Your Application"

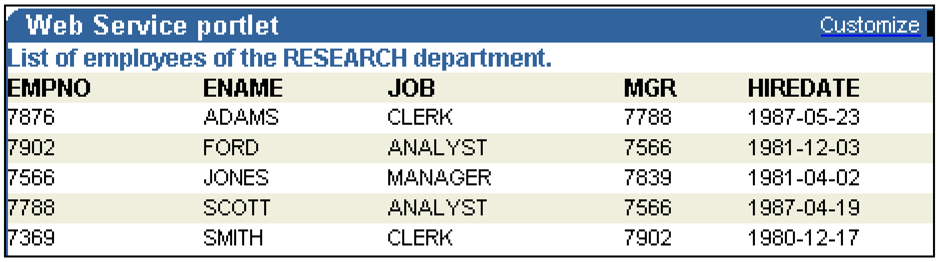
For information about registering and configuring OmniPortlet, see Section 9.2, "Registering Portlet Producers with a Custom WebCenter Application" and Section E.2, "OmniPortlet Configuration Tips."

Note: You can find more information about developing different types of portlets and information about producers and other portlet technologies in Chapter 27, "Overview of Portlets."

31.1 Introduction to OmniPortlet

OmniPortlet is a subcomponent of Oracle WebCenter Framework that enables developers to easily publish data from various data sources using a variety of layouts without writing any code. You can base an OmniPortlet on almost any kind of data source, such as a web service, a SQL database, spreadsheet (character-separated values), XML, and even application data from an existing web page. Figure 31–1 shows an example of a portlet created using OmniPortlet in a tabular layout.

Figure 31–1 Example of an OmniPortlet with the Tabular Layout



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

OmniPortlet enables the custom WebCenter application developer and component developer to do the following:

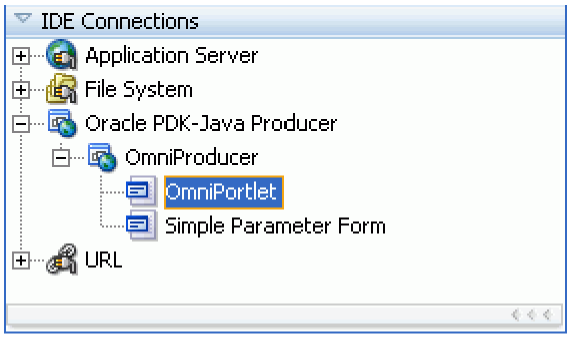
- Display data from multiple sources (CSV, XML, web service, SQL, and so on)
- Sort the data to display
- Format data using a variety of layouts (bulleted list, chart, HTML, and so on)
- Use portlet parameters
- Expose personalizable settings to page viewers

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, HTML, news, bulleted list, and form.

31.2 Adding OmniPortlet to Your Application

As described in Chapter 9, "Consuming Portlets," you can add an OmniPortlet to a page created through Oracle JDeveloper. OmniPortlet is included in the Integrated WebLogic Server (Integrated WLS) that is installed with Oracle JDeveloper. After you start the Integrated WLS, you can register the OmniPortlet producer by using the Register Oracle PDK-Java Producer wizard. When this producer is registered, the portlets become available on the Oracle JDeveloper Resource Palette or in the Application Resources panel. For example, if you register the OmniPortlet producer as "OmniProducer," OmniPortlet displays in your IDE Connections list in the Resource Palette, as shown in Figure 31–2.

Figure 31–2 OmniPortlet in the Resource Palette



Note: For more information about:

- Registering the producer, see Section 9.2, "Registering Portlet Producers with a Custom WebCenter Application."
- Installing, initializing, and stopping the default server and pointing Oracle JDeveloper to it, see Section 3.8, "Using Integrated WLS."
- Adding an instance of OmniPortlet to your page, see Section 9.3, "Adding Portlets to a Page."

After you register the portlet producer, you can simply drag OmniPortlet onto your `*.jspx` page.

Note: When you add an instance of OmniPortlet onto your page in Oracle JDeveloper, open the Property Inspector for the portlet and ensure that the `AllModesSharedScreen` and `RenderPortletInIFrame` properties are set as follows:

- **AllModesSharedScreen** is set to `False` to display the Customize and Personalize in full page size.
- **RenderPortletInIFrame** is set to `True` to display the OmniPortlet in its own iFrame in the View mode.

For information on configuring OmniPortlet in Oracle JDeveloper, refer to Section E.2, "OmniPortlet Configuration Tips."

31.3 Customizing OmniPortlet

After you add an OmniPortlet to your application at design time, you can customize the content, layout, as well as other options, by running your application to a browser. This section provides a high-level introduction to the runtime customization experience. For more detailed information on using and customizing this portlet, refer to *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

Note: For more information about configuring OmniPortlet, see Appendix E.2, "OmniPortlet Configuration Tips."

The OmniPortlet wizard initially contains five steps. When you first define your OmniPortlet, you set the data source type, data source options, filter options, view options, and layout. When you have completed these steps of the wizard, you can reenter the wizard by clicking the **Customize** link for the portlet. When you reenter the wizard, you can change the definitions on the Source, Filter, View, and Layout tabs.

Note: On the IBM Linux on Power platform, if the action buttons (Next, Previous, Finish, and Cancel) are minimized to dots when defining the OmniPortlet, increase the stack size shell limit to unlimited and restart the `WLS_portlet` instance. Run the following command to set the stack size shell limit to unlimited: `prompt> ulimit -s unlimited`.

You can use a number of different data sources with OmniPortlet:

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

For more information on using these data sources with OmniPortlet, refer to the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

31.4 Troubleshooting OmniPortlet Problems

This section provides information to help you troubleshoot problems you may encounter while using OmniPortlet.

Note: As the OmniPortlet producer exists and executes in a tier different from the WebCenter application and does not have access to the session information, you must expose XML files as PUBLIC in order for OmniPortlet to access them.

31.4.1 Cannot Define OmniPortlet Using the Define Link

You are not able to define the OmniPortlet at runtime.

Problem

OmniPortlet only supports a `RenderPortletInIFrame` value of `true`. This means that OmniPortlet must be rendered within an IFrame and therefore, the OmniPortlet property, `RenderPortletInIFrame`, must be set to `true`. In the Property Inspector, the `RenderPortletInIFrame` property is available under Display Options.

Currently, the `RenderPortletInIFrame` property has a value of `false` and, as a result, when you click the **Define** link at runtime, the **Type** tab may not display and you cannot proceed with defining the OmniPortlet.

Solution

You can choose **Customize** from the Action list to define OmniPortlet, or select the OmniPortlet in the Structure window in Oracle JDeveloper, and in the Property Inspector, set `RenderPortletInIFrame` to `true`.

32 Creating Content-Based Portlets with Web Clipping

Web Clipping is a publishing portlet that enables you to integrate any web application with your WebCenter application. Web Clipping is designed to give you quick integration by leveraging the existing user interface of a web application. 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 includes the information you are required to know to use Web Clipping in your Oracle JDeveloper environment. For information on how to use Web Clipping at runtime, refer to the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

This chapter includes the following sections:

- Section 32.1, "Introduction to Web Clipping"
- Section 32.2, "Adding Web Clipping to Your Application"
- Section 32.3, "Integrating Authenticated Web Content Using Single Sign-On"
- Section 32.4, "Advanced Features of Web Clipping"
- Section 32.5, "Current Limitations of Web Clipping"

For information about additional Web Clipping portlet producer configuration like repository and proxy settings and producer security, see Appendix E, "Additional Portlet Configuration."

Note: You can find more information about developing different types of portlets and information about producers and other portlet technologies in Chapter 27, "Overview of Portlets."

32.1 Introduction to Web Clipping

Web Clipping enables the 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 WebCenter application as a portlet.

Web Clipping portlets support the following features:

- **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**, enabling the Web Clipping engine to correctly identify a web clipping and deliver it as portlet content even if the web clipping gets reordered within the source page or if its character font, size, or style changes.
- **Reuse of 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 methods of form submission.
- **Personalization**, enabling page designers to expose input parameters that page viewers can modify when page viewers personalize the portlet. These parameters can be exposed as public parameters that a page designer can map as page parameters. This feature enables end users to obtain personalized clippings.
- **Integrated authenticated web content through Single Sign-On**, including integration with external applications, which enables you to leverage Oracle 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 authentication for each user. You can use this feature if proxy servers require authentication. You can specify proxy server authentication details including type (Basic or Digest) and realm in the `provider.xml` file. In addition, you can specify one of the following schemes for entering user credentials:
  - All users automatically log in using a user name and password you provide.
  - All users are required to log in using a user name and password they provide.
  - All public users (not authenticated into the WebCenter application) automatically log in using a user name and password you provide, while valid users (authenticated into the WebCenter application) log in by using a user name and password they provide.

  For more information, see Section E.3.2, "HTTP or HTTPS Proxy Configuration."
- **Navigation and clipping of HTTPS-based external web sites**, if appropriate server certificates are acquired. For information about server certificates, see Section E.3.3.1, "Adding Certificates for Trusted Sites."
- **Open Transport API for customizing authentication mechanisms to clipped sites**. By default, Web Clipping provider supports only HTTP challenge-based authentication methods like Basic and Digest and form submission logins. To support custom authentication methods like Kerberos proxy authentication, users can use the Web Clipping Transport API. For more information, see Section 32.4.1, "Using Web Clipping Open Transport API."
- **Clipping of page content from HTML 4.0.1 pages**, including the following:
  - Clipping of `<applet>`, `<body>`, `<div>`, `<embed>`, `<img>`, `<object>`, `<ol>`, `<span>`, `<table>`, and `<ul>` tagged content
  - Preservation of `<head>` styles and fonts, and Cascading Style Sheets (CSS)
  - UTF-8 compliant character sets
  - Navigation through hyperlinks (HTTP GET), form submissions (HTTP POST), frames, and URL redirection

- **Globalization Support** in URLs and URL parameters. For information about how Web Clipping determines the character set of clipped content, see Section 32.5, "Current Limitations of Web Clipping".

Web Clipping definitions are stored persistently in a repository. By default, in Oracle JDeveloper, the Web Clipping producer is configured to use Oracle Metadata Services (MDS), which is file-based, as a repository. You can alternatively select to use a database schema for your Web Clipping repository. For information about configuring a Web Clipping repository, see Section E.3.1, "Web Clipping Repository Configuration."

Any secure information, such as passwords, is stored in an encrypted form, according to the Data Encryption Standard (DES), by using Oracle encryption technology.

32.2 Adding Web Clipping to Your Application

As described in Chapter 9, "Consuming Portlets", you can add a Web Clipping portlet to a JSP document created through Oracle JDeveloper. The Web Clipping portlet producer is included in the Integrated WebLogic Server (WLS), the default server installed with Oracle JDeveloper.

Note: The Web Clipping portlet producer is also included within the `WLS_Portlets` managed server in the default domain in a full Oracle Fusion Middleware installation. Therefore, for your application, you can choose to register this producer from the `WLS_Portlets` managed server and the default producer in Integrated WLS.

In Oracle JDeveloper, after you start Integrated WLS, you can register the Web Clipping producer by using the Register Oracle PDK-Java Portlet Producer wizard. When this producer is registered, portlets become available either in the Application Resources panel or the Resource Palette. From here, you can drag portlets onto your `*.jspx` page.

For more information about:

- Registering the producer, see Section 9.2, "Registering Portlet Producers with a Custom WebCenter Application."

  The following is a sample URL that you may specify for the Web Clipping producer:

  ```
  http://localhost:7101/portalTools/webClipping/providers/webClipping
  ```

- Installing, initializing, and stopping the default server Integrated WLS, see Section 3.8, "Using Integrated WLS."
- Adding an instance of Web Clipping portlet to your page, see Section 9.3, "Adding Portlets to a Page."

After you register the Web Clipping portlet producer, you can simply drag the Web Clipping portlet onto your page. If you are using a `PanelCustomizable` or `ShowDetailFrame` component, then drag the portlet on top of that component instead of `af:form`. In the Structure pane, the Web Clipping portlet must display under the `PanelCustomizable` or `ShowDetailFrame` component. In the Page Editor, the portlet must display inside the `PanelCustomizable` or `ShowDetailFrame` component.

Note:

When you add an instance of Web Clipping onto your page in Oracle JDeveloper, open the Property Inspector for the portlet and ensure that the AllModesSharedScreen and RenderPortletInIFrame properties are set as follows:

- **AllModesSharedScreen** is set to `False` to display the Customize and Personalize in full page size. If you do not set this property to `False`, then when you personalize the Web Clipping portlet at runtime, the text displayed on the page may be distorted.
- **RenderPortletInIFrame** is set to `True` to display the Web Clipping in its own iFrame in the View mode.

When you run your application page, the Web Clipping portlet displays in your default browser. For information about how to work with a Web Clipping portlet at runtime, see the "Working with the Web Clipping Portlet" chapter in the *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.

Note: To clip SSL-enabled web sites, you must add certificates of those sites to the certificate store. You are not required to add certificates of SSL-enabled web sites that use Equifax, VeriSign, or Cybertrust certificates because these certificates are included in the default certificate store.

For information about adding certificates, see Section E.3.3.1, "Adding Certificates for Trusted Sites."

32.3 Integrating Authenticated Web Content Using Single Sign-On

You can leverage Oracle Single Sign-On to integrate content from external web sites that require authentication, into a Web Clipping portlet. This section walks you through an example that demonstrates how you can do this.

The example integrates a secured page from an external application named My Oracle Support into a Web Clipping portlet.

To integrate an external application:

1. Open your WebCenter application in Oracle JDeveloper.
2. Register the external application, specifying the authentication information by performing the following steps.
   a. In the Applications Navigator, right-click your project and choose **New**.
   b. In the New Gallery, expand the **General** category, select **External Application**, then select **External Application** on the right, and click **OK**.
   c. On the Name page of the Register External Application wizard, enter a name and display name for the external application, for example, `MyOracleSupport`.
   d. On the General page, enter the following details:
      – In **Login URL**, enter the URL of the application, for example, `http://metalink.oracle.com/metalink/plsql/sit_`

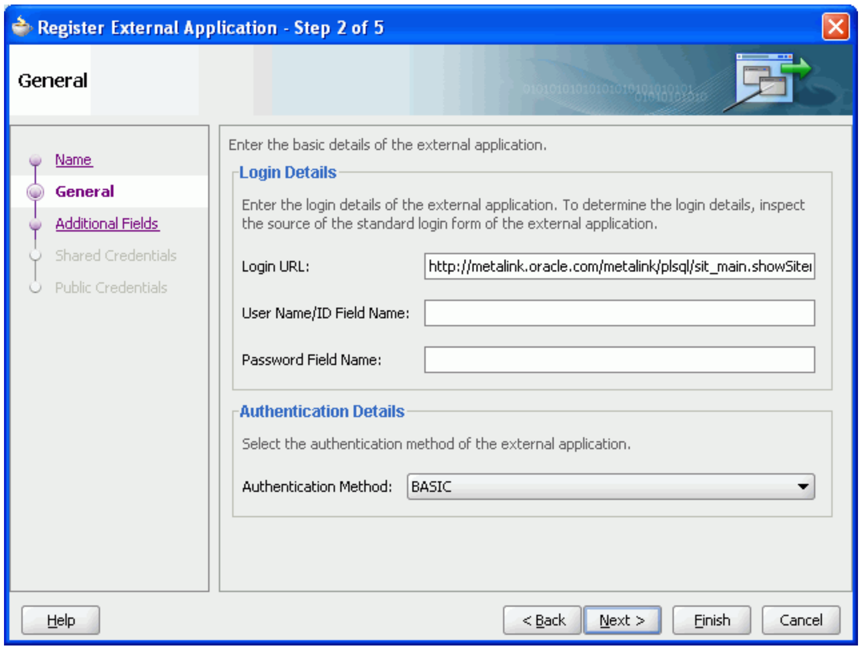
`main.showSitemap?p_showTitle=0`. To determine the URL, navigate to the desired application in a browser and note the URL.

For Form-based Authorization, view the source of the login page for the external application and note the URL to be accessed during the login action.

- In **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. If the Authentication method uses Basic Authentication, then you are not required to enter a field name. For MyOracleSupport, you can leave this field blank.
- In **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. If the authentication method uses Basic Authentication, then you are not required to enter a field name. For MyOracleSupport, you can leave this field blank.
- Select **BASIC** as the authentication method.

  Figure 32–1 shows the General page of the Register External Application wizard.

Figure 32–1 Registering an External Application


Register External Application - Step 2 of 5

General

Enter the basic details of the external application.

Login Details

Enter the login details of the external application. To determine the login details, inspect the source of the standard login form of the external application.

Login URL: http://metalink.oracle.com/metalink/plsql/sit_main.showSitei

User Name/ID Field Name:

Password Field Name:

Authentication Details

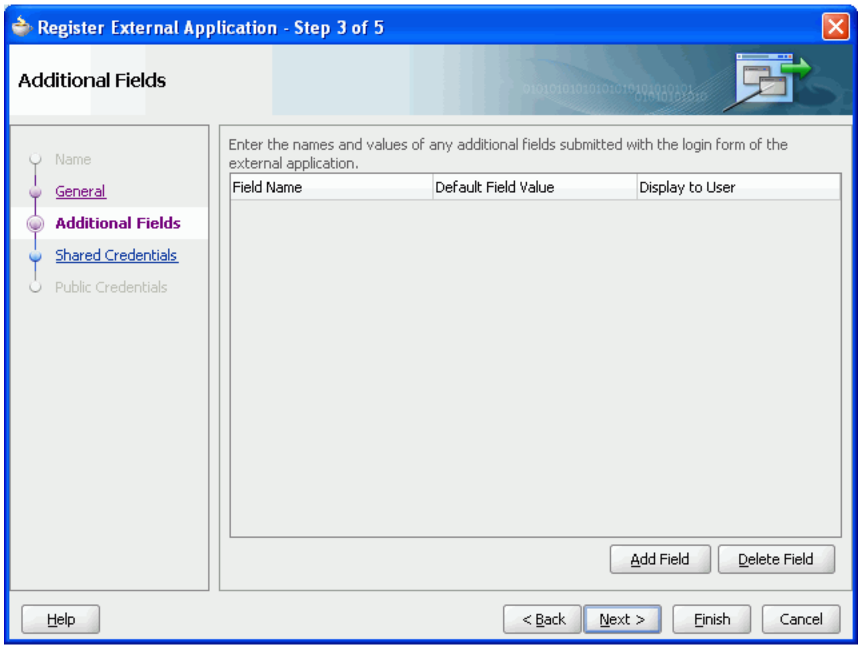
Select the authentication method of the external application.

Authentication Method: BASIC

e. On the Additional Fields page, you can specify names and values of any additional fields that are submitted with the login form of the external application. To specify a field name that is used to indicate a redirection URL, enter `redirectFieldName` in **Field Name** and specify the required value in **Default Field Value**. For this example, you are not required to enter additional fields.

Figure 32–2 shows the Additional Fields page of the Register External Application wizard.

Figure 32–2 Specifying Additional Fields



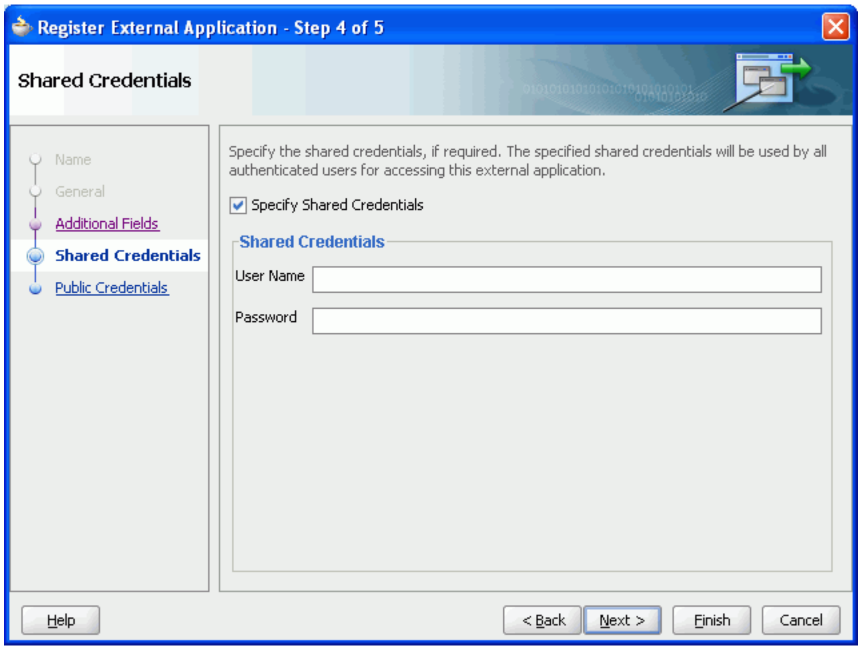
Enter the names and values of any additional fields submitted with the login form of the external application.

Field Name	Default Field Value	Display to User

f. On the Shared Credentials page, specify whether you want to use shared credentials that enable authenticated users to access the external application. To enable the shared credentials feature, select the **Specify Shared Credentials** checkbox. Then enter the user name and password and click **Next**.

Figure 32–3 shows the Shared Credentials page of the Register External Application wizard.

Figure 32–3 Specifying Shared Credentials



Specify the shared credentials, if required. The specified shared credentials will be used by all authenticated users for accessing this external application.

✓ Specify Shared Credentials

Shared Credentials

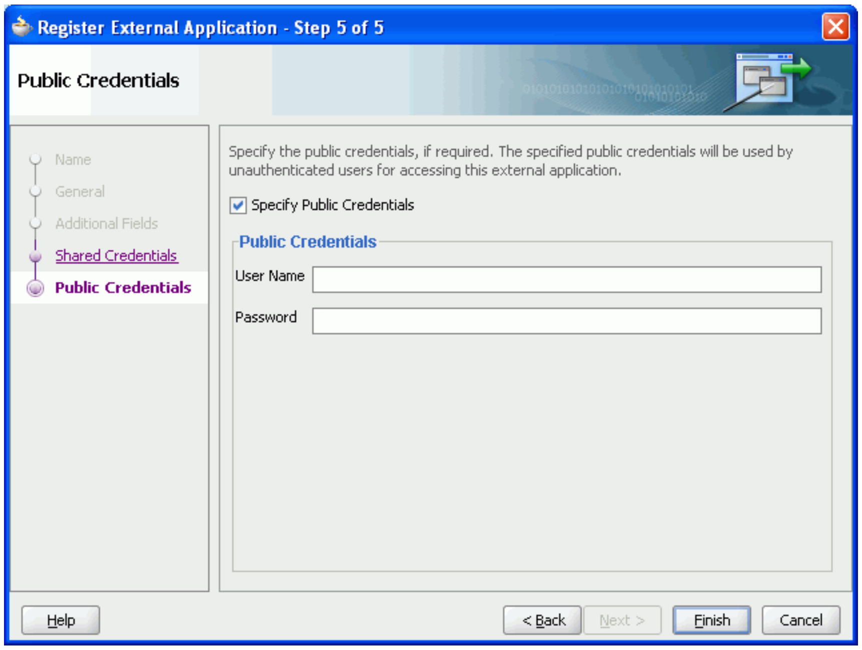
User Name

Password

g. On the Public Credentials page, specify whether unauthenticated users (public users) can access the external application. To enable the public credentials feature, select the **Specify Public Credentials** checkbox. Then enter the user name and password.

Figure 32–4 shows the Public Credentials page of the Register External Application wizard.

Figure 32–4 Specifying Public Credentials



Specify the public credentials, if required. The specified public credentials will be used by unauthenticated users for accessing this external application.

[✓] Specify Public Credentials

Public Credentials

User Name

Password

h. Click **Finish**.

3. For the Web Clipping portlet, register the Web Clipping producer by performing the following steps:

a. In the Applications Navigator, right-click your project and choose **New**.

b. In the New Gallery, expand the **Web Tier** category, select **Portlets**, then select **Oracle PDK-Java Producer Registration** from the list on the right, and click **OK**.

c. On the Specify Producer Name page of the Register Oracle PDK-Java Portlet Producer wizard, enter the producer name, for example `webClippingMyOracleSupport`.

d. Click **Next**.

e. On the Specify Connection Details page, specify the following details:

– In the **URL Endpoint** field, specify the URL for the Web Clipping producer in the following format:

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

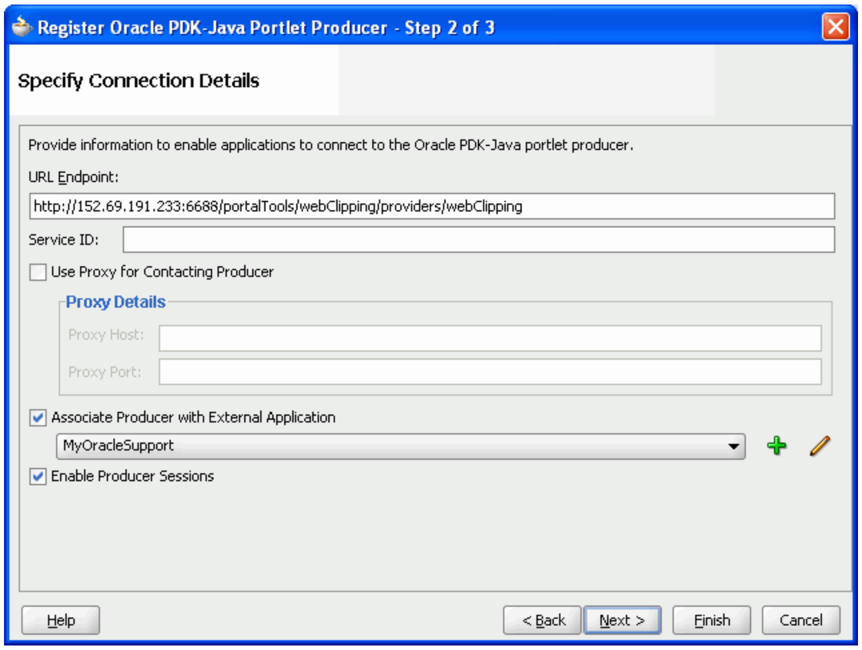
Note that *`host:port`* refers to the host and port where the Web Clipping producer is located.

– If you use a proxy server to contact web producers from your application, then select the **Use Proxy for Contacting Producer** checkbox and enter the proxy details.

– Select the **Associate Producer with External Application** checkbox, and from the list of values, select **MyOracleSupport** that you registered earlier. The **Enable Producer Sessions** checkbox also gets selected in this step.

  Figure 32–5 shows the Specify Connection Details page of the Register Oracle PDK-Java Portlet Producer wizard.

Figure 32–5 Associating a Web Clipping Producer with an External Application


Register Oracle PDK-Java Portlet Producer - Step 2 of 3

Specify Connection Details

Provide information to enable applications to connect to the Oracle PDK-Java portlet producer.

URL Endpoint: http://152.69.191.233:6688/portalTools/webClipping/providers/webClipping

Service ID:

[ ] Use Proxy for Contacting Producer

Proxy Details

Proxy Host:

Proxy Port:

[✓] Associate Producer with External Application

MyOracleSupport

[✓] Enable Producer Sessions

f. Click **Next**.

g. On the Specify Additional Registration Details page, specify the execution timeout, subscriber ID, and shared key values, if required.

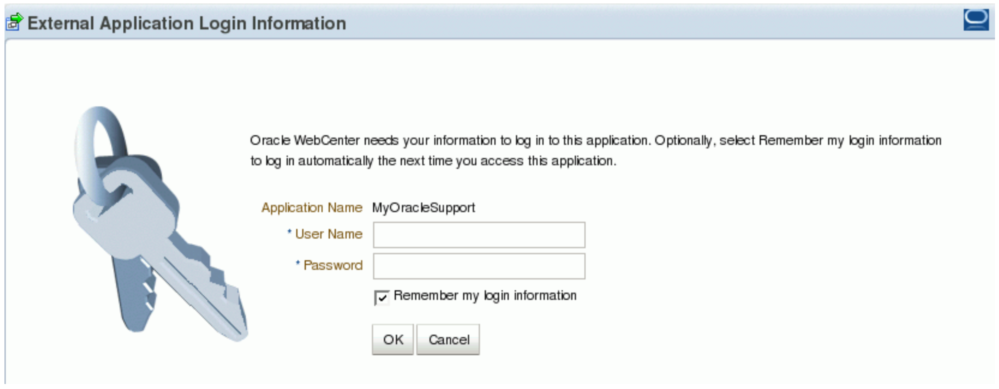
h. Click **Finish**.

4. Add a portlet to a `*.jspx` page, using the `webClippingMyOracleSupport` producer that you just registered.

5. Run the `*.jspx` page.

6. If you did not specify shared or public credentials for the external application representing `MyOracleSupport`, then the portlet contains an **Update login information** link. Click this link and enter your credentials. Then, click **OK** to log on to `MyOracleSupport`. (Figure 32–6)

Figure 32–6 Logging into the Integrated External Application



Oracle WebCenter needs your information to log in to this application. Optionally, select Remember my login information to log in automatically the next time you access this application.

Application Name MyOracleSupport

* User Name

* Password

[✓] Remember my login information

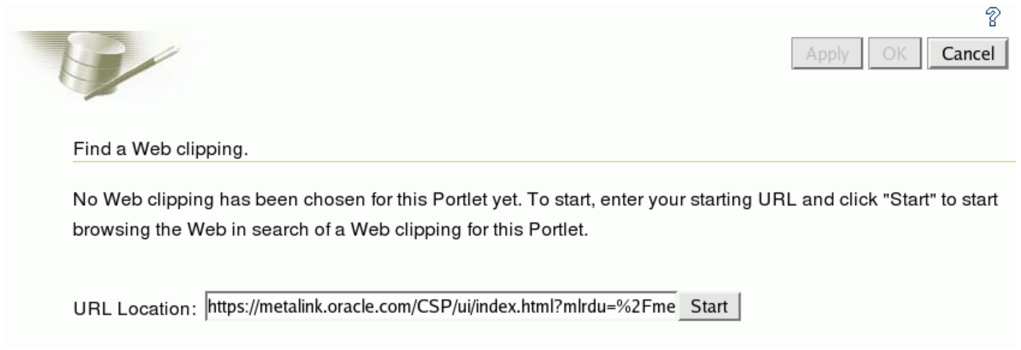
7. Select a section of a page that you intend to display in the Web Clipping portlet by performing the following steps:

   a. Click the **Actions** icon on the header of the Web Clipping portlet, and select **Customize**.

      The Find a Web Clipping page displays.

   b. In the **URL Location** field, the default URL for the external application is displayed. (Figure 32–7)

Figure 32–7 Finding a Web Clipping



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: https://metalink.oracle.com/CSP/ui/index.html?mlrdu=%2Fme

   c. Click **Start.** The Web Clipping Studio displays the page from the integrated external application.

   d. Browse to the web page that contains the content you want to clip. After you navigate to the required web page, click **Section** in the Web Clipping Studio banner.

      Web Clipping Studio divides the web page into clippable sections.

   e. At the top-left corner of the section you want to clip, click **Choose.**

   f. Web Clipping Studio displays a preview of your selected section. If it is the section you want to clip, then click **Select** on the Web Clipping Studio banner.

      The Web Clipping Studio displays the Find a Web clipping page, showing the properties of the clipping.

   g. From the **URL Rewriting** list, choose **Inline** to specify that link targets open inside the portlet, rather than in a new browser window.

h. Click **OK** to display the selected web clipping in the Web Clipping portlet on your page.

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

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

32.4 Advanced Features of Web Clipping

Web Clipping supports certain advanced features. You can configure custom authentication methods by using the Web Clipping Transport API and rewrite image links to use a resource proxy.

32.4.1 Using Web Clipping Open Transport API

To support custom authentication methods, users can use the Web Clipping Transport API. To extend the Web Clipping transport layer to support custom authentication methods, users must perform the following implementation and deployment procedures:

Implementation

Users can implement their own transport classes.

- Users can override two use cases of the `oracle.portal.wcs.transport.http.HttpTransportLiaison` interface. In Web Clipping, this interface is used to abstract the HTTP transport layer. By default, the two use cases of this interface are manifested by following implementations:
  - `HttpClientStudioTransportLiaison`, which handles HTTP transport in Web Clipping Studio mode
  - `HttpClientProviderTransportLiaison`, which handles HTTP transport in Web Clipping Producer show mode

  To support more authentication methods, users must override the `addRequestHeaders` methods for both the Studio and Provider `HttpClientTransportLiaison` implementations to add their own authentication-specific headers. For information, see *Oracle WebLogic Server Web Clipping API Reference*.

- Users must compile the new subclasses and package them into a JAR file. For example, to compile the new subclasses, users can use the following command:

  ```
  javac -classpath path_to_wcejar -d classes/
  ```

  Where, *`path_to_wcejar`* refers to the path to the `wce.jar` file.

  To create the JAR file, for example, users can run the following command from the `classes` directory:

  ```
  jar cvf ../mytransport.jar
  ```

  Where, *`mytransport.jar`* refers to the JAR file users want to create.

Deployment

Users must deploy the JAR file to support the custom authentication method.

To deploy the JAR file, users must perform the following tasks:

1. Place the JAR file into the class path or shared library that is used by the Web Clipping producer at runtime.
2. Register the transport class in the `web.xml` file for Web Clipping producer by making the following modifications to the context parameters defined for `HttpClientProviderTransportLiaison` and `HttpClientStudioTransportLiaison`:
   - Change the parameter value for `oracle.webclipping.provider.TransportLiaisonClass` to the name of the new class extended from the `HttpClientProviderTransportLiaison` class.
   - Change the parameter value for `oracle.webclipping.studio.TransportLiaisonClass` to the name of the new class extended from the `HttpClientStudioTransportLiaison` class.
3. Restart the producer server for the changes to take effect.

32.4.2 Rewriting Image Links to Use a Resource Proxy

Web Clipping enables users to rewrite image links to use a resource proxy. To enable this feature, you must add the following entry in the `web.xml` file of the Web Clipping producer:

```
<env-entry>
    <env-entry-name>oracle/webclipping/rewriteImageLink</env-entry-name>
    <env-entry-type>java.lang.Boolean</env-entry-type>
    <env-entry-value>false</env-entry-value>
</env-entry>
```

32.5 Current Limitations of Web Clipping

This section lists current limitations of Web Clipping.

- If the site that you intend to clip uses a large amount of JavaScript to manipulate cookies or uses the `document.write` JavaScript method to modify the HTML document being written, then you may not be able to clip content from the site.
- When you integrate with partner applications (by using `mod_osso`), you cannot clip directly through those partner applications in an authenticated manner. However, you can use partner applications through the external application framework.
- You cannot use the Web Clipping portlet to clip Oracle Portal pages and ADF pages. As a workaround, reregister the same producer in the destination portal and edit the portal manually.
- You cannot use the Web Clipping portlet to clip a web page that contains multiple frames, that is, a frameset.
- Note the following about Web Clipping and the use of a CSS:
  - If a web page contains multiple portlets that use a CSS, then they should not conflict if the CSS uses distinct style names, such as `OraRef`, to specify a style

within an HTML tag, rather than using an HTML tag name, such as `<A>`, as the name of the style.

  – If one portlet uses a CSS, and that CSS overwrites the behavior of HTML tags by using the name of the tag, such as `<A>`, as the name of the style, and a second portlet on the same page does not use a CSS, the second portlet is affected by the style instructions of the CSS of the first portlet.
  – If two portlets on the same page use a different CSS and each CSS overwrites the behavior of HTML tags by using the name of an HTML tag, such as `<A>`, as the name of the style, then the style that is displayed depends on the browser.

- Web Clipping checks for Globalization Support settings in the following way:
  1. Web Clipping checks the `Content-Type` in the HTTP header for the `charset` attribute. If this is present, then it assumes that this is the character encoding of the HTML page.
  2. If the `charset` attribute is not present, then it checks the HTML `META` tag on the page to determine the character encoding.
  3. If the HTML `META` tag is not found, then Web Clipping uses the `charset` in the previous browsed page. If this is the first page, then it defaults to the ISO-8859-1 character encoding.
  4. If the value of the `charset` for `Content-Type` or `META` tag is not supported (for example, if the `charset` was specified as `NONE`), then Web Clipping uses the default character set, ISO-8859-1, not the charset in the previously browsed page.
- To use the Web Clipping portlet, you must use Netscape 7.0 or later, Microsoft Internet Explorer 5.5 or later for Windows 2000, or Microsoft Internet Explorer 6.0 or later for Windows XP. If you use browser versions older than these, then you may encounter JavaScript errors.

33 Testing and Deploying Your Portlets

This chapter explains how to test and deploy JSR 168 and Oracle PDK-Java portlets. In a development environment, you can test your portlets on Integrated WebLogic Server (WLS), which comes packaged with Oracle JDeveloper, and also deploy portlet applications to an Oracle WebLogic Managed Server instance residing outside JDeveloper.

This chapter includes the following sections:

- Section 33.1, "Introduction to Portlet Deployment Testing in a Development Environment"
- Section 33.2, "Testing a Portlet Application on Integrated WebLogic Server"
- Section 33.3, "Deploying a Portlet Application to an Oracle WebLogic Managed Server Instance"
- Section 33.4, "Registering and Viewing Your Portlet"

For information about testing and deploying other custom WebCenter applications, see Chapter 25, "Testing and Deploying Your WebCenter Application".

For information about creating portlets, see Chapter 28, "Creating Portlets with the Oracle JSF Portlet Bridge" and Chapter 29, "Creating Portlets with the Portlet Wizard".

33.1 Introduction to Portlet Deployment Testing in a Development Environment

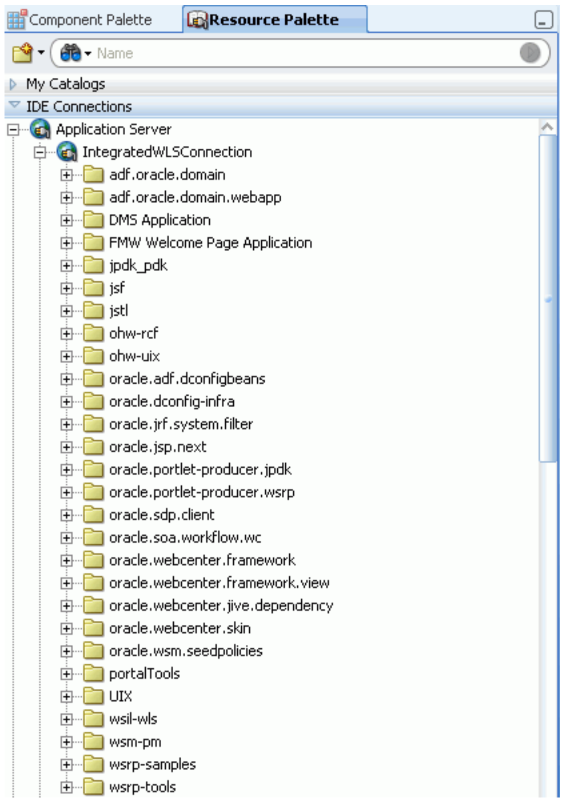
Before you deploy your portlet application, you are advised to test it in a development environment. Integrated WLS in JDeveloper enables you to test your portlets in a single step, without creating a deployment profile. To learn the benefits of testing applications on Integrated WLS, see Integrated WLS Server. An additional benefit is that portlet customizations that you perform at design time are maintained in your application workspace and become an integral part of the application source definition. These changes are packaged with the EAR file when the application is deployed to a WebLogic Managed Server instance (managed servers). This eliminates the requirement to export and import portlet customizations.

To test your portlet application on an Oracle WebLogic Managed Server instance, or to deploy for production, you must configure a connection to the managed server, create deployment plans, and then deploy to the server instance as described in this chapter. For information about deploying portlet applications using Oracle Enterprise Manager Fusion Middleware Control, WLS Administration Control, or WLST commands, see the chapter, "Managing Portlet Producers" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

33.2 Testing a Portlet Application on Integrated WebLogic Server

The connection to Integrated WLS (Default Server) connection is preconfigured and shows as **IntegratedWLSConnection** under the **Application Server** node in the **IDE Connections** panel of the Resource Palette, as shown in Figure 33–1. You can run multiple applications simultaneously and watch the progress of each application in the Run Manager panel. The Run Manager panel also lets you stop Default Server instances.

Figure 33–1 IntegratedWLSConnection in IDE Connections



Application Server
IntegratedWLSConnection
adf.oracle.domain
adf.oracle.domain.webapp
DMS Application
FMW Welcome Page Application
jpdk_pdk
jsf
jstl
ohw-rcf
ohw-uix
oracle.adf.dconfigbeans
oracle.dconfig-infra
oracle.jrf.system.filter
oracle.jsp.next
oracle.portlet-producer.jpdk
oracle.portlet-producer.wsrp
oracle.sdp.client
oracle.soa.workflow.wc
oracle.webcenter.framework
oracle.webcenter.framework.view
oracle.webcenter.jive.dependency
oracle.webcenter.skin
oracle.wsm.seedpolicies
portalTools
UIX
wsil-wls
wsm-pm
wsrp-samples
wsrp-tools

For more information about Integrated WLS Server, see Integrated WLS Server.

This section includes the following subsections:

- Section 33.2.1, "How to Test JSR 168 Portlets on Integrated WebLogic Server"
- Section 33.2.2, "What Happens When You Test JSR 168 Portlets on Integrated WebLogic Server"
- Section 33.2.3, "How to Test PDK-Java Portlet Applications on Integrated WebLogic Server"
- Section 33.2.4, "What Happens When You Test PDK-Java Portlet Applications on Integrated WebLogic Server"

33.2.1 How to Test JSR 168 Portlets on Integrated WebLogic Server

When you run your portlet application on Integrated WLS, an instance of Default Server starts.

To test a JSR 168 portlet application:

1. Right-click the `portlet.xml` page in the project folder and select **Run**. Running `portlet.xml` triggers packaging and deployment of your portlet application on an instance of Integrated WLS named after the application.

   Note: When the Integrated WLS instance stops, the application is undeployed, and therefore, becomes unavailable.

   For a more persistent testing scenario, you can deploy your portlet application to Integrated WLS by right-clicking the application and then selecting **Deploy**. This deploys your application to the Default Server instance so that it is always available when the Default Server is running. Thus, the application remains accessible for multiple consumer applications. If you choose this method, then you must first create deployment profiles, as described in Section 33.3.1, "How to Create Deployment Profiles". If you deploy your application to Integrated WLS, then the Deployment Configuration dialog displays to let you configure and customize deployment settings. The file system MDS repository pre-created by JDeveloper displays in the **Repository Name** field.

   ---

2. In the Select deployment type dialog, select **Yes** and then click **OK**.

   See the DefaultServer Log window to monitor the deployment progress. The DefaultServer - Log shows the URL of the application page. The WSRP producer URL uses the following syntax:

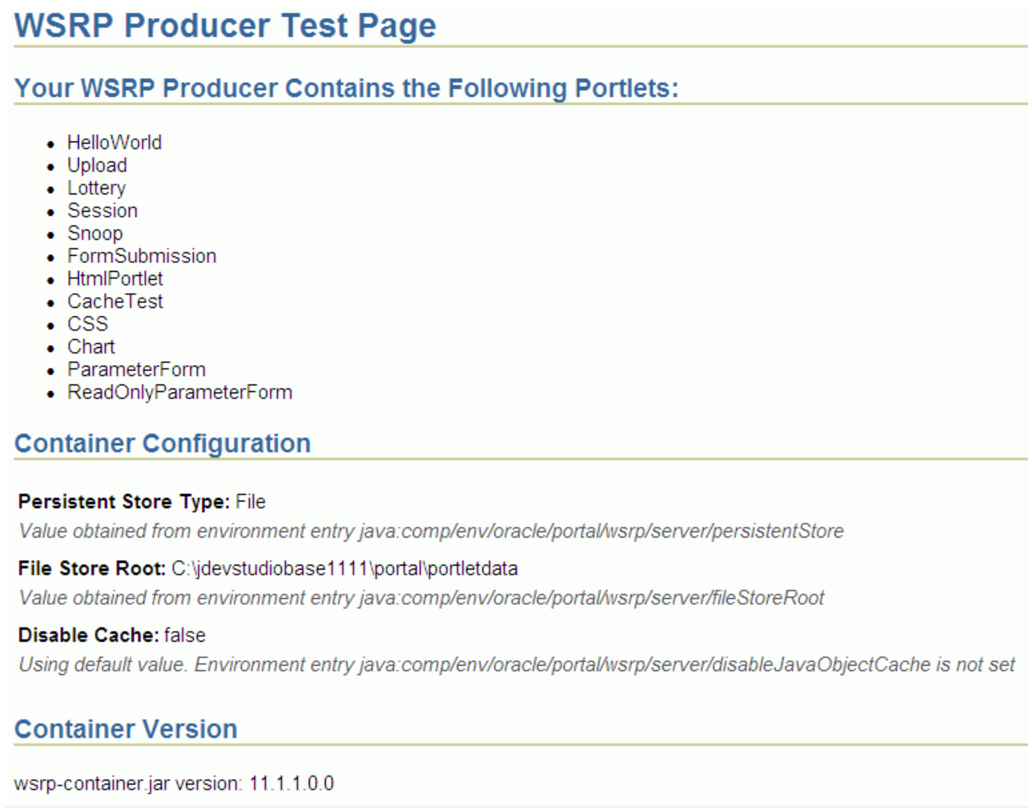
   ```
   http://host:port/applicationname-Portlets-context-root/info
   ```

   where:

   - `host` is the server to which your producer has been deployed.
   - `port` is the HTTP Listener port. Typically, it is `7101`. When the server is started, the port is displayed in the console.
   - `context-root` is the Web application's context root.

   A test page similar to Figure 33–2 displays in a browser window.

Figure 33–2 WSRP Producer Test Page



WSRP Producer Test Page

Your WSRP Producer Contains the Following Portlets:

- HelloWorld
- Upload
- Lottery
- Session
- Snoop
- FormSubmission
- HtmlPortlet
- CacheTest
- CSS
- Chart
- ParameterForm
- ReadOnlyParameterForm

Container Configuration

Persistent Store Type: File
Value obtained from environment entry java:comp/env/oracle/portal/wsrp/server/persistentStore

File Store Root: C:\jdevstudiobase1111\portal\portletdata
Value obtained from environment entry java:comp/env/oracle/portal/wsrp/server/fileStoreRoot

Disable Cache: false
Using default value. Environment entry java:comp/env/oracle/portal/wsrp/server/disableJavaObjectCache is not set

Container Version

wsrp-container.jar version: 11.1.1.0.0

Note: This procedure is for testing purposes only. After this procedure, you are required to register your producer as described in Section 33.4, "Registering and Viewing Your Portlet".

33.2.2 What Happens When You Test JSR 168 Portlets on Integrated WebLogic Server

When you run JSR 168 portlets on Integrated WLS, the following happens:

- `wsdls` and other configuration files are added to the WEB-INF directory to configure the portlets as a Web service.
- The `web.xml` file is updated with listener and server classes, filters, parameters, and other configurations that are required to run the JSR 168 portlet application successfully.

  For example, the `oracle.portlet.server.adapter.web.ServerContextListener` class, `WSRP_v2_PortletManagement_Service` and `WSRPBaseService` filters, and so on.
- Libraries required for JSR 168 portlets are added to the `weblogic.xml` file, for example, `oracle.portlet-producer.wsrp`.

These configurations vary depending upon the portlet requirements.

33.2.3 How to Test PDK-Java Portlet Applications on Integrated WebLogic Server

When you run your portlet application on Integrated WLS, an instance of Default Server starts.

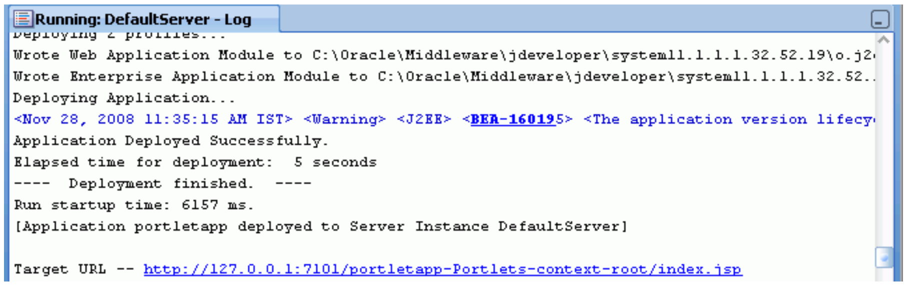
To test a portlet application, right-click a JSP page, for example, `index.jsp` in the project folder and select **Run**. Running `index.jsp` triggers packaging and deployment of your portlet application on an instance of Integrated WLS named after the application.

Note: When the Integrated WLS instance stops, the application is undeployed, and therefore, becomes unavailable.

For a more persistent testing scenario, you can deploy your portlet application to Integrated WLS by right-clicking the application and then selecting **Deploy**. This deploys your application to the Default Server instance so that it is always available when the default server is running. Thus, the application remains accessible for multiple consumer applications. If you choose this method, then you must first create deployment profiles, as described in Section 33.3.1, "How to Create Deployment Profiles". If you deploy your application to Integrated WLS, then the Deployment Configuration dialog displays to let you configure and customize deployment settings. The file system MDS repository pre-created by JDeveloper displays in the **Repository Name** field.

See the DefaultServer Log window to monitor the deployment progress. The DefaultServer - Log shows the URL of the application test page, as shown in Figure 33–3. The format of the URL is `http://`*`host:port/application_name`*`-Portlets-context-root/index.jsp`.

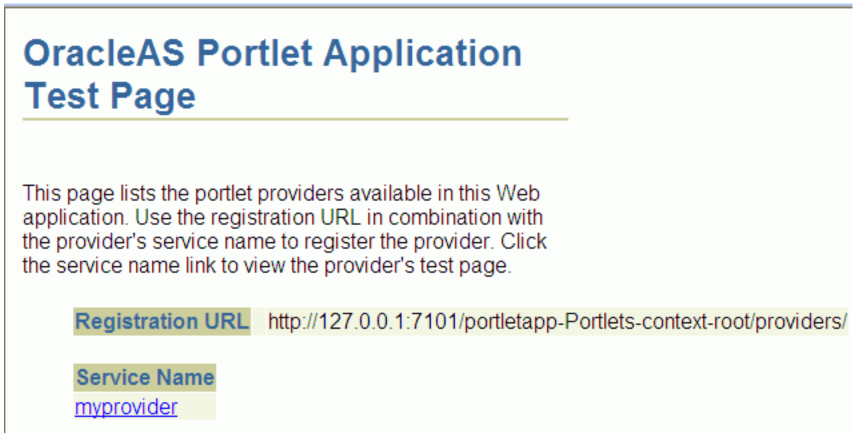
Figure 33–3 DefaultServer - Log



```
Deploying 2 profiles...
Wrote Web Application Module to C:\Oracle\Middleware\jdeveloper\system11.1.1.1.32.52.19\o.j2
Wrote Enterprise Application Module to C:\Oracle\Middleware\jdeveloper\system11.1.1.1.32.52..
Deploying Application...
<Nov 28, 2008 11:35:15 AM IST> <Warning> <J2EE> <BEA-160195> <The application version lifecy
Application Deployed Successfully.
Elapsed time for deployment:  5 seconds
----  Deployment finished.  ----
Run startup time: 6157 ms.
[Application portletapp deployed to Server Instance DefaultServer]

Target URL -- http://127.0.0.1:7101/portletapp-Portlets-context-root/index.jsp
```

The PDK-Java Application Test page displays in a browser window, as shown in Figure 33–4.

Figure 33–4 Portlet Application Test Page



OracleAS Portlet Application Test Page

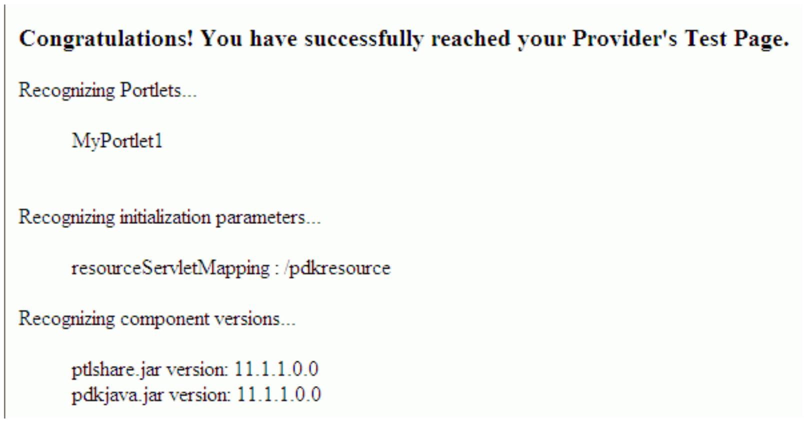
This page lists the portlet providers available in this Web application. Use the registration URL in combination with the provider's service name to register the provider. Click the service name link to view the provider's test page.

Registration URL http://127.0.0.1:7101/portletapp-Portlets-context-root/providers/

Service Name
myprovider

Click the link underneath **Service Name**. Your browser should open with a page similar to the one shown in Figure 33–5. The URL of this page is the one required to register the producer with another application.

Figure 33–5 Producer Test Page



Congratulations! You have successfully reached your Provider's Test Page.

Recognizing Portlets...

MyPortlet1

Recognizing initialization parameters...

resourceServletMapping : /pdkresource

Recognizing component versions...

ptlshare.jar version: 11.1.1.0.0
pdkjava.jar version: 11.1.1.0.0

Alternatively, you can construct the URL yourself as follows:

```
http://host:port/context-root/providers/producer_name
```

where:

`host` is the server to which your producer has been deployed.

`port` is the HTTP Listener port. Typically, it is `7101`. When the server is started, the port is displayed in the console.

`context-root` is the Web Application's Context Root, which you specified earlier and can be found in the WAR Deployment Profile Properties under General.

`producer_name` is the name of the portlet's producer. A WAR file may contain multiple producers, hence you should always include the name of the producer for clarity. Otherwise, you will get the default producer, which is the first producer created. The default producer is defined by the `_default.properties` file. This is created with the first producer in a project.

If you enter this URL in your browser, you should see a page similar to the one in Figure 33–5.

33.2.4 What Happens When You Test PDK-Java Portlet Applications on Integrated WebLogic Server

When you run PDK-Java portlets on Integrated WLS the following happens:

- Configuration files for PDK-Java portlets are added to the WEB-INF directory.
- The `web.xml` file is updated with listener and server classes, filters, parameters, and other configurations that are required to run the portlet application successfully.

  For example, the ResourceServlet pdkresource, the `oracle.portlet.server.service.ContextFilter` filter, and so on. These configurations vary depending upon the portlet requirements.
- Libraries required for PDK-Java portlets are added to the `weblogic.xml` file, for example, `oracle.portlet-producer.jpdk`.

33.3 Deploying a Portlet Application to an Oracle WebLogic Managed Server Instance

To test your portlet application on an Oracle WebLogic Managed Server instance, or to deploy it for production, you must first create an application WAR deployment profile, a deployment descriptor, and a connection to the Oracle WebLogic Managed Server instance. Before you deploy or test your portlet application to a managed server, ensure that the managed server is created using the correct template and it contains all the required shared libraries as described in the section "Creating and Provisioning a WebLogic Managed Server Instance" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

For information about deploying portlet applications using Oracle Enterprise Manager Fusion Middleware Control, Oracle WebLogic Administration Console, or WLST commands, see the chapter, "Managing Portlet Producers" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

This section includes the following subsections:

- Section 33.3.1, "How to Create Deployment Profiles"
- Section 33.3.2, "How to Create and Provision a WebLogic Managed Server Instance"
- Section 33.3.3, "How to Create and Register the Metadata Service Repository"
- Section 33.3.4, "How to Create a WebLogic Managed Server Connection"
- Section 33.3.5, "How to Deploy a Portlet Application to an Oracle WebLogic Managed Server Instance"
- Section 33.3.6, "What Happens When You Deploy a Portlet Application to an Oracle WebLogic-Managed Portlet Server"

33.3.1 How to Create Deployment Profiles

The project-level deployment profile is packaged as a Web Application Archive (WAR) file. Deployment descriptors are server configuration files that define the configuration of an application for deployment and are deployed with the portlet application as needed. The deployment descriptors that a project requires depend on the technologies the project uses and on the type of the target application server.

Note: You can deploy PDK-Java portlets only as EAR files. Therefore, while creating deployment profiles, ensure that the WAR file is included in the application's EAR file. For information about how to create an EAR file, see Section 25.3.2, "How to Create Deployment Profiles" in Chapter 25, "Testing and Deploying Your WebCenter Application".

This section includes the following subsections:

- Section 33.3.1.1, "Creating a WAR Deployment Profile"
- Section 33.3.1.2, "Creating a Deployment Descriptor"

33.3.1.1 Creating a WAR Deployment Profile

To create a WAR deployment profile:

1. In the Application Navigator, expand **Web Content** and **WEB-INF**.
2. Right-click **web.xml** and choose **Create WAR Deployment Profile**.

   Alternatively, from the main menu, choose **File** and then **New**. In the New Gallery, expand **General**, select **Deployment Profiles** and then **WAR File**, and click **OK**.
3. In the Create Deployment Profile -- WAR File dialog, enter a name for your deployment profile and click **OK**.
4. In the Edit WAR Deployment Profile Properties dialog, select the **Specify Java EE Web Context Root** option and enter a context root. You can also enter the name of the deployment profile. Then click **OK**.
5. In the Project Properties dialog, under **Deployment Profiles**, select the WAR File you just created and click **OK**.

33.3.1.2 Creating a Deployment Descriptor

To create a deployment descriptor:

1. From the main menu, choose **File** and then **New**.
2. In the New Gallery, expand **General**, select **Deployment Descriptors** and then **Java EE Deployment Descriptor**, and click **OK**.
3. In the Create Java EE Deployment Descriptor dialog, on steps 1 to 4, accept the default values and click **Finish**.

33.3.2 How to Create and Provision a WebLogic Managed Server Instance

To deploy a portlet producer application to a WebLogic Managed Server instance, you must first create a server instance and provision it with a required set of shared libraries. For more information, see the section "Creating and Provisioning a WebLogic Managed Server Instance" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

33.3.3 How to Create and Register the Metadata Service Repository

After creating the WebLogic Managed Server instance, you must create and register a Metadata Service Repository (MDS) schema for the application on the WebLogic

Domain's Administration Server instance. For more information, see the section "Creating and Registering the Metadata Service (MDS) Repository" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

33.3.4 How to Create a WebLogic Managed Server Connection

In JDeveloper, you can deploy your portlet applications to Oracle WebLogic Managed Server instances that reside outside JDeveloper. To do this, you must first create a connection to the server instance.

Before You Begin:

Before you create a connection to the managed server, ensure that the server instance is up and running.

To create a WebLogic Managed Server connection:

1. From the main menu, select **File** and then **New**.
2. In the New Gallery, expand **General**, select **Connections** and then **Application Server Connection**, and click **OK**.
3. In the Create Application Server Connection wizard, on Step 1, enter a name for the new connection, for example, `PortletServer`. Then click **Next**.
4. On Step 2, specify the user name and password for authentication and click **Next**.
5. On Step 3, enter the host name of the WebLogic Server instance, for example, `webcenter.portletserver.oracle.com` and the port number, for example, `7001`.
6. In the WLS Domain, specify the name of the domain in which the WebLogic Server instance is created, for example, `wc_domain`. Click **Next**.
7. On Step 4, click **Test Connection**. If the test is successful, the connection is set up.
8. Click **Finish**.

33.3.5 How to Deploy a Portlet Application to an Oracle WebLogic Managed Server Instance

After you have created the deployment profiles and a connection to the managed server for portlet deployment, you can deploy your portlet application to this server instance.

To deploy a portlet application:

1. In the Application Navigator, open the application to be deployed.
2. Right-click the project folder, choose **Deploy**, *deployment profile name*, **to**, and then choose the *connection name*.

   Note: If your portlet application contains JSR 168 portlets, then the Select deployment type dialog displays. Click **OK** to add the configuration required to expose this application as a WSRP service.

3. In the Select Deployment Target dialog, select the managed server name, for example, `WLS_Portlet`, and click **OK**.

4. In the Deployment Configuration dialog, under the **MDS** tab, in the **Repository Name** dropdown list, select the metadata repository to be used by the application you are deploying.
5. In the **Partition Name** field, specify the application's partition name. Each application is recommended to have its own partition. The application name can be used as the partition name.
6. Under the **Connections** tab, modify the connections packaged with the portlet producer application being deployed.

   Note: Consider the following about producer connections:

   - In the **Connections** tab, you cannot change secure properties of the connections. You can modify these connections post-deployment using Fusion Middleware Control. For information, see the chapter "Managing Services" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.
   - To edit a WSRP connection, you must also edit the associated Web service connection, which follows the naming convention `*connectionname*-wsconn`, for example, `myWSRPproducer-wsconn`.
   - To edit PDK-Java producers, you must also edit the underlying URL connection, which follows the naming convention `*connectionname*-urlconn`, for example, `myPDKproducer-urlconn`.
   - WSRP 1.0 standard does not support export and import capabilities. Therefore, customizations made to WSRP 1.0 producers are not migrated during deployment.

7. Click **Deploy**. The Deployment - Log displays the deployment status.

   The message **Deployment started** displays in the Deployment - Log window. If the application is successfully deployed to the targeted server instance, the message **Deployment finished** displays in the log.

33.3.6 What Happens When You Deploy a Portlet Application to an Oracle WebLogic-Managed Portlet Server

If you are deploying JSR 168 portlets, the configuration settings described in Section 33.2.2, "What Happens When You Test JSR 168 Portlets on Integrated WebLogic Server" are added to the EAR file.

If you are deploying PDK-Java portlets, the configuration settings described in Section 33.2.4, "What Happens When You Test PDK-Java Portlet Applications on Integrated WebLogic Server" are added to the application EAR file at design time.

33.4 Registering and Viewing Your Portlet

After you have created and deployed the producer and its portlets, you should register the producer with an application and add one or more portlets to a page to check that it is working correctly. Registering a producer gives applications the information they require to locate and communicate with that producer. After you register a producer, it is exposed as a connection, and the producer and its portlets become available in the Application Resources panel under the Connections node, or in the Resource Palette.

To register producers for your JSR 168 portlets, follow the instructions provided in Section 9.2.1, "How to Register a WSRP Portlet Producer".

To register producers for your PDK-Java portlets, follow the instructions provided in Section 9.2.3, "How to Register an Oracle PDK-Java Portlet Producer".

To add your portlets to a page, follow the instructions provided in Section 9.3, "Adding Portlets to a Page".

Part VI

Appendixes

Part VI contains the following appendixes:

A Files for WebCenter Applications

This appendix provides reference information about the files that are created and modified as you build up your WebCenter application. This appendix includes the following sections:

For a complete reference for the Oracle Application Development Framework (Oracle ADF) metadata files that you create in your data model and user interface projects, see the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

To learn more how files are affected by major actions, see Section 26.2.2, "Developer Actions Affecting Metadata Files."

A.1 About Files

When you use Oracle WebCenter Framework to build applications and components, a number of files are created as you perform such actions as building and consuming portlets. As you work with your application, you will find it useful to know a little bit about each of these files and how they relate to your application. You can group the files affected by the Oracle WebCenter Framework into two broad categories as follows:

- Files that are common to any Oracle ADF application, such as `web.xml`. For these files, it is useful to know what specific additions and changes are made for WebCenter applications. Those modifications are described in this appendix, but for more complete descriptions of these common files, see the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.
- Files that are unique to WebCenter applications, such as `portlet.xml`. For these files, it is useful to know what the file is for and what it contains. These files are described completely in this appendix.

A.2 Files Overview

The files that are created and modified are closely associated with the objects that you create as part of your WebCenter application. Hence, the easiest way to discuss these files is by object as follows:

- Files Related to JSR 168 Portlets
- Files Related to PDK-Java Portlets
- Files Related to Pages

A.3 Files Related to JSR 168 Portlets

When you build a JSR 168 portlet, the following files are created for you:

- portlet.xml
- oracle-portlet.xml
- oracle-portlet-tags.jar
- portlet_mode.jsp
- portlet_name.java
- portlet_nameBundle.jar
- web.xml
- connections.xml

A.3.1 portlet.xml

`portlet.xml` defines the characteristics of your JSR 168 portlet. For complete details on `portlet.xml`, you should see the Java Portlet Specification available at:

`http://jcp.org/en/jsr/detail?id=168`

Example A–1 provides a sample fragment from a `portlet.xml` file. Note that this example does not include all of the available elements of `portlet.xml`.

Example A–1 portlet.xml Sample

```
<portlet>
   <description xml:lang="en">JSR 168 map portlet </description>
   <portlet-name>portlet1</portlet-name>
   <display-name xml:lang="en">Map Portlet</display-name>
   <portlet-class>jsrportlet.MapPortlet</portlet-class>
   <expiration-cache>0</expiration-cache>
   <supports>
      <mime-type>text/html</mime-type>
      <portlet-mode>edit</portlet-mode>
   </supports>
   <supported-locale>en</supported-locale>
   <resource-bundle>jsr.resource.MapPortletBundle</resource-bundle>
   <portlet-preferences>
      <preference>
         <name>portletTitle</name>
      </preference>
   </portlet-preferences>
   <security-role-ref>
      <role-name>viewer</role-name>
   </security-role-ref>
</portlet>
```

For JSR 168 portlets, the `portlet.xml` file contains all information related to portlets and their settings. Note that not all of these settings are used in the previous sample.

- `<description>` provides a description of the portlet, which can be used to provide details to the end user.
- `<portlet-name>` uniquely identifies the portlet within the portlet application.
- `<display-name>` is used when presenting a list of available portlets to the user.
- `<portlet-class>` contains the fully qualified class name of the class implementing the `javax.portlet.Portlet` interface or extending the `GenericPortlet` abstract class that becomes the entry point for the portlet logic. The portlet container uses this class when it invokes the portlet life cycle methods.
- `<supports>` provides information about the portlet modes supported for each content type.
- `<title>` is the static title of the portlet, usually displayed in the portlet decoration on the portlet window.
- `<short-title>` is the title that is used on devices (such as mobile phones) that have limited display capabilities.
- `<keywords>` are used by applications that offer search capabilities for their users.
- `<security-role-ref>` maps a role name to a security role in `web.xml`. The list of roles in `web.xml` that the `<security-role-ref>` maps to is published to the consumer as the producer's user categories. In `web.xml`, `<security-role>` appears similar to the following:

  ```
  <security-role>
     <description>Viewer role</description>
     <role-name>viewer</role-name>
  </security-role>
  ```

A.3.2 oracle-portlet.xml

`oracle-portlet.xml` is an Oracle extension of `portlet.xml` to support WSRP 2.0 features. For example, a portlet's navigational parameters (a WSRP 2.0 feature) are defined in `oracle-portlet.xml`. In the future, when JSR 286 becomes available, this extension file will no longer be necessary.

The `oracle-portlet.xml` file is generated when:

- The **Enable inter-portlet communication using Oracle WSRP V2 extensions** checkbox is selected in the step 1, General Portlet Information of the Create JSR 168 Java Portlet wizard. For information, see Section 29.2.1, "How to Create a JSR 168 Java Portlet."
- A page or a task flow is converted into a JSF portlet using the Oracle JSF Portlet Bridge. For information, see Chapter 28, "Creating Portlets with the Oracle JSF Portlet Bridge."

A.3.2.1 oracle-portlet.xml Syntax

The top-level element of `oracle-portlet.xml` is `<portlet-app-extension>`.

```
<portlet-app-extension
 xsi:schemaLocation="http://xmlns.oracle.com/portlet/oracle-portlet-app"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
 xmlns="http://xmlns.oracle.com/portlet/oracle-portlet-app">
```

The `<portlet-app-extension>` element must include the child elements shown in Example A–2.

Example A–2 oracle-portlet.xml Element Hierarchy

```
<portlet-app-extension>
  <allow-export> ... </allow-export>
  <allow-import> ... </allow-import>
 <portlet-extension>
   <portlet-name> ... </portlet-name>
     <navigation-parameters>
       <name> ... </name>
       <type> ... </type>
       <label> ... </label>
       <hint> ... </hint>
       <usage> ... </usage>
       <aliases> ... </aliases>
     </navigation-parameters>
    <portlet-id> ... </portlet-id>
    <allow-export> ... </allow-export>
    <allow-import> ... </allow-import>
    <strict-authentication>...</strict-authentication>
    <hide-portlet> ... </hide-portlet>
    <require-iframe> ... </require-iframe>
    <minimum-wsrp-version>2</minimum-wsrp-version>
 </portlet-extension>
</portlet-app-extension>
```

The child elements have the following usages:

- `<portlet-app-extension>` indicates the start and end of the portlet application definition.
- `<portlet-extension>` indicates the start and end of a portlet definition.
- `<portlet-name>` identifies the portlet to which the extensions that follow it apply.
- `<navigation parameters>` defines the parameters for the previously identified portlet.
- `<name>` is the name of the navigation parameter. This name must be unique within the portlet.
- `<type>` is the type of the navigation parameter. Currently the only supported type is string.
- `<label>` is the label for the navigation parameter that end users see on the customization and personalization pages of the portlet.
- `<hint>` is currently not used.
- `<usage>` is currently not used.
- `<aliases>` is currently not used.
- `<portlet-id>` is the unique identifier of the portlet.
- `<allow-export>` is a flag that indicates whether the portlet supports the export of its customization data. This value can be `true` or `false`.

- `<allow-import>` is a flag that indicates whether the portlet supports the import of its customization data. This value can be `true` or `false`.
- `<strict-authentication>` is a flag to be set at the producer level. This value can be `true` or `false`. If this flag is set to `true`, it indicates that the portlet container must expose the current user and role membership information supplied by the local server. If `false`, the user and role membership information suggested by the consumer is used instead.
- `<hide-portlet>` is a flag that indicates that the portlet must not appear under the producer connection in the Application Resources panel or Resource Palette.

  Only `<require-iframe>` and `<minimum-wsrp-version>` tags can appear after this tag. If any other tag is added after the <hide-portlet> tag, then deployment may fail.
- `<require-iframe>` is used to indicate to the Oracle ADF Faces portlet consumer that the portlet must be rendered in an `iFrame`.
- `<minimum-wsrp-version>` is the minimum WSRP version supported. The minimum version is 2.

A.3.2.2 oracle-portlet.xml Sample With Navigation Parameters

Example A–3 provides a sample of `oracle-portlet.xml` with two portlets with three navigational parameters each.

Example A–3 oracle-portlet.xml Sample With Navigational Parameters

```
<portlet-extension>
  <portlet-name>ParameterForm</portlet-name>
   <navigation-parameters>
     <name>ora_wsrp_navigparam_Parameter1</name>
     <type>xsi:string</type>
     <label xml:lang="en">First parameter</label>
     <hint xml:lang="en">First parameter set by portlet</hint>
     <usage/>
     <aliases/>
   </navigation-parameters>
   <navigation-parameters>
     <name>ora_wsrp_navigparam_Parameter2</name>
     <type>xsi:string</type>
     <label xml:lang="en">Second parameter</label>
     <hint xml:lang="en">Second parameter set by portlet</hint>
     <usage/>
     <aliases/>
   </navigation-parameters>
   <navigation-parameters>
     <name>ora_wsrp_navigparam_Parameter3</name>
     <type>xsi:string</type>
     <label xml:lang="en">Third parameter</label>
     <hint xml:lang="en">Third parameter set by portlet</hint>
     <usage/>
     <aliases/>
   </navigation-parameters>
     <portlet-id>4</portlet-id>
     <allow-export>true</allow-export>
     <allow-import>true</allow-import>
     <require-iframe>true</require-iframe>
     <minimum-wsrp-version>2</minimum-wsrp-version>
</portlet-extension>
```

```
<portlet-extension>
  <portlet-name>ReadOnlyParameterForm</portlet-name>
   <navigation-parameters>
    <name>ora_wsrp_navigparam_Parameter1</name>
    <type>xsi:string</type>
    <label xml:lang="en">First parameter</label>
    <hint xml:lang="en">First parameter set by portlet</hint>
    <usage/>
    <aliases/>
   </navigation-parameters>
   <navigation-parameters>
    <name>ora_wsrp_navigparam_Parameter2</name>
    <type>xsi:string</type>
    <label xml:lang="en">Second parameter</label>
    <hint xml:lang="en">Second parameter set by portlet</hint>
    <usage/>
    <aliases/>
   </navigation-parameters>
   <navigation-parameters>
    <name>ora_wsrp_navigparam_Parameter3</name>
    <type>xsi:string</type>
    <label xml:lang="en">Third parameter</label>
    <hint xml:lang="en">Third parameter set by portlet</hint>
    <usage/>
    <aliases/>
   </navigation-parameters>
   <portlet-id>5</portlet-id>
   <allow-export>true</allow-export>
   <allow-import>true</allow-import>
    <require-iframe>true</require-iframe>
    <minimum-wsrp-version>2</minimum-wsrp-version>
</portlet-extension>
```

A.3.3 oracle-portlet-tags.jar

`oracle-portlet-tags.jar` is the Oracle implementation of the JSP tag library defined by the Java Portlet Specification.

A.3.4 portlet_mode.jsp

Depending on the implementation style of the portlet mode that you choose to create for your portlet, a corresponding JSP file is created in your *`portlet_name`*`\html` directory to define that mode. For example, if you choose to have View and Edit modes for your portlet, then you will need `view.jsp` and `edit.jsp` in your *`portlet_name`*`\html` directory. For JSR 168 portlets, you can have the following JSP files for your portlet modes:

- `about.jsp`
- `config.jsp`
- `edit_defaults.jsp`
- `edit.jsp`
- `help.jsp`
- `preview.jsp`
- `print.jsp`

- `view.jsp`

For further explanation of portlet modes, see Section 29.2.5.1, "Guidelines for Portlet Modes."

A.3.5 portlet_name.java

`portlet_name.java` is the class that acts as the entry point for the portlet logic. This class must implement the `javax.portlet.Portlet` interface or extend the `GenericPortlet` abstract class. The portlet container uses this class when it invokes the portlet lifecycle methods.

A.3.6 portlet_nameBundle.jar

`portlet_nameBundle.jar` is a resource bundle class, containing translation of the strings used by the portlet.

A.3.7 web.xml

`web.xml` is a Java EE standard descriptor that contains details about Web applications. For more information about `web.xml`, see the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

A.3.8 connections.xml

`connections.xml` contains WSRP producer connection information. Example A–4 shows a sample `connections.xml` file.

Example A–4 connections.xml

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<References xmlns="http://xmlns.oracle.com/adf/jndi">
  <Reference name="SampleWSRPProducer-wsconn"

className="oracle.adf.model.connection.webservice.impl.WebServiceConnectionImpl"
xmlns="">
     <Factory
className="oracle.adf.model.connection.webservice.api.WebServiceConnectionFactory"
/>
     <RefAddresses>
        <XmlRefAddr addrType="WebServiceConnection">
           <Contents>
              <wsconnection
description="http://portlet.uk.oracle.com:9999/portletapp/portlets/wsrp2?WSDL">
                 <model name="{urn:oasis:names:tc:wsrp:v2:wsdl}WSRP_v2_Service"
xmlns="http://oracle.com/ws/model">
                    <service name="{urn:oasis:names:tc:wsrp:v2:wsdl}WSRP_v2_
Service">
                       <port name="WSRP_v2_PortletManagement_Service"
binding="{urn:oasis:names:tc:wsrp:v2:bind}WSRP_v2_PortletManagement_Binding_SOAP">
                          <soap
addressUrl="http://hostname:port/portletapp/portlets/WSRP_v2_PortletManagement_
Service"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                          <operation name="setPortletProperties">
                             <soap
soapAction="urn:oasis:names:tc:wsrp:v2:setPortletProperties"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                             <input name="setPortletProperties"/>
                             <output name="setPortletPropertiesResponse"/>
                          </operation>
                          <operation name="getPortletProperties">
                             <soap
soapAction="urn:oasis:names:tc:wsrp:v2:getPortletProperties"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                             <input name="getPortletProperties"/>
                             <output name="getPortletPropertiesResponse"/>
                          </operation>
                          <operation name="importPortlets">
                             <soap
soapAction="urn:oasis:names:tc:wsrp:v2:importPortlet"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                             <output name="importPortletsResponse"/>
                             <input name="importPortlets"/>
                          </operation>
                          <operation name="destroyPortlets">
                             <soap
soapAction="urn:oasis:names:tc:wsrp:v2:destroyPortlets"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                             <input name="destroyPortlets"/>
                             <output name="destroyPortletsResponse"/>
                          </operation>
                          <operation name="exportPortlets">
                             <soap
soapAction="urn:oasis:names:tc:wsrp:v2:exportPortlet"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                             <output name="exportPortletsResponse"/>
                             <input name="exportPortlets"/>
                          </operation>
                          <operation name="releaseExport">
                             <soap
soapAction="urn:oasis:names:tc:wsrp:v2:importPortlet"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                             <input name="releaseExport"/>
                             <output name="releaseExportResponse"/>
                          </operation>
                          <operation name="getPortletsLifetime">
                             <soap
soapAction="urn:oasis:names:tc:wsrp:v2:getPortletsLifetime"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                             <output name="getPortletsLifetimeResponse"/>
                             <input name="getPortletsLifetime"/>
                          </operation>
                          <operation name="copyPortlets">
                             <soap
soapAction="urn:oasis:names:tc:wsrp:v2:copyPortlets"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                             <output name="copyPortletsResponse"/>
                             <input name="copyPortlets"/>
                          </operation>
                          <operation name="getPortletPropertyDescription">
                             <soap
soapAction="urn:oasis:names:tc:wsrp:v2:getPortletPropertyDescription"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                             <input name="getPortletPropertyDescription"/>
                             <output
name="getPortletPropertyDescriptionResponse"/>
                          </operation>
                          <operation name="clonePortlet">
```

```
                            <soap
soapAction="urn:oasis:names:tc:wsrp:v2:clonePortlet"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                            <output name="clonePortletResponse"/>
                            <input name="clonePortlet"/>
                         </operation>
                         <operation name="setPortletsLifetime">
                            <soap
soapAction="urn:oasis:names:tc:wsrp:v2:setPortletsLifetime"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                            <output name="setPortletsLifetimeResponse"/>
                            <input name="setPortletsLifetime"/>
                         </operation>
                         <operation name="getPortletDescription">
                            <soap
soapAction="urn:oasis:names:tc:wsrp:v2:getPortletDescription"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                            <input name="getPortletDescription"/>
                            <output name="getPortletDescriptionResponse"/>
                         </operation>
                         <operation name="setExportLifetime">
                            <soap
soapAction="urn:oasis:names:tc:wsrp:v2:setExportLifetime"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                            <input name="setExportLifetime"/>
                            <output name="setExportLifetimeResponse"/>
                         </operation>
                      </port>
                      <port name="WSRP_v2_Markup_Service"
binding="{urn:oasis:names:tc:wsrp:v2:bind}WSRP_v2_Markup_Binding_SOAP">
                         <soap
addressUrl="http://hostname:port/portletapp/portlets/WSRP_v2_Markup_Service"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                         <operation name="initCookie">
                            <soap
soapAction="urn:oasis:names:tc:wsrp:v2:initCookie"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                            <output name="initCookieResponse"/>
                            <input name="initCookie"/>
                         </operation>
                         <operation name="getMarkup">
                            <soap
soapAction="urn:oasis:names:tc:wsrp:v2:getMarkup"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                            <output name="getMarkupResponse"/>
                            <input name="getMarkup"/>
                         </operation>
                         <operation name="releaseSessions">
                            <soap
soapAction="urn:oasis:names:tc:wsrp:v2:releaseSessions"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                            <input name="releaseSessions"/>
                            <output name="releaseSessionsResponse"/>
                         </operation>
                         <operation name="performBlockingInteraction">
                            <soap
soapAction="urn:oasis:names:tc:wsrp:v2:performBlockingInteraction"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                            <input name="performBlockingInteraction"/>
                            <output name="performBlockingInteractionResponse"/>
```

```
                        </operation>
                        <operation name="getResource">
                           <soap
soapAction="urn:oasis:names:tc:wsrp:v2:getResource"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                           <output name="getResourceResponse"/>
                           <input name="getResource"/>
                        </operation>
                        <operation name="handleEvents">
                           <soap
soapAction="urn:oasis:names:tc:wsrp:v2:handleEvents"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                           <input name="handleEvents"/>
                           <output name="handleEventsResponse"/>
                        </operation>
                     </port>
                     <port name="WSRP_v2_Registration_Service"
binding="{urn:oasis:names:tc:wsrp:v2:bind}WSRP_v2_Registration_Binding_SOAP">
                        <soap
addressUrl="http://hostname:port/portletapp/portlets/WSRP_v2_Registration_Service"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                        <operation name="register">
                           <soap
soapAction="urn:oasis:names:tc:wsrp:v2:register"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                           <input name="register"/>
                           <output name="registerResponse"/>
                        </operation>
                        <operation name="getRegistrationLifetime">
                           <soap
soapAction="urn:oasis:names:tc:wsrp:v2:getRegistrationLifetime"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                           <output name="getRegistrationLifetimeResponse"/>
                           <input name="getRegistrationLifetime"/>
                        </operation>
                        <operation name="setRegistrationLifetime">
                           <soap
soapAction="urn:oasis:names:tc:wsrp:v2:setRegistrationLifetime"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                           <input name="setRegistrationLifetime"/>
                           <output name="setRegistrationLifetimeResponse"/>
                        </operation>
                        <operation name="deregister">
                           <soap
soapAction="urn:oasis:names:tc:wsrp:v2:deregister"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                           <input name="deregister"/>
                           <output name="deregisterResponse"/>
                        </operation>
                        <operation name="modifyRegistration">
                           <soap
soapAction="urn:oasis:names:tc:wsrp:v2:modifyRegistration"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                           <input name="modifyRegistration"/>
                           <output name="modifyRegistrationResponse"/>
                        </operation>
                     </port>
                     <port name="WSRP_v2_ServiceDescription_Service"
binding="{urn:oasis:names:tc:wsrp:v2:bind}WSRP_v2_ServiceDescription_Binding_
SOAP">
```

```
                           <soap
addressUrl="http://hostname:port/portletapp/portlets/WSRP_v2_ServiceDescription_
Service"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                           <operation name="getServiceDescription">
                              <soap
soapAction="urn:oasis:names:tc:wsrp:v2:getServiceDescription"
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"/>
                              <input name="getServiceDescription"/>
                              <output name="getServiceDescriptionResponse"/>
                           </operation>
                        </port>
                     </service>
                  </model>
               </wsconnection>
            </Contents>
         </XmlRefAddr>
      </RefAddresses>
  </Reference>
  <Reference name="SampleWSRPProducer"
className="oracle.portlet.client.connection.wsrp.WSRPProducerConnection" xmlns="">
     <Factory
className="oracle.portlet.client.connection.wsrp.WSRPProducerConnectionFactory"/>
     <RefAddresses>
        <XmlRefAddr addrType="connectionDesc">
           <Contents>
              <wsrpproducerconnection wsConnection="SampleWSRPProducer-wsconn"
timeout="30"/>
           </Contents>
        </XmlRefAddr>
     </RefAddresses>
  </Reference>
</References>
```

A.4 Files Related to PDK-Java Portlets

When you build a PDK-Java portlet, the following files are created for you:

- _default.properties
- index.jsp
- portlet_name_modePage.jsp
- producer_name.properties
- provider.xml
- web.xml
- connections.xml

A.4.1 producer_name.properties

`producer_name.properties` specifies deployment details about the producer, such as the location of the `provider.xml` file. For example, this file is used if the registration URL to the PDK-Java samples is of the form:

```
http://host:port/jpdk/provider/samples
```

or:

```
http://host:port/jpdk/provider
```

where the service ID field contains `samples`. See also Section A.4.2, "_default.properties".

A.4.2 _default.properties

`_default.properties` specifies deployment details about the producer, such as the location of the `provider.xml` file. For example, this file is used if the registration URL to the PDK-Java samples is of the form:

```
http://host:port/jpdk/provider
```

Note that the producer name is not supplied here so it has to default. See also Section A.4.1, "producer_name.properties".

A.4.3 index.jsp

`index.jsp` serves as a convenient starting point when testing PDK-Java producers from Oracle JDeveloper. This file lists all of the producers available in the application.

A.4.4 portlet_name_modePage.jsp

Depending on the implementation style of the portlet mode that you choose to create for your portlet, a corresponding JSP file is created in your `\htdocs\`*`portlet_name`* directory to define that mode. For example, if you choose to have View and Edit modes for a portlet named `portletOne`, then you need `portletOneShowPage.jsp` and `PortletOneEditPage.jsp` in your `\htdocs\portletOne` directory. For PDK-Java portlets, you can have the following JSP files for your portlet modes:

- `portlet_nameAboutPage.jsp`
- `portlet_nameEditDefaultsPage.jsp`
- `portlet_nameEditPage.jsp`
- `portlet_nameHelpPage.jsp`
- `portlet_nameShowDetailsPage.jsp`
- `portlet_nameShowPage.jsp`

For further explanation of portlet modes, see Section 29.2.5.1, "Guidelines for Portlet Modes".

A.4.5 provider.xml

`provider.xml` is the definition file for your PDK-Java producer.

Note: PDK-Java producers were formerly referred to as providers.

A.4.5.1 provider.xml Syntax

For more information about the elements and syntax of `provider.xml`, see the Provider Definition Extensible Markup Language (XML) Tag Reference v2 on the Oracle Technology Network.

`http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html`

A.4.5.2 provider.xml Sample

Example A–5 provides a sample `provider.xml` file.

Example A–5 provider.xml Sample

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<?providerDefinition version="3.1"?>
<provider class="oracle.portal.provider.v2.DefaultProviderDefinition">
 <localePersonalizationLevel>none</localePersonalizationLevel>
 <session>true</session>
 <defaultLocale>en</defaultLocale>
 <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>SampleRenderer</name>
  <title>SampleRenderer example</title>
  <shortTitle>SampleRenderer</shortTitle>
  <description>Example portlet rendered using the SampleRenderer</description>
  <timeout>40</timeout>
  <timeoutMessage>SampleRenderer example timed out</timeoutMessage>
  <acceptContentType>text/html</acceptContentType>
  <showEdit>true</showEdit>
  <showEditToPublic>false</showEditToPublic>
  <showEditDefault>true</showEditDefault>
  <showPreview>true</showPreview>
  <showDetails>true</showDetails>
  <hasHelp>true</hasHelp>
  <hasAbout>true</hasAbout>
  <renderer
   class="oracle.portal.sample.v2.devguide.samplerenderer.SampleRenderer"/>
   <personalizationManager
   class="oracle.portal.provider.v2.personalize.PrefStorePersonalizationManager">
   <dataClass>oracle.portal.provider.v2.personalize.
     NameValuePersonalizationObject
   </dataClass>
   </personalizationManager>
 </portlet>
</provider>
```

A.4.6 web.xml

`web.xml` is a Java EE standard descriptor that contains details about Web applications. For more information about `web.xml`, see the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

A.4.7 connections.xml

`connections.xml` contains PDK-Java producer connection information. Example A–4 shows a sample `connections.xml` file.

A.5 Files Related to Pages

When you create or modify pages, the following files are created or modified:

- Created at page design time:

  - adf-config.xml
  - DataBindings.cpx
  - faces-config.xml
  - page_name.jspx
  - PageDef.xml
  - web.xml
- Created at application deployment time:
  - mds Subdirectory
- Created upon consumption of JSR 168 portlets
  - wsdl Subdirectory

A.5.1 adf-config.xml

The `adf-config.xml` file is an extensible configuration file used by any ADF component. This file is used by the various WebCenter Services, including pages, portlets, and all the Web 2.0 services such as search, worklist, forums, and so on, to set configuration information. This file, in conjunction with `connections.xml`, contains the configuration information required for most of the services to function.

Table A–1 describes the child elements of `<adf-portlet-config>`. The `init-param` names in the first column correspond to the names of the servlet `init-params` used when the portlet client is accessed through the Web adapter.

Table A–1 Child Elements of adf-portlet-config

Element (init-param)	Description	Default Value
`parallelPoolSize` `(parallel.pool.size)`	The number of threads to use for parallel execution of tasks.	10
`parallelQueueSize` `(parallel.queue.size)`	The size of the queue of tasks waiting for parallel execution. Tasks are rejected once the queue size is exceeded.	20
`defaultTimeout` `(default.timeout)`	The default timeout period in seconds for requests made to producers. This value is used when a timeout is not defined at the portlet or producer level.	10
`minimumTimeout` `(minimum.timeout)`	The minimum timeout period in seconds for requests made to producers. This value is used to impose a lower limit on timeout periods specified by portlets or producers.	0.1
`maximumTimeout` `(maximum.timeout)`	The maximum timeout period in seconds for requests made to producers. This value is used to impose an upper limit on timeout periods specified by portlets or producers.	60

Table A–1 (Cont.) Child Elements of adf-portlet-config

Element (init-param)	Description	Default Value
`resourceProxyPath` (`resource.proxy.path`)	The base path of the resource proxy servlet, relative to the context root of the application. Used to construct links to the resource servlet within portlet markup.	/resourceproxy
`supportedLocales` (`supported.locales`)	The set of supported locales defined using strings of the form: `language[_country[_variant]]`	Commented out by default. You should uncomment it if you have multiple locales. See Example A–6.
`portletTechnologies` (`portlet.technologies`)	The set of portlet technologies supported by the client defined by the fully qualified names of classes that implement the PortletTechnologyConfig interface.	{o.p.c.ci.web.WebPortletTechnologyConfig, o.p.c.ci.wsrp.WSRPPortletTechnologyConfig}
`cacheSettings` (`cache.*`)	Cache configuration information. Used to enable or disable the cache, define its maximum size and impose limits on the amount of space available for different users and subscribers.	The cache is enabled and no size restrictions are imposed.

<adf-portlet-config> element Sample

Example A–6 illustrates the usage of the `<adf-portlet-config>` element.

Example A–6 <adf-portlet-config> element Sample

```
<adf-portlet-config xmlns="http://xmlns.oracle.com/adf/portlet/config">
  <supportedLocales>
    <value>en</value>
    <value>fr</value>
    <value>de</value>
    <value>es</value>
  </supportedLocales>
  <portletTechnologies>
   <value>oracle.portlet.client.containerimpl.web.
     WebPortletTechnologyConfig</value>
   <value>oracle.portlet.client.containerimpl.wsrp.
     WSRPPortletTechnologyConfig</value>
  </portletTechnologies>
  <defaultTimeout>20</defaultTimeout>
  <minimumTimeout>1</minimumTimeout>
  <maximumTimeout>60</maximumTimeout>
  <resourceProxyPath>/portletresource</resourceProxyPath>
  <cacheSettings>
    <maxSize>10000000</maxSize>
    <subscriber default="true">
      <systemLevel>
        <maxSize>5000000</maxSize>
      </systemLevel>
      <userLevel>
        <maxSize>8000000</maxSize>
      </userLevel>
    </subscriber>
  </cacheSettings>
```

```
</adf-portlet-config>
```

A.5.2 DataBindings.cpx

`DataBindings.cpx` is a file common to Web applications. For more information about `DataBindings.cpx`, see the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

A.5.3 faces-config.xml

`faces-config.xml` is a file common to JSF applications. It describes the page flow of your application. For more information about `faces-config.xml`, see the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

A.5.4 page_name.jspx

`page_name``.jspx` is the JSP file for your page. Whenever you add or remove components, such as portlets or data controls from the page, this file is updated.

A.5.5 PageDef.xml

`PageDef.xml` is a file common to Oracle ADF applications. This file holds information about portlet bindings. Also, portlet parameters can be tied to page variables in this file.

For more information about `PageDef.xml`, see the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

PageDef.xml Sample

Example A–7 provides a sample `PageDef.xml` file.

Example A–7 PageDef.xml Sample

```
<?xml version="1.0" encoding="UTF-8" ?>
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
   version="10.1.3.37.97" id="app_SRFeedbackPageDef"
   Package="oracle.srdemo.view.pageDefs">
  <parameters/>
  <executables>
    <methodIterator id="findAllServiceRequestIter"
                    Binds="findAllServiceRequest.result"
                    DataControl="SRPublicFacade" RangeSize="4"
                    BeanClass="oracle.srdemo.model.entities.ServiceRequest"/>
    <variableIterator id="variables">
      <variable Name="portlet1_Param1" Type="java.lang.Object"
      DefaultValue="
      ${bindings.findAllServiceRequestIter.currentRow.dataProvider['svrId']}"/>
      <variable Name="portlet1_Param2" Type="java.lang.Object"/>
      <variable Name="portlet1_Param3" Type="java.lang.Object"/>
      <variable Name="portlet1_Param4" Type="java.lang.Object"/>
      <variable Name="portlet1_Param5" Type="java.lang.Object"/>
    </variableIterator>
    <methodIterator id="findServiceRequestByIdIter"
                    Binds="findServiceRequestById.result"
                    DataControl="SRPublicFacade" RangeSize="10"
                    BeanClass="oracle.srdemo.model.entities.ServiceRequest"/>
    <portlet id="portlet1"
      portletInstance="/oracle/adf/portlet/OmniProducer_1150310748178/
      applicationPortlets/Portlet100_eebc7f18_010b_1000_8001_82235f640cea"
      class="oracle.adf.model.portlet.binding.PortletBinding"
        xmlns="http://xmlns.oracle.com/portlet/bindings">
      <parameters>
        <parameter name="Param1" pageVariable="portlet1_Param1"/>
        <parameter name="Param2" pageVariable="portlet1_Param2"/>
        <parameter name="Param3" pageVariable="portlet1_Param3"/>
        <parameter name="Param4" pageVariable="portlet1_Param4"/>
        <parameter name="Param5" pageVariable="portlet1_Param5"/>
      </parameters>
    </portlet>
  </executables>
  <bindings>
    <methodAction id="findAllServiceRequest"
                  InstanceName="SRPublicFacade.dataProvider"
                  DataControl="SRPublicFacade"
                  MethodName="findAllServiceRequest" RequiresUpdateModel="true"
                  Action="999" IsViewObjectMethod="false"
                  ReturnName="SRPublicFacade.methodResults.
                     SRPublicFacade_dataProvider_findAllServiceRequest_result"/>
    <table id="findAllServiceRequest1" IterBinding="findAllServiceRequestIter">
      <AttrNames>
        <Item Value="assignedDate"/>
        <Item Value="problemDescription"/>
        <Item Value="requestDate"/>
        <Item Value="status"/>
        <Item Value="svrId"/>
        <Item Value="custComment"/>
        <Item Value="custCommentDate"/>
        <Item Value="custCommentContactBy"/>
        <Item Value="mgrNotes"/>
        <Item Value="mgrNotesDate"/>
      </AttrNames>
    </table>
    <methodAction id="findServiceRequestById"
                  InstanceName="SRPublicFacade.dataProvider"
                  DataControl="SRPublicFacade"
                  MethodName="findServiceRequestById" RequiresUpdateModel="true"
                  Action="999" IsViewObjectMethod="false"
                  ReturnName="SRPublicFacade.methodResults.
                     SRPublicFacade_dataProvider_findServiceRequestById_result">
      <NamedData NDName="svrIdParam"
         NDValue=
          "${bindings.findAllServiceRequestIter.currentRow.dataProvider['svrId']}"
         NDType="java.lang.Integer"/>
    </methodAction>
    <attributeValues id="svrId" IterBinding="findServiceRequestByIdIter">
      <AttrNames>
        <Item Value="svrId"/>
      </AttrNames>
    </attributeValues>
    <attributeValues id="custComment" IterBinding="findServiceRequestByIdIter">
      <AttrNames>
        <Item Value="custComment"/>
      </AttrNames>
    </attributeValues>
    <attributeValues id="mgrNotes" IterBinding="findServiceRequestByIdIter">
      <AttrNames>
        <Item Value="mgrNotes"/>
      </AttrNames>
    </attributeValues>
    <list id="ServiceRequestcustCommentContactBy"
          IterBinding="findServiceRequestByIdIter" ListOperMode="0"
          StaticList="true" NullValueFlag="1">
      <AttrNames>
        <Item Value="custCommentContactBy"/>
      </AttrNames>
      <ValueList>
        <Item Value=" "/>
        <Item Value="Phone"/>
        <Item Value="Email"/>
        <Item Value="SMS"/>
      </ValueList>
    </list>
    <methodAction id="mergeEntity" InstanceName="SRPublicFacade.dataProvider"
                  DataControl="SRPublicFacade" MethodName="mergeEntity"
                  RequiresUpdateModel="true" Action="999"
                  IsViewObjectMethod="false"
                  ReturnName="SRPublicFacade.methodResults.
                              SRPublicFacade_dataProvider_mergeEntity_result">
      <NamedData NDName="entity"
         NDValue=
         "${bindings.findServiceRequestByIdIter.currentRow.dataProvider}"
         NDType="java.lang.Object"/>
    </methodAction>
  </bindings>
</pageDefinition>
```

A.5.6 web.xml

`web.xml` is a Java EE standard descriptor that contains details about Web applications. For more information about `web.xml`, see the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

A.5.7 mds Subdirectory

When you deploy a WebCenter application, an `mds` subdirectory is created in your project directory. This subdirectory contains other subdirectories and metadata files, such as portlet customizations and personalizations.

A.5.8 wsdl Subdirectory

When you consume JSR 168 portlets in an application, a `wsdl` subdirectory is created in the `WEB-INF` directory. The files in this subdirectory are internal files created for portlets from a WSRP producer.

A.6 Files Related to Security

When you implement or modify security, the following files are created or modified:

- jazn-data.xml
- cwallet.sso
- jps-config.xml
- adf-config.xml

A.6.1 jazn-data.xml

When you implement or modify security for your application, the `jazn-data.xml` file is created or modified.

The `jazn-data.xml` file is used to facilitate the deployment of realm and policy information for your application. In your development environment (JDeveloper), `jazn-data.xml` is located in your application's *`workspacedir`*`/src/META-INF` directory. After deployment, the contents of this file are merged in domain-level identity or policy stores. This file contains the policies that are created at design time, and is merged at deployment by `JpsApplicationLifecycleListener` into the `system-jazn-data.xml` file (or appropriately configured policy store) for the deployment target.

When you migrate security information with `JpsApplicationLifecycleListener`, the `jazn-data.xml` file can be used as the source file for the migration. For more information about migrating roles, see the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

jazn-data.xml Sample

Example A–8 provides a sample `jazn-data.xml` file.

Example A–8 jazn-data.xml Sample

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<jazn-data>
   <jazn-realm>
      <realm>
         <name>jazn.com</name>
      </realm>
   </jazn-realm>
   <policy-store>
      <applications>
         <application>
            <name>mdsapp-secure</name>
            <app-roles>
               <app-role>
                  <name>unauthenticated-role</name>
                  <display-name>anonymous-role</display-name>

<class>oracle.security.jps.service.policystore.ApplicationRole</class>
               </app-role>
               <app-role>
                  <name>authenticated-role</name>
                  <display-name>authenticated-users</display-name>

<class>oracle.security.jps.service.policystore.ApplicationRole</class>
               </app-role>
               <app-role>
                  <name>approle</name>
                  <display-name>approle</display-name>
                  <description>approle</description>
                  <guid/>

<class>oracle.security.jps.service.policystore.ApplicationRole</class>
                  <members>
                     <member>

<class>oracle.security.jps.internal.core.principals.JpsXmlUserImpl</class>
```

```
                        <name>orcladmin</name>
                     </member>
                  </members>

               </app-role>
            </app-roles>
            <jazn-policy>
<grant>
<grantee>
 <principals>
     <principal>
<class>oracle.security.jps.internal.core.principals.JpsAuthenticatedRoleImpl</clas
s>
<name>authenticated-role</name>
     </principal>
 </principals>
</grantee>
<permissions>
 <permission>
<class>oracle.adf.share.security.authorization.RegionPermission</class>
<name>project1.pageDefs.securePageDef</name>
<actions>view</actions>
 </permission>
</permissions>
</grant>
<grant>
<grantee>
 <principals>
     <principal>
<class>oracle.security.jps.internal.core.principals.JpsAnonymousRoleImpl</class>
         <name>anonymous-role</name>
     </principal>
 </principals>
</grantee>
<permissions>
 <permission>
<class>oracle.adf.share.security.authorization.RegionPermission</class>
<name>project1.pageDefs.publicPageDef</name>
<actions>view</actions>
 </permission>
</permissions>
</grant>
            </jazn-policy>
         </application>
      </applications>
   </policy-store>
   <jazn-policy/>
</jazn-data>
```

A.6.2 cwallet.sso

`cwallet.sso` contains external application credentials, if any, and the connection's secure data such as database password. The `cwallet.sso` file is located in the *`workspaceroo`*`t/src/META-INF` directory.

A.6.3 jps-config.xml

`jps-config.xml` is located at *`workspaceroot`*`/src/META-INF`. The `jps-config.xml` file packaged with the application is used only at design time. At runtime, the configurations stored in the domain-level `jps-config.xml` file become functional.

`jps-config.xml` configures OPSS services. For more information on the element hierarchy and attributes of `jps-config.xml`, see the appendix "OPSS Configuration File Reference" in the *Oracle Fusion Middleware Security Guide*.

The recommendation for migration is to create a new `jps-config.xml` file with the source and destination contexts defined as required by the migration commands. For details on migrating identities, see the section"Migrating Identities Manually" in the *Oracle Fusion Middleware Security Guide*.

The policy store is migrated in the following ways: entire policy store, application-specific policies, and global polices. For details, see the section "Migrating Policies Manually" in the *Oracle Fusion Middleware Security Guide*.

The credential store is migrated in the following ways: entire credential store or application-specific credentials. When migrating application-specific credentials, ensure that `srcFolder` and `targetFolder` are the same. `srcFolder` and `targetFolder` can be determined from `appUID` in the `adf-config.xml` file, see Example A–9. For details, see the section "Migrating Credentials Manually" in the *Oracle Fusion Middleware Security Guide*.

Example A–9 appUID in the adf-config.xml file

```
<adf:adf-properties-child xmlns="http://xmlns.oracle.com/adf/config/properties">
  <adf-property name="adfAppUID" value="Application1-1167"/>
</adf:adf-properties-child>
```

A.6.4 adf-config.xml

Security configurations are stored in `adf-config.xml`. This file is located at *`workspaceroot`*`/.adf/META-INF/adf-config.xml`. See Example A–10.

Example A–10 adf-config.xml

```
<adf-security-child xmlns="http://xmlns.oracle.com/adf/security/config">
        <JaasSecurityContext
initialContextFactoryClass="oracle.adf.share.security.JAASInitialContextFactory"
jaasProviderClass="oracle.adf.share.security.providers.jps.JpsSecurityContext"
             authorizationEnforce="true"
             authenticationRequire="true"/>
    <CredentialStoreContext
credentialStoreClass="oracle.adf.share.security.providers.jps.CSFCredentialStore"
             credentialStoreLocation="../../src/META-INF/jps-config.xml"/>
</adf-security-child>
```

Note: The `crdentialStoreLocation` entry in the sample is valid at design time only.

A.7 Files Related to WebCenter Web 2.0 Services

This section includes sample `connections.xml` and `adf-config.xml` files for some of the WebCenter Web 2.0 Services.

Example A–11 shows the minimum required configuration for the SES Web service connection.

Example A–11 SES Web Service Sample Connection

```
<Reference name="GenericSesConnection"
className="oracle.adf.mbean.share.connection.webcenter.search.SesConnectionReferen
ceable"
credentialStoreKey="GenericSesConnection" xmlns="">
      <Factory
className="oracle.adf.mbean.share.connection.webcenter.search.SesConnectionFactory
"/>
      <RefAddresses>
         <XmlRefAddr addrType="SesConfig">
            <Contents>
               <SesConfig>
                  <AttributeMap>
                     <Attribute name="date">
                        <SesAttribute>LastModifiedDate</SesAttribute>
                     </Attribute>
                     <Attribute name="person">
                        <SesAttribute>Author</SesAttribute>
                     </Attribute>
                  </AttributeMap>

<SoapUrl>http://hostname:port/search/query/OracleSearch</SoapUrl>
               </SesConfig>
            </Contents>
         </XmlRefAddr>
         <StringRefAddr addrType="appUsername">
            <Contents>wpadmin</Contents>
         </StringRefAddr>
         <SecureRefAddr addrType="appPassword"/>
      </RefAddresses>
   </Reference>
```

Example A–12 shows a sample `adf-config.xml` for the Documents service.

Example A–12 ADF Configuration for the Documents Service

```
<doclibC:adf-doclib-config xmlns="http://xmlns.oracle.com/webcenter/doclib/config"
   primaryConnectionName="Oracle Content Server">
    <doclibC:spaces-repository
      spacesRoot="/repository path"
      adminUserName="sysadmin"
      applicationName="application_name" />
  </doclibC:adf-doclib-config>
```

Example A–13 shows a sample `adf-config.xml` for the Worklist service.

Example A–13 ADF Configuration for the Worklist Service

```
  <worklistC:adf-worklist-config>
    <worklistC:ConnectionName>WebCenter-Worklist</worklistC:ConnectionName>
  </worklistC:adf-worklist-config>
```

B Oracle Composer Component Properties and Files

This appendix provides reference information about Oracle Composer-specific tags, configuration files, and style properties. This information is useful while performing the tasks described in Chapter 4, "Enabling Runtime Editing of Pages Using Oracle Composer" and Chapter 5, "Extending Runtime Editing Capabilities Using Oracle Composer."

This appendix includes the following sections:

- Section B.1, "Oracle Composer Component Properties"
- Section B.2, "Oracle Composer-Specific Files and Configurations"
- Section B.3, "Oracle Composer Default Add-Ons and Property Panels"
- Section B.4, "Oracle Composer Help Topic IDs"
- Section B.5, "Oracle Composer Components Style-Specific Properties"
- Section B.6, "Customizable Components (HTML) Properties"

B.1 Oracle Composer Component Properties

This section is a quick reference for all Oracle Composer tags. When you add Oracle Composer components to the page, default values are assigned to certain attributes. You can change these default values and define values for the remaining attributes in JDeveloper.

For information about adding these components to the page, see Section 4.2, "Designing Editable Pages Using Oracle Composer Components."

Note: The Property Inspector in Oracle JDeveloper displays certain extended metadata attributes under the Customization Attributes and Annotations Attributes categories. You can use these properties to provide additional metadata information. For more information, see Extended Metadata and Annotation Properties in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

B.1.1 Page Customizable

Use the `Page Customizable` component to enable page editing at runtime. The `Page Customizable` component denotes the customizable part of a page and shows the Oracle Composer toolbar in Edit mode at runtime. Components enclosed within a `Page Customizable` can be customized and edited.

In large applications, where you want to enable runtime page editing on multiple pages, you can include the `Page Customizable` in the page template, somewhere in the top of the hierarchy. This way, all pages created using the template will be editable at runtime.

Geometry Management

The `Page Customizable` can be stretched by a parent layout component that stretches its children, for example, a `Panel Stretch Layout` or `Panel Splitter` component.

The `Page Customizable` component stretches its first child component to fill the space available to it. It is recommended that the `Page Customizable` contain only one child component.

Attributes

Table B–1 describes the attributes of a `Page Customizable` component.

Table B–1 Attributes of a Page Customizable Component

Attribute	Type	Supports EL	Description
Common Attributes			
id	String	No	The identifier for the component. The identifier must follow a subset of the syntax allowed in HTML: ▪ Must not be a zero-length String. First character must be an ASCII letter (A-Za-z) or an underscore ('_'). ▪ Subsequent characters must be an ASCII letter or digit (A-Za-z0-9), an underscore ('_'), or a dash ('-').
rendered	boolean	Yes	Specifies whether the component is rendered. When set to `false`, no output will be delivered for this component (the component will not in any way be rendered, and cannot be made visible on the client). The default value is `true`.
Style Attributes			
styleClass	String	Yes	The CSS style class to use for this component. The style class can be defined in your jspx page or in a skinning CSS file, for example.

Table B–1 (Cont.) Attributes of a Page Customizable Component

Attribute	Type	Supports EL	Description
inlineStyle	String	Yes	The CSS styles to use for this component. This is intended for basic style changes. The `inlineStyle` is a set of CSS styles that are applied to the root DOM element of the component. If the `inlineStyle`'s CSS properties do not affect the DOM element you want affected, then you will have to create a skin and use the skinning keys which are meant to target particular DOM elements, like `::label` or `::icon-style`.
Advanced Attributes			
binding	`oracle.adf.view.page.editor.component.PageCustomizable`	No	An EL reference that will store the component instance on a bean. This can be used to give programmatic access to a component from a backing bean, or to move creation of the component to a backing bean.

Supported Facets

The `Page Customizable` component provides an `editor` facet, which is prepopulated with a `Page Editor Panel` component. To ensure that Oracle Composer works properly, the `editor` facet must contain the `Page Editor Panel` component.

B.1.2 Change Mode Link and Change Mode Button

Use the `Change Mode Button` or `Change Mode Link` to enable switching to Edit mode of the page at runtime.

The `Change Mode Link` and `Change Mode Button` components share a common set of attributes. Table B–2 describes the attributes of a `Change Mode Link` or `Change Mode Button` component.

Table B–2 Attributes of a Change Mode Link or Change Mode Button Component

Attribute	Type	Supports EL?	Description
Common Attributes			
id	String	Yes	The identifier for the component. The identifier must follow a subset of the syntax allowed in HTML: ■ Must not be a zero-length String. First character must be an ASCII letter (A-Za-z) or an underscore ('_ '). ■ Subsequent characters must be an ASCII letter or digit (A-Za-z0-9), an underscore ('_'), or a dash ('-').
rendered	boolean	Yes	Specifies whether the component is rendered. When set to `false`, no output will be delivered for this component (the component will not in any way be rendered, and cannot be made visible on the client). The default value is `true`.
Behavior Attributes			
immediate	boolean	Yes	Specifies whether data validation - client-side or server-side - will be skipped when events are generated by this component. When `immediate` is `true`, the command's action and ActionListeners, including the default ActionListener provided by the JavaServer Faces implementation, will be executed during Apply Request Values phase of the request processing lifecycle, rather than waiting until the Invoke Application phase. Because validation runs during Process Validators (after Apply Request Values, but before Invoke Application), setting immediate to `true` will skip validation.
Advanced Attributes			
binding	`oracle.adf.view.page.editor.component.ChangeModeLink` or `oracle.adf.view.page.editor.component.ChangeModeButton`	Yes	An EL reference that will store the component instance on a bean. This can be used to give programmatic access to a component from a backing bean, or to move creation of the component to a backing bean.

If you do not want to use the `Change Mode Link` or `Change Mode Button` component, you can add your own button or link component and use the `ChangeModeLink` or `ChangeModeButton` API to enable switching of page mode.

Disabling Runtime Editing in Your Application Pages

To disable runtime editing of pages, you can revoke Edit privileges for users, or delete the `Change Mode Link` or `Change Mode Button` component from the page so that the user cannot navigate to Edit mode of the page.

B.1.3 Layout Customizable Component

The `Layout Customizable` component applies a predefined layout to its child components. It is a container component that enables end users to lay out its child components in several predefined ways. Use the `Layout Customizable` component only to enable changing of the page layout at runtime. If you just want container components in which to layout components, but do not want to enable runtime layout changes, then it is recommended that you use ADF Faces layout components like `Panel Stretch Layout` and `Panel Group Layout`. It is recommended that you have just one `Layout Customizable` on the page.

A `Layout Customizable` component contains a direct child `Panel Customizable` component and two facets with `Panel Customizable` components by default to enable users to add content inside them at runtime. However, you can add any components inside the `Layout Customizable` component and its facets. The content in the three `Panel Customizable` components can be placed in different ways using predefined layouts. For more information, see Predefined Layout Types.

Table B–3 describes the attributes of a `Layout Customizable` component.

Table B–3 Attributes of the Layout Customizable Component

Attribute	Type	Supports EL?	Description
Common Attributes			
id	String	No	The identifier for the component. The identifier must follow a subset of the syntax allowed in HTML: ■ Must not be a zero-length String. First character must be an ASCII letter (A-Za-z) or an underscore ('_'). ■ Subsequent characters must be an ASCII letter or digit (A-Za-z0-9), an underscore ('_'), or a dash ('-').
rendered	boolean	Yes	Specifies whether the component is rendered. When set to `false`, no output will be delivered for this component (the component will not in any way be rendered, and cannot be made visible on the client). The default value is `true`.
type	String	Yes	Specifies the layout to be used for the page. You can choose from a list of eight predefined layouts. For information, see Table B–5. Default value is `threeColumn`.
Appearance Attributes			

Table B–3 (Cont.) Attributes of the Layout Customizable Component

Attribute	Type	Supports EL?	Description
text	String	Yes	Specifies the text to be displayed for the Change Layout menu. Use the `text` attribute to provide a descriptive label for the `Layout Customizable` component so that the component is clearly visible on the page at runtime. For example: `text="Change Layout"`
accessKey	char	Yes	A character used to gain quick access to this button. For accessibility reasons, this functionality is not supported in screen reader mode. If the same access key appears in multiple input fields in the same page of output, the rendering user agent will cycle among the elements accessed by the similar keys. Note that user agents are inconsistent about dealing with two links having same access key, and so the cycling behavior is dependent on what the user agent provides. This attribute is sometimes referred to as the "mnemonic". The character specified by this attribute must exist in the `Text` attribute of this button instance. If it does not, the user will receive no visual indication of the existence of the `accessKey`. The easiest, and most convenient way to specify both the text and the mnemonic together is to use `textAndAccessKey`. Note that the `accessKey` is triggered by browser-specific and platform-specific modifier keys. It even has browser-specific meaning. For example, Internet Explorer 7.0 will set focus when you press Alt+<accessKey>. Firefox 2.0 on some operating systems will set focus when you press Alt+Shift+<accessKey>. Firefox 2.0 on other operating systems will set focus when you press Control+<accessKey>. Refer to your browser's documentation for how it treats accessKey.
showIcon	boolean	Yes	Specifies whether the default layout change icon is visible or not. The default value is `true`.

Table B–3 (Cont.) Attributes of the Layout Customizable Component

Attribute	Type	Supports EL?	Description
showLayoutChanger	boolean	Yes	Specifies whether the layout changer icon or text must be displayed. The default value is `true`. When set to `true`, the runtime behavior of the layout changer is as follows: ■ Displayed in View mode. ■ Displayed in Edit mode. It is always displayed in Edit mode, even if the value of the `showLayoutChanger` attribute is `false`. ■ If there is a security or MDS restriction on the Layout Customizable component, then the layout changer is not displayed in View or Edit modes.
shortDesc	String	Yes	The short description of the component. This text is commonly used by user agents to display tooltip help text, in which case the behavior for the tooltip is controlled by the user agent, for example, Firefox 2 truncates long tooltips. For form components, the `shortDesc` is displayed in a note window.
Style Attributes			
styleClass	String	Yes	A CSS style class to use for this component. The style class can be defined in your JSPX page or in a skinning CSS file, for example.
inlineStyle	String	Yes	The CSS styles to use for this component. This is intended for basic style changes. The `inlineStyle` is a set of CSS styles that are applied to the root DOM element of the component. If the `inlineStyle`'s CSS properties do not affect the DOM element you want affected, then you will have to create a skin and use the skinning keys which are meant to target particular DOM elements, like `::label` or `::icon-style`.
Behavior Attributes			
binding	`oracle.adf.view.page.editor.component.LayoutCustomizable`	Yes	An EL reference that will store the component instance on a bean. This can be used to give programmatic access to a component from a backing bean, or to move creation of the component to a backing bean. For example: `binding="#{yourManagedBean.Binding}"`

Supported Facets

Table B–4 Facets of the Layout Customizable Component

Name	Description
contentA	Prepolutated with a `Panel Customizable` component. This `Panel Customizable` will contain content to be rendered in area A in the layouts shown in Table B–5.
contentB	Prepolutated with a `Panel Customizable` component. This `Panel Customizable` will contain content to be rendered in area B in the layouts shown in Table B–5.
facetSeparator	Content to be rendered once between each facet.
separator	Content to be rendered once between each of the other children.

Predefined Layout Types

When you add a `Layout Customizable` component to the page, three child `Panel Cutsomizable` components are added by default; one direct child and two `Panel Customizables` in the two facets. When you select the layout type, the components in these `Panel Customizables` are arranged according to the layout type chosen.

Table B–5 describes the eight layouts that you can apply to your page or an area of the page at design time. To easily describe how components are laid out, let us assume the following:

- The child `Panel Customizable` of `contentA` facet is called **A**.
- The child `Panel Customizable` of `contentB` facet is called **B**.
- The primary `Panel Customizable`, which is a direct child of the `Layout Customizable`, is called **C**.

Table B–5 Values for Type Attribute of Layout Customizable

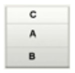
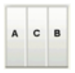
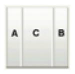
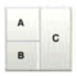
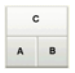
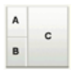
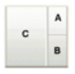
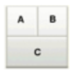
Layout Type	Image Showing Content Arrangement
oneColumn	
threeColumn	
threeColumnNarrow	
twoColumn	
twoColumnBottom	
twoColumnNarrowLeft	
twoColumnNarrowRight	

Table B–5 (Cont.) Values for Type Attribute of Layout Customizable

Layout Type	Image Showing Content Arrangement
twoColumnTop	

B.1.4 Panel Customizable Component

Use the `Panel Customizable` component as a container for page content that can be customized at runtime. You can add content inside a `Panel Customizable` component at runtime using Oracle Composer's Resource Catalog. When you add child `Show Detail Frame` components inside a `Panel Customizable`, the `Show Detail Frame` components provide runtime customization options for arranging or deleting components on the page.

Use this component only to perform runtime customizations on child components. If you just want a container to arrange components, then it is recommended that you use an ADF Faces container like `Panel Group Layout`.

Geometry Management

The `Panel Customizable` component can be stretched by a parent layout component that stretches its children, for example, a `Panel Stretch Layout` or `Panel Splitter` component. By setting the `layout` attribute on the `Panel Customizable` component to `stretch`, the child component can be stretched to the size of the `Panel Customizable`.

Table B–6 describes the attributes of a `Panel Customizable` component.

Table B–6 Attributes of a Panel Customizable Component

Attribute	Type	Supports EL?	Description
Common Attributes			
id	String	No	The identifier for the component. The identifier must follow a subset of the syntax allowed in HTML: ■ Must not be a zero-length String. First character must be an ASCII letter (A-Za-z) or an underscore ('_'). ■ Subsequent characters must be an ASCII letter or digit (A-Za-z0-9), an underscore ('_'), or a dash ('-').
rendered	boolean	Yes	Specifies whether the component is rendered. When set to `false`, no output will be delivered for this component (the component will not in any way be rendered, and cannot be made visible on the client). The default value is `true`.

Table B–6 (Cont.) Attributes of a Panel Customizable Component

Attribute	Type	Supports EL?	Description
layout	String	Yes	Specifies how the children of the `Panel Customizable` must be laid out. Available options are `horizontal`, `vertical`, `scroll`, and `stretch`. The default value is `vertical`. Depending on the value of the `layout` attribute, child components are laid out as follows: ■ If you select `vertical`, then the child components are displayed one below the other and can be moved either up or down within the layout. ■ If you select `horizontal`, then the child components are displayed adjacent to each other and can be moved either to the left or right within the layout. ■ If you select `scroll`, then the child components are displayed one below the other, with scrollbars displayed if content overflows. ■ If you select `stretch`, then the first child component is stretched to fill up available space in the `Panel Customizable` component. Subsequent child components are ignored. However, they are not removed from the page.
Style Attributes			
styleClass	String	Yes	A CSS style class to use for this component. The style class can be defined in your JSPX page or in a skinning CSS file, for example.

Table B–6 (Cont.) Attributes of a Panel Customizable Component

Attribute	Type	Supports EL?	Description
inlineStyle	String	Yes	The CSS styles to use for this component. This is intended for basic style changes. The `inlineStyle` is a set of CSS styles that are applied to the root DOM element of the component. If the `inlineStyle`'s CSS properties do not affect the DOM element you want affected, then you will have to create a skin and use the skinning keys which are meant to target particular DOM elements, like `::label` or `::icon-style`.
Behavior Attributes			
allowAction	boolean	Yes	Determines if this component allows addition of child `Show Detail Frame` components, moving `Show Detail Frame` components out of it, and rearranging child `Show Detail Frame` components at runtime. Available values are `none` and `all`. The default value is `all`. **Note:** You can also set this attribute while defining component security in the application's `adf-config.xml` file. For information, see Section 5.8.4, "Applying Action-Level Restrictions on Show Detail Component Actions."
showEditIcon	boolean	Yes	While in Edit mode, in Design view, renders an Edit icon on the `Panel Customizable` header that enables you to edit component properties at runtime. Available values are `true` and `false`. Default value is `true`. This is relevant only in an Oracle Composer-enabled page, that is, if the `Panel Customizable` component is nested within a `Page Customizable` component.
Advanced Attributes			
binding	`oracle.adf.view.page.editor.component.PanelCustomizable`	Yes	An EL reference that will store the component instance on a bean. This can be used to give programmatic access to a component from a backing bean, or to move creation of the component to a backing bean.

Disabling the Edit Option on a Panel Customizable Component

When you switch to Edit mode of a page at runtime, in Design view you will notice a Pencil icon on all `Panel Customizable` or `Box` components that can be edited. To disable property editing on a specific `Panel Customizable` or `Box` component in Design view, you must set the `showEditAction` attribute on the component to `false` at design time. The Pencil icon is no longer displayed on the component in Design view. However, users can still select such a component in Source view and edit its properties.

B.1.5 Show Detail Frame Component

Add `Show Detail Frame` components inside `Panel Customizable` components to enable runtime customizations like move, minimize, restore, and delete of child components. You can move `Show Detail Frame` components only if they are added inside a `Panel Customizable` component.

Geometry Management

A `Show Detail Frame` component can be stretched by a parent layout component that stretches its children, for example, `Panel Customizable`.

A `Show Detail Frame` component stretches its child component to fill the height and width available to it. It is recommended that you add only one child component inside a `Show Detail Frame` component. However, if you have more than one child component, the first child component is stretched to the height and width of the `Show Detail Frame`'s content area and subsequent child components are ignored.

The resize handler at the lower right corner of a `Show Detail Frame` enables you to resize the component vertically. From then on, the `Show Detail Frame` cannot be stretched by its parent.

Table B–7 describes the attributes of a `Show Detail Frame` component.

Table B–7 Attributes of a Show Detail Frame Component

Attribute	Type	Supports EL?	Description
Common Attributes			
id	String	No	The identifier for the component. The identifier must follow a subset of the syntax allowed in HTML: ▪ Must not be a zero-length String. First character must be an ASCII letter (A-Za-z) or an underscore ('_'). ▪ Subsequent characters must be an ASCII letter or digit (A-Za-z0-9), an underscore ('_'), or a dash ('-').
rendered	boolean	Yes	Specifies whether the component is rendered. When set to `false`, no output will be delivered for this component (the component will not in any way be rendered, and cannot be made visible on the client). The default value is `true`.
text	String	Yes	A title for the `Show Detail Frame` component.

Table B–7 (Cont.) Attributes of a Show Detail Frame Component

Attribute	Type	Supports EL?	Description

icon	String	Yes	If you decide to add an icon on the header of the `Show Detail Frame` component, then this specifies the URI for the image to be used.
			For example:
			`icon="http://source-pc/images/accessability.gif"`
			Note that an image that is stored at the document root does not require a full path. For example:
			`icon="detail.gif"`

Appearance Attributes			
shortDesc	String	Yes	The short description of the component. This text is commonly used by user agents to display tooltip help text, in which case the behavior for the tooltip is controlled by the user agent, for example, Firefox 2 truncates long tooltips. For form components, the `shortDesc` is displayed in a note window.
background	String	Yes	Working with the skin CSS, provides a means of applying a different look and feel for this `Show Detail Frame` instance. Available values are `light`, `medium`, and `dark`. The default value is `medium`.
displayHeader	boolean	Yes	Indicates whether the header of the `Show Detail Frame` is displayed. The default value is `true`. If you choose to set `displayHeader` to `false` and if you have exposed some actions on the component, then a toolbar is displayed when you move the mouse over the component area. The toolbar contains a drop down icon, which displays a menu of available options. This toolbar will display only in Edit mode, if there are actions available on the component.
displayShadow	boolean	Yes	Specifies whether a shadow must be cast by the chrome of the `Show Detail Frame` component. The default value is `false`.

Table B–7 (Cont.) Attributes of a Show Detail Frame Component

Attribute	Type	Supports EL?	Description
expansionMode	String	Yes	The default state of the `Show Detail Frame`. Available values are `minimized` and `normal`. The default display mode is `normal`.
Actions Attributes			
displayActions	String	Yes	Specifies when seeded interactions must be displayed. Available options are `onHover` and `always`. The default is `always`.
inheritGlobalActions	boolean	Yes	This attribute is of significance if you add child custom action components, and the attribute specifies whether the global `action` value of a `custom action` must override the local value. The default value is `false`. For information about global custom actions, see Section 4.2.13.2, "Defining Custom Actions at the Global Level."
showMoveAction	String	Yes	Specifies whether the Move action must be displayed in the Actions menu. Available values are `menu` and `none`. The default value is `menu`.
showRemoveAction	String	Yes	Renders a Remove icon on the `Show Detail Frame` header. Clicking this icon removes the component from the page. Available values are `chrome` and `none`. The default value is `none`.
showResizer	String	Yes	Specifies whether a resize handle must be displayed on the lower right corner of the `Show Detail Frame`. You can alter only the height of a `Show Detail Frame` while resizing it. Available values are `none` and `always`. The default value is `always`.
showMinimizeAction	String	Yes	Specifies whether the minimize action must be displayed on the header. Available values are `chrome` and `none`. The default is `chrome`.

Table B–7 (Cont.) Attributes of a Show Detail Frame Component

Attribute	Type	Supports EL?	Description
showEditAction	boolean	Yes	While in Edit mode, in Design view, renders an Edit icon on the `Show Detail Frame` header that enables you to edit component properties at runtime. Default value is `true`. This is relevant only in an Oracle Composer-enabled page, that is, if the `Show Detail Frame` component is nested within a `Page Customizable` component.
Style Attributes			
styleClass	String	Yes	A CSS style class to use for this component. The style class can be defined in your JSPX page or in a skinning CSS file, for example.
inlineStyle	String	Yes	The CSS styles to use for this component. This is intended for basic style changes. The `inlineStyle` is a set of CSS styles that are applied to the root DOM element of the component. If the `inlineStyle`'s CSS properties do not affect the DOM element you want affected, then you will have to create a skin and use the skinning keys which are meant to target particular DOM elements, like `::label` or `::icon-style`.
contentStyle	String	Yes	The CSS style to apply to the `Show Detail Frame` content area. Manually enter any style in compliance with CSS version 2.0 or later. Th CSS styles in this attribute are commonly used to define the height and background color of the `Show Detail Frame`. For example, `height:200px; background-color:green;`
Behavior Attributes			

Table B–7 (Cont.) Attributes of a Show Detail Frame Component

Attribute	Type	Supports EL?	Description
partialTriggers	String	Yes	The IDs of the components that should trigger a partial update. This component will listen on the trigger components. If a trigger component receives an event that will cause it to update in some way, this component will request to be updated too. Identifiers are relative to the source component (this component), and must account for NamingContainers. If your component is inside of a naming container, you can use a single colon to start the search from the root of the page, or multiple colons to move up through the NamingContainers - "`::`" will pop out of the component's naming container (or itself if the component is a naming container) and begin the search from there, "`:::`" will pop out of two naming containers (including itself if the component is a naming container) and begin the search from there.
disclosureListener	`javax.el.MethodExpression`	Yes	A method reference to a disclosure listener. A disclosure event is fired when the user expands or collapses the `Show Detail Frame` component.
stretchContent	boolean	Yes	Specifies whether stretching of child components is enabled within the `Show Detail Frame`. The default value is `true`. You can use the `stretchContent` attribute to stretch the child component that supports being stretched. If you select `true`, then the child component is stretched to the height and width of the `Show Detail Frame`'s content area. The default height of the `Show Detail Frame` is 200px. The height of the `Show Detail Frame` is defined using the `ContentStyle` attribute.
Advanced Attributes			
binding	`oracle.adf.view.page.editor .component.ShowDetailFrame`	Yes	An EL reference that will store the component instance on a bean. This can be used to give programmatic access to a component from a backing bean, or to move creation of the component to a backing bean.

Table B–7 (Cont.) Attributes of a Show Detail Frame Component

Attribute	Type	Supports EL?	Description
attributeChangeListener	`javax.el.MethodExpression`	Yes	A method reference to an attribute change listener. Attribute change events are not delivered for any programmatic change to a property. They are only delivered when a renderer changes a property without the application's specific request. An example of an attribute change events might include the width of a column that supported client-side resizing.
selectChild	String	Yes	Specifies whether runtime selection and customization must be enabled on child components. This is relevant only in an Oracle Composer-enabled page, that is, if the `Show Detail Frame` component is nested within a `Page Customizable` component. Default value is `yes`. However, when you add a task flow as a child of the `Show Detail Frame` component, this attribute defaults to `no`. If you select `yes`, then child components can be selected and customized at runtime.
Other Attributes			
helpTopicID	String	Yes	Specifies a Help topic ID that can be used to link to a custom Help topic from the component. To link to a Help topic from your `Show Detail Frame` component instance, you can either link to an HTML file that is stored directly in your application directory, or to a Help topic that is part of a JAR file.

Supported Facets

Table B–8 Show Detail Frame Facets

Name	Description
titleBarAction	Used if an action is to be associated with title of the `Show Detail Frame` component. The component specified by the facet will be rendered in place of the `Show Detail Frame`'s `text` attribute.
additionalActions	Used if some additional actions are to be added to the Actions menu available on the `Show Detail Frame` header. To ensure that the additional actions appear similar to other actions on the menu, use `Command Menu Item` components or a `Group` component with nested `Command Menu Item` components inside this facet.

Displaying Child Component Properties Along with Show Detail Frame Properties

When you switch to Edit mode of a page at runtime, in Design view you will notice a Pencil icon on all `Show Detail Frame` components that can be edited. Clicking this icon invokes the Component Properties dialog in which users can edit the `Show Detail Frame`'s properties. If you want the Component Properties dialog to display the `Show Detail Frame`'s properties and its child component's properties, then you can use the custom attribute, `sdf_selection_rule`.

In JDeveloper, right-click the `Show Detail Frame` component and select **Insert inside cust:showDetailFrame**, **JSF core**, and then **Attribute**. In the Insert Attribute dialog, specify `sdf_selection_rule` as the name and `sdf_for_edit_mode_only` as the value. In Source mode, this attribute will appear as shown in the following example:

```
<cust:showDetailFrame text="showDetailFrame 1" id="sdf1">
  <f:attribute name="sdf_selection_rule" value="sdf_for_edit_mode_only"/>
</cust:showDetailFrame>
```

Disabling the Edit Option on a Show Detail Frame Component

When you switch to Edit mode of a page at runtime, in Design view you will notice a Pencil icon on all `Show Detail Frame` components that can be edited. To disable property editing on a specific `Show Detail Frame` or `Movable Box` component in Design view, then you can do this by setting the `showEditAction` attribute on the component to `false` at design time. The Pencil icon is no longer displayed on the component when you enter Edit mode of the page at runtime. However, users can still select such a component in Source view and edit its properties.

B.1.6 Custom Action

Table B–9 describes the attributes of a `Custom Action` component that you can add within a `Show Detail Frame` component. `Custom Action` components can be linked to navigation rules specified on a task flow so that the `Show Detail Frame` will display it in the Actions menu.

You can add as many `Custom Action` components as there are ADFc outcomes in the task flow. If you add a `Custom Action` attribute without a corresponding ADFc outcome, or if the `action` attribute value does not match an ADFc outcome defined on the view that is currently being displayed by the task flow, then that action is not displayed at runtime.

Table B–9 Attributes of a Custom Action Component

Attribute	Type	Supports EL?	Description
Common Attributes			
id	String	No	The identifier for the component. The identifier must follow a subset of the syntax allowed in HTML: ■ Must not be a zero-length String. First character must be an ASCII letter (A-Za-z) or an underscore ('_'). ■ Subsequent characters must be an ASCII letter or digit (A-Za-z0-9), an underscore ('_'), or a dash ('-').
action	String	Yes	Specifies the action outcomes defined for the task flow. The value of this attribute must match the relevant ADFc outcome in the task flow definition file.

Table B–9 (Cont.) Attributes of a Custom Action Component

Attribute	Type	Supports EL?	Description
rendered	boolean	Yes	Specifies whether the component is rendered. When set to `false`, no output will be delivered for this component (the component will not in any way be rendered, and cannot be made visible on the client). The default value is `true`.
text	String	Yes	Represents the text of the menu item link. This is applicable for a custom action rendered as a menu item link.
Appearance Attributes			

icon	String	Yes	For a custom action rendered as a toolbar link, this specifies the URI for the icon to be rendered on the chrome.
			For example:
			`icon="http://source-pc/images/accessability.gif"`
			Note that an image that is stored at the document root does not require a full path. For example:
			`icon="detail.gif"`
			For a custom action rendered as a menu item link, this represents the icon to be displayed against the menu item text. That is, toward its left for LTR locales.

shortDesc	String	Yes	The short description of the component. This text is commonly used by user agents to display tooltip help text, in which case the behavior for the tooltip is controlled by the user agent, for example, Firefox 2 truncates long tooltips. For form components, the `shortDesc` is displayed in a note window. This is applicable only for a custom action rendered as a toolbar icon.
location	String	Yes	Determines whether the custom action is rendered as a toolbar icon or a menu item link. Available values are `chrome`, `menu`, or `both`. Default value is `chrome`.

B.1.7 ImageLink

Use the `Image Link` component to include an image with a hyperlink on the page. The `Image Link` component makes the runtime customization experience easier by adding an image and a link in one step.

Table B–10 describes the attributes of an `Image Link` component.

Table B–10 Attributes of an Image Link Component

Attribute	Type	Supports EL?	Description
Common Attributes			
id	String	No	The identifier for the component. The identifier must follow a subset of the syntax allowed in HTML: ■ Must not be a zero-length String. First character must be an ASCII letter (A-Za-z) or an underscore ('_'). ■ Subsequent characters must be an ASCII letter or digit (A-Za-z0-9), an underscore ('_'), or a dash ('-').
rendered	boolean	Yes	Specifies whether the component is rendered. When set to `false`, no output will be delivered for this component (the component will not in any way be rendered, and cannot be made visible on the client). The default value is `true`.
source	String	Yes	Displays the location of the image.
shortDesc	String	Yes	The short description of the component. This text is commonly used by user agents to display tooltip help text, in which case the behavior for the tooltip is controlled by the user agent, for example, Firefox 2 truncates long tooltips. For form components, the `shortDesc` is displayed in a note window.
Link Attributes			
destination	String	Yes	Specifies the URL this image component references.
targetFrame	String	Yes	Specifies the target frame for this component. You can specify values like `_blank`, `_self`, or any other value allowed by HTML, including the names of frames.
Appearance Attributes			

accessKey	String	Yes	Specifies a keyboard shortcut to gain quick access to this component.
			Note: The key specified in this attribute must exist in the `text` attribute for this component instance.
			You can specify browser-specific access keys as follows:
			■ For Internet Explorer, specify
			`Alt + <`*`key`*`>`
			■ For Firefox, specify
			`Ctrl+Alt+<`*`key`*`>`

longDescURL	String	Yes	Specifies the URL to a document that contains a long description of the image.
Style Attributes			

Table B–10 (Cont.) Attributes of an Image Link Component

Attribute	Type	Supports EL?	Description
styleClass	String	Yes	A CSS style class to use for this component. The style class can be defined in your JSPX page or in a skinning CSS file, for example.
inlineStyle	String	Yes	The CSS styles to use for this component. This is intended for basic style changes. The `inlineStyle` is a set of CSS styles that are applied to the root DOM element of the component. If the `inlineStyle`'s CSS properties do not affect the DOM element you want affected, then you will have to create a skin and use the skinning keys which are meant to target particular DOM elements, like `::label` or `::icon-style`.
Behavior Attributes			
disabled	boolean	Yes	Specifies whether the `Image Link` component should be considered disabled. The default value is `false`. The `disabled` attribute is of significance only when the `Image Link` component is used in container components like global headers and tab bars.
partialTriggers	String	Yes	The IDs of the components that should trigger a partial update. This component will listen on the trigger components. If trigger component receives an event that will cause it to update in some way, this component will request to be updated too. Identifiers are relative to the source component (this component), and must account for NamingContainers. If your component is inside of a naming container, you can use a single colon to start the search from the root of the page, or multiple colons to move up through the NamingContainers - "`::`" will pop out of the component's naming container (or itself if the component is a naming container) and begin the search from there, "`:::`" will pop out of two naming containers (including itself if the component is a naming container) and begin the search from there.
Advanced Attributes			

Table B–10 (Cont.) Attributes of an Image Link Component

Attribute	Type	Supports EL?	Description
binding	`oracle.adf.view.page.editor.component.ImageLink`	No	An EL reference that will store the component instance on a bean. This can be used to give programmatic access to a component from a backing bean, or to move creation of the component to a backing bean.
visible	boolean	Yes	Specifies whether the component must be visible on the client. The default value is `true`. If `Rendered` was set to `false`, then the component is not rendered and cannot be made visible on the client even if `visible` is set to `true`.
attributeChangeListener	`javax.el.MethodExpression`	Yes	A method reference to an attribute change listener. Attribute change events are not delivered for any programmatic change to a property. They are only delivered when a renderer changes a property without the application's specific request. An example of an attribute change events might include the width of a column that supported client-side resizing.

B.2 Oracle Composer-Specific Files and Configurations

This section describes the various files that can be modified to extend Oracle Composer capabilities. While some files, for example the Oracle Composer extension file, `pe_ext.xml`, are entirely relevant to Oracle Composer only, others are ADF configuration files that can take entries for Oracle Composer-specific configurations.

This section includes the following topics:

- Section B.2.1, "pe_ext.xml"
- Section B.2.2, "adf-config.xml"
- Section B.2.3, "web.xml"

B.2.1 pe_ext.xml

The Oracle Composer extension file, `pe_ext.xml`, enables you to extend editing capabilities provide by Oracle Composer. Within this file you can add elements to register new Oracle Composer add-ons and custom property panels, selectively render panels, register event handlers, and define property filters. This file is not available by default when you add Oracle Components to a page. You can create this file in the `META-IN`F directory. The following sections describe the different elements of the Oracle Composer extension file schema and explain when you may want to use the schema elements.

B.2.1.1 addon-config

Use the `addon-config` element to include entries for new Oracle Composer add-ons and custom property panels. For example, to add a new About button on your page that invokes a panel displaying information about your application, you must first create a task flow containing the information and then register this task flow in the `pe_ext.xml` file. Oracle Composer add-ons declared inside the `addon-config` element are configured in the `adf-config.xml` file, whereas custom property panels declared within the `addon-config` element are configured within the property-panels element in the extension file itself. For information about configuring add-ons in `adf-config.xml`, see Section B.2.2.1, "addon-panels." For information about configuring custom property panels using property-panels, see Section B.2.1.2, "property-panels."

Within the `addon-config` element you can have one `panels` elements to define add-ons.

panels

Use the `panels` element to register new add-ons or property panels in Oracle Composer. You can have only one `panels` element in an extension file. Within the `panels` element you can insert any number of individual `panel` elements that define the add-ons or property panels to be displayed in Oracle Composer. You must have one `panel` element for every add-on and property panel you want to display.

panel

Use `panel` elements to define individual add-ons and property panels that must be displayed to users.

Table B–11 describes the attributes of the `panel` element.

Table B–11 panel Element Attributes

Attribute	Description
`name`	An identifier that will be used while referencing the new add-on in `adf-config.xml`, or a property panel in the `property-panels` section. This must be unique in the application.
`title`	The title to be displayed on the button used to invoke the task flow in Oracle Composer. You can use EL for this property to show a localized title. For custom property panels, the title to be displayed on the new tab for that panel.
`icon`	The icon that will appear next to the title on the button. This is optional.
`taskflow-id`	The ID of the task flow that defines the add-on or property panel.
`help-topic-id`	A help topic ID that can be used to link to a custom help topic from the add-on or property panel. This is optional.

When a `panel` element is deleted from the file, that add-on or property panel is not displayed on the page.

The following example shows how the `addon-config`, `panels`, and `panel` elements can be used:

```
<addon-config>
  <panels>
    <panel name="oracle.fod.custom.panel" title="About FOD"
           icon="adf/webcenter/images/about.gif"
           taskflow-id="/WEB-INF/about-fod.xml#about-fod"
    />
  </panels>
</addon-config>
```

For more information about add-ons, see Section 5.2, "Creating Oracle Composer Add-Ons."

B.2.1.2 property-panels

Use the `property-panels` element to register custom property panels that you declared in the `addon-config` element in the extension file. For information about custom property panels, see Section 5.3, "Creating Custom Property Panels." You can have only one `property-panels` element in an extension file. Within the `property-panels` element you can insert any number of individual `property-panel` elements to define custom property panels that must be displayed as tabs in the Component Properties dialog.

property-panel

Use the `property-panel` element to define a custom property panel that must be displayed for a particular component in Oracle Composer. You can insert any number of individual `property-panel` elements to define custom property panels that must be displayed as tabs in the Component Properties or Page Properties dialog. To specify the component or task flow for which you want to register a custom property panel, you must add a `component` or `taskflow-id` element respectively within the `property-panel`.

Table B–12 property-panel Element Attributes

Attribute	Description
`name`	An identifier for the custom property panel you want to display in the Component Properties dialog.
`rendered`	Whether this panel must be displayed to users in the Component Properties dialog. This can take an EL value.

component

Use the `component` element to specify the fully-qualified class name of the component for which you are registering the custom property panel.

taskflow-id

Use the `taskflow-id` element to specify the name of the task flow for which you are registering the custom property panel.

panel

Use the `panel` element to specify a custom property panel that must be displayed as a tab in the Component Properties or Page Properties dialog for the component. Table B–13 describes the attributes of a `panel` element.

Table B–13 panel Element Attributes

Attribute	Description
`name`	Name with which you declared the panel in the `addon-config` section.
`rendered`	Whether the custom property panel must be rendered or not.
`parameter`	Parameters to be passed to the implementing taskflow. Takes an EL only and must return a `Map<String, String>`.

When a `panel` element is deleted from the file, that property panel is not displayed in the Component Properties dialog.

The following example shows how to register a custom property panel for a `Command Button` component:

```
<property-panels>
  <property-panel name="cmdbtn">
    <component>oracle.rich.CommandButton</component>
    <panel name="prop.panel.cmdbtn" />
  </property-panel>
</property-panels>
</pe-extension>
```

The following example shows how to register a custom property panel for a task flow:

```
<property-panels>
  <property-panel name="dashboard">
    <taskflow-id>/WEB-INF/dashboard-taskflow#prop-panel</taskflow-id>
    <panel name="dashboard.prop-panel" />
  </property-panel>
</property-panels>
</pe-extension>
```

B.2.1.3 event-handlers

Use the `event-handlers` element to register handlers for events generated by Oracle Composer. You can have only one `event-handlers` element in an extension file. Within `event-handlers` you can have individual `event-handler` elements to register handlers for save, close, deletion, addition, and selection events.

event-handler

Use the `event-handler` element to register an event handler in Oracle Composer. For example, if you have implemented the `processSave` method in a Java class called `SaveHandler` to perform a specific action when a Save event is invoked, then you can register that implementation in Oracle Composer by adding an `event-handler` entry in the `pe_ext.xml` file as follows:

```
<pe-extension xmlns="http://xmlns.oracle.com/adf/pageeditor/extension">
. . .
<event-handlers>
  <event-handler event="save">view.SaveHandler</event-handler>
. . .
</event-handlers>
</pe-extension>
```

For information about configuring event handlers, see Section 5.4, "Configuring Event Handlers for Oracle Composer UI Events."

B.2.1.4 filter-config

Use the `filter-config` element to define property filters for components that can be edited at runtime. For a selected component, the Component Properties dialog displays properties that can be edited. If you do not want to expose all properties for editing, you can filter specific properties so that they are not displayed in the Component Properties dialog. You can specify global-level filters to filter common attributes across all components, or you can define tag-level filters to filter properties for a particular component only. You can have only one `filter-config` element in an extension file.

For information about property filters, see Section 5.5, "Defining Property Filters."

global-attribute-filter

Use the `global-attribute-filter` element to filter specific properties across all components. You can have only one `global-attribute-filter` element in the extension file. Within this element you can include any number of `attribute` elements to define property filters for components from different libraries.

taglib-filter

Use the `taglib-filter` element to define property filters for components within a particular library. You can include multiple `taglib-filter` elements to filter properties for components in different libraries. Within this element you can include `tag` elements for every component belonging to the library.

Table B–14 tag Element Attribute

Attribute	Description
`namespace`	Namespace of the tag library containing the component whose properties you want to filter.

tag

Use the `tag` element to specify a component whose properties you want to filter. You can include multiple `tag` elements within a `taglib-filter` element. Within this element you can include `attribute` elements for all properties you want to filter.

Table B–15 tag Element Attribute

Attribute	Description
`name`	Name of the component whose properties you want to filter, for example, `commandButton`, `portlet`.

attribute

Use the `attribute` element to specify a property that must be filtered either across all components or for the specified component only. To define global-level filters, you must include `attribute` elements inside the `global-attribute-filter`. To define property filters for a particular component, you must include `attribute` elements inside the `taglib-filter` element.

Table B–16 attribute Element Attributes

Attribute	Description
`name`	Name of the property you want to filter.
`filtered`	Whether the property must be filtered or not. Takes a value of `true` or `false`.

The following example shows how you can define global and component-level property filters:

```
<filter-config>
  <global-attribute-filter>
    <attribute name="accessKey" />
    <attribute name="attributeChangeListener" />
    <attribute name="autoSubmit" />
    <attribute name="binding" />
  </global-attribute-filter>
  <taglib-filter namespace="http://xmlns.oracle.com/adf/faces/rich">
    <tag name="commandButton">
      <attribute name="text" />
      <attribute name="icon" />
    </tag>
</filter-config>
```

B.2.2 adf-config.xml

The `adf-config.xml` file specifies application-level settings that are usually determined at deployment and are often changed at runtime. This file is created when you create your application and is located in the `ADF META-INF` folder under `Descriptors` in the Application Resources panel. The following example shows the minimal `adf-config.xml` file created when you create a WebCenter application:

```
<?xml version="1.0" encoding="windows-1252" ?>
<adf-config xmlns="http://xmlns.oracle.com/adf/config"
            xmlns:adf="http://xmlns.oracle.com/adf/config/properties">
  <adf:adf-properties-child xmlns="http://xmlns.oracle.com/adf/config/properties">
    <adf-property name="adfAppUID" value="Application4-2138"/>
  </adf:adf-properties-child>
</adf-config>
```

When you add the `Page Customizable` to a page, the necessary configuration is added to the `adf-config.xml` file and is as shown in the following example:

```
<?xml version="1.0" encoding="windows-1252" ?>
<adf-config xmlns="http://xmlns.oracle.com/adf/config"
            xmlns:adf="http://xmlns.oracle.com/adf/config/properties"
            xmlns:mdsC="http://xmlns.oracle.com/adf/mds/config">
  <adf:adf-properties-child xmlns="http://xmlns.oracle.com/adf/config/properties">
    <adf-property name="adfAppUID" value="configfilesapp-7198"/>
  </adf:adf-properties-child>
  <mdsC:adf-mds-config version="11.1.1.000">
    <mds-config xmlns="http://xmlns.oracle.com/mds/config">
      <persistence-config>
        <metadata-namespaces>
          <namespace path="/oracle/adf/rc/metadata"
                     metadata-store-usage="WebCenterFileMetadataStore"/>
          <namespace path="/persdef/"
                     metadata-store-usage="WebCenterFileMetadataStore"/>
        </metadata-namespaces>
        <metadata-store-usages>
          <metadata-store-usage id="WebCenterFileMetadataStore"
                                default-cust-store="true">
            <metadata-store
class-name="oracle.mds.dt.persistence.stores.file.SrcControlFileMetadataStore">
              <property name="metadata-path" value="../../mds"/>
            </metadata-store>
          </metadata-store-usage>
        </metadata-store-usages>
      </persistence-config>
      <cust-config>
        <match>
          <customization-class name="oracle.adf.share.config.SiteCC"/>
        </match>
      </cust-config>
      <cache-config>
        <max-size-kb>100000</max-size-kb>
      </cache-config>
    </mds-config>
  </mdsC:adf-mds-config>
</adf-config>
```

You must update the `adf-config.xml` file when you perform tasks like registering new add-ons and custom property panels, selectively rendering add-ons, creating customization layers, enabling and disabling Oracle Composer sandbox, creating custom Resource Catalogs, and disabling Source view. The following sections describe the various `adf-config.xml` elements used for Oracle Composer-specific configurations.

B.2.2.1 addon-panels

Use the `addon-panels` element to register custom add-ons and property panels in Oracle Composer. You can have only one `addon-panels` element in the file. Within this you can have any number of `addon-panel` elements to register custom panels. For more information about Oracle Composer add-ons and property panels, see Section 5.2, "Creating Oracle Composer Add-Ons" and Section 5.3, "Creating Custom Property Panels."

addon-panel

Use an `addon-panel` element to register an add-on in the `adf-config.xml` file. An `addon-panel` element must only be included inside the `addon-panels` element. You can have any number of `addon-panel` elements to register custom panels.

The following example shows how you can use `addon-panels` and `addon-panel` elements to register a new add-on:

```
  <addon-panels>
    <!-- Page Properties add-on -->
    <addon-panel name="oracle.adf.pageeditor.addonpanels.page-settings" />

    <!-- Page Reset add-on -->
    <addon-panel name="oracle.adf.pageeditor.addonpanels.page-reset" />

    <addon-panel name="oracle.fod.custom.panel" />

  </addon-panels>
```

The `addon-panel` element also supports the `rendered` attribute. This attribute can take an EL value and specifies whether the panel must be rendered in Oracle Composer or not.

When registering custom add-ons, you must also include entries for the default add-ons. If the default add-ons are not registered, then only your custom add-on will be available in Oracle Composer.

B.2.2.2 sandbox-namespaces

Use the `sandbox-namespaces` element to register namespaces for all metadata for which you want to enable sandbox creation at runtime. For more information about enabling sandbox creation, see Section 5.7.1, "How to Enable Oracle Composer Sandbox Creation."

The following example shows how to use the `sandbox-namespaces` element:

```
<sandbox-namespaces>
  <namespace path="/test"/>
  <namespace path="/pageDefs"/>
</sandbox-namespaces>
```

B.2.2.3 session-options-factory

When creating customization layers in your application, use the `session-options-factory` element to register the `ComposerSessionOptionsFactory` implementation with Oracle Composer. This class will contain the logic to save customizations in different layers based on the specified criteria. For more information, see Section 5.6.1, "Adding Customization Layers to View and Edit Modes: Example."

The following example shows how to register a class named `AppsSessionOptionsFactoryImpl`:

```
<page-editor-config xmlns="http://xmlns.oracle.com/adf/pageeditor/config">

<session-options-factory>view.AppsSessionOptionsFactoryImpl</session-options-facto
ry>
</page-editor-config>
```

B.2.2.4 rcv-config

Use the `rcv-config` element to register custom Resource Catalogs with Oracle Composer and to register the `ResourceCatalogSelector` implementation when using multiple Resource Catalogs.

If you wish to display a custom Resource Catalog to users in place of the default one, you must first create a catalog definition file with the required content and then register this catalog definition in the Oracle Composer extension file and `adf-config.xml` file. The custom Resource Catalog is then displayed when a user clicks an Add Content button on the page. For more information, see Section 6.3, "Creating a Custom Resource Catalog."

The following example shows how you can use the `rcv-config` element to register a custom catalog called `users-catalog`:

```
<rcv-config xmlns="http://xmlns.oracle.com/adf/rcs/viewer/adf-config">
  <default-catalog catalog-name="users-catalog"/>
</rcv:rcv-config>
```

You can also configure multiple Resource Catalogs with Oracle Composer such that different catalogs are displayed to different users depending on the specified criteria. When configuring multiple Resource Catalogs, use the `rcv-config` element with a `catalog-selector` element to register the `ResourceCatalogSelector` implementation, which contains the logic for selecting a Resource Catalog.

The following example shows how you can use the `rcv-config` element to register a `ResourceCatalogSelector` implementation called `CatalogSelector`:

```
<rcv-config xmlns="http://xmlns.oracle.com/adf/rcs/viewer/adf-config">
  <catalog-selector class-name="webcenter.CatalogSelector"/>
  <default-catalog catalog-name="default-catalog"/>
</rcv-config>
```

B.2.2.5 customizableComponentsSecurity

Use the `customizableComponentsSecurity` element to apply restrictions on `Show Detail Frame` actions.

Within the `customizableComponentsSecurity` element, you can specify the following:

enableSecurity

Use the `enableSecurity` element to override the security inheritance behavior on `Show Detail Frame` actions. This element can take a value of `true` or `false`. If set to `true`, the ability for a user to modify a component will first be determined from the page permissions and then adjusted according to the current set of actions defined for that type of permission. If set to `false`, all actions are available to users. A user's page permissions and actions configured in `adf-config.xml` are ignored.

actionsCategory

Use the `actionsCategory` element to apply restrictions on a group of `Show Detail Frame` actions simultaneously. Within the `actionsCategory` element, you can have **`actionCategory`** elements for the supported categories. For more information about action categories, see Section 5.8.4.2, "Defining Security at the Actions Category Level."

The following example shows how `actionsCategory` can be used:

```
<cust:customizableComponentsSecurity
xmlns="http://xmlns.oracle.com/adf/faces/customizable/config">
  <cust:enableSecurity value="true"/>

  <cust:actionsCategory>
    <cust:actionCategory name="personalizeActionsCategory" value="false"/>
    <cust:actionCategory name="editActionsCategory" value="true"/>
  </cust:actionsCategory>

  <cust:actions>
  .........................................
  </cust:actions>

</cust:customizableComponentsSecurity>
```

actions

Use the `actions` element to apply restrictions on individual actions. Within each actions element, you can have **`action`** elements for all supported actions. For more information about action-level restrictions, see Section 5.8.4.3, "Defining Security at the Actions Level."

The following example shows how `actions` can be used:

```
<cust:customizableComponentsSecurity
xmlns="http://xmlns.oracle.com/adf/faces/customizable/config">
  <cust:enableSecurity value="true"/>

  <cust:actionsCategory>
    .........................................
  </cust:actionsCategory>

  <cust:actions>
    <cust:action name="showMinimizeAction" value="true"/>
    <cust:action name="showMoveAction" value="false"/>
  </cust:actions>

</cust:customizableComponentsSecurity>
```

For more information about applying action-level restrictions, see Section 5.8.4, "Applying Action-Level Restrictions on Show Detail Component Actions."

custom-actions

Use the custom-actions element to define custom actions that must be displayed along with the other Show Detail Frame actions. You must add only one custom-actions element inside the customizableComponentsSecurity section. Within this you can add any number of custom-action elements to define individual custom actions.

Note: Alternatively, you can add a custom-actions element inside an adf-config-child instead of customizableComponentsSecurity section.

For more information about custom actions, see Section 4.2.14, "How to Enable Custom Actions On a Show Detail Frame Enclosing a Task Flow: Example."

B.2.2.6 mds-config

When defining type-level customization restrictions on components, use the mds-config element to register a standalone XML file describing the restrictions to be applied on specific components. For information about applying type-level restrictions, see Section 5.8.1.1, "How to Define Type-Level Customization Policies."

You can add the mds-config element as follows:

```
<mds-config xmlns="http://xmlns.oracle.com/mds/config">
  <type-config>
    <standalone-definitions>
      <file>Directory_Name/standalone.xml</file>
    </standalone-definitions>
  </type-config>
</mds-config>
```

B.2.3 web.xml

The web.xml file is a Java EE standard descriptor that contains details about Web applications. When you add a Page Customizable component to your page, the web.xml file available in the <*Application_Root*>\<*Project_Name*>\public_html\WEB-INF directory is updated to enable change persistence among other settings.

You must update web.xml further when creating an application that uses multiple customization layers and when enabling Oracle Composer sandbox. The following sections describe the web.xml configurations specific to Oracle Composer.

B.2.3.1 CHANGE_PERSISTENCE Context Parameter

The CHANGE_PERSISTENCE context parameter in the application's web.xml file specifies how end user customizations must be persisted. To enable changes in an Oracle Composer-enabled page to be persisted in MDS, the CHANGE_PERSISTENCE context parameter must be set to MDSDocumentChangeManager, as shown in the following example:

```
<context-param>
  <param-name>org.apache.myfaces.trinidad.CHANGE_PERSISTENCE</param-name>
  <param-value>oracle.adf.view.rich.change.MDSDocumentChangeManager</param-value>
</context-param>
```

This context parameter registers the MDSDocumentChangeManager class to be used to ensure persistence to the MDS metadata store. Oracle Composer will work only if this context parameter is set to `MDSDocumentChangeManager`. This configuration happens automatically when you add a `Page Customizable` component to a page in a new WebCenter application. However, in an existing ADF application, this context parameter may be set to some other value, for example `session` or `oracle.adf.view.rich.change.FilteredPersistenceChangeManager`. Therefore, when you add Oracle Composer components to such an application, you must ensure that the `CHANGE_PERSISTENCE` parameter is set to `MDSDocumentChangeManager`.

B.2.3.2 WebCenterComposerFilter

Use `WebCenterComposerFilter` in the `web.xml` file to register Oracle Composer's `ComposerSessionOptionsFactory` with Oracle ADF for every HTTP request. The request is then handled depending on your implementation of `ComposerSessionOptionsFactory`. This filter can be defined as shown in the following example:

```
<filter-mapping>
<filter-name>WebCenterComposerFilter</filter-name>
  <url-pattern>/faces/*</url-pattern>
  <dispatcher>FORWARD</dispatcher>
  <dispatcher>REQUEST</dispatcher>
</filter-mapping>
```

You must use `WebCenterComposerFilter` if you are performing the following tasks:

- Creating customization layers. For information, see Section 5.6.1, "Adding Customization Layers to View and Edit Modes: Example."
- Enabling sandbox creation in Oracle Compose. For information, see Section 5.7.1, "How to Enable Oracle Composer Sandbox Creation."

B.3 Oracle Composer Default Add-Ons and Property Panels

This section lists the add-ons and property panels available by default in Oracle Composer.

Add-Ons

This section lists the add-ons available by default and provides the names with which these add-ons are registered in the `adf-config.xml` file. These add-on names are useful if you choose to create custom add-ons or exclude default add-ons in Oracle Composer. If you create custom add-ons in your application, while registering those add-ons in `adf-config.xml`, you must also include the entries for the default add-ons. Without this, the default add-ons will not display in Oracle Composer. For information about add-ons, see Section 5.2, "Creating Oracle Composer Add-Ons."

- **Page Properties**

  Used to edit page properties and create page parameters.

  This add-on is registered with the name `oracle.adf.pageeditor.addonpanels.page-settings` in the `adf-config.xml` file.

- **Reset Page**

  Used to reset page customizations.

This add-on is registered with the name `oracle.adf.pageeditor.addonpanels.page-reset` in the `adf-config.xml` file.

Property Panels

This section lists the property panels available by default in the Component Properties and Page Properties dialogs and provides the names with which these panels are registered in the `pe_ext.xml` file. These names are useful if you choose to override the default panels with custom property panels, or hide some or all of the default panels. For information about creating property panels, see Section 5.3, "Creating Custom Property Panels."

- **Display Options**

  Used to define display-related behavior of a component. This panel is displayed in the Component Properties dialog for all components.

  This panel is referenced with the name `oracle.adf.pageeditor.pane.generic-property-inspector` in the `pe_ext.xml` file.

- **Style**

  Used to define the appearance of the component instance. This panel is displayed in the Component Properties dialog for all components.

  This panel is referenced with the name `oracle.adf.pageeditor.pane.inline-style-editor` in the `pe_ext.xml` file.

- **Content Style**

  Used to define the appearance of content inside a component instance. This panel is displayed in the Component Properties dialog for all components.

  This panel is referenced with the name `oracle.adf.pageeditor.pane.content-style-editor` in the `pe_ext.xml` file.

- **Events**

  Used to wire a contextual event to an action handler to enable the passing of values from a producer component to a consumer component when the event is triggered on the producer. This panel displays in the Component Properties dialog for all components.

  This panel is referenced with the name `oracle.adf.pageeditor.pane.events` in the `pe_ext.xml` file.

- **Region Parameters**

  Used to define parameters for the task flow region. This panel displays in the Component Properties dialog only for task flow regions.

  This panel is referenced with the name `oracle.adf.pageeditor.pane.region-param` in the `pe_ext.xml` file.

- **Portlet Paramters**

  Used to define portlet parameters. This panel displays in the Component Properties dialog only for portlets.

  This panel is referenced with the name `oracle.adf.pageeditor.pane.portlet-param` in the `pe_ext.xml` file.

- **Layout Customizable**

  Used to define the layout by selecting from a set of predefined layouts. This panel displays in the Component Properties dialog only for `Layout Customizable` components.

  This panel is referenced with the name `oracle.adf.pageeditor.pane.layout-cust-prop` in the `pe_ext.xml` file.

- **Page Parameters**

  Used to define the page parameters that can be wired to components on the page. This panel displays in the Page Properties dialog.

  This panel is referenced with the name `oracle.adf.pageeditor.pane.inline-style-editor` in the `pe_ext.xml` file.

- **Page Security**

  Used to define page permissions. This panel displays in the Page Properties dialog for secured application pages.

  This panel is referenced with the name `oracle.webcenter.page.pane.page-sec` in the `pe_ext.xml` file.

B.4 Oracle Composer Help Topic IDs

Help topics for Oracle Composer panels and dialogs are available in the `pages_cs.jar` file on the Oracle WebCenter Suite 11g Demonstrations and Samples page on the Oracle Technology Network (OTN) at:

`http://webcenter.oracle.com`

To integrate this help in the Oracle Composer UI in your custom WebCenter application, then see Table B–17 for a description of the context-sensitive help topics available in `pages_cs.jar`.

To integrate help using the JAR file, you must have created a help provider. For information about displaying help for components and creating a help provider, see the section titled "Displaying Help for Components" in the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*.

Alternatively, you can extract the raw HTML files from `pages_cs.jar` and hook up those files directly from Oracle Composer panels or dialogs. However, in this case the reference links within these files will not work and must be deleted.

Table B–17 Oracle Composer Runtime Help Topics

Topic ID to be Used	Corresponding Oracle Composer UI
pages_cs_pgeditmodesv	Source view of a page in Edit mode
pages_cs_pgeditmodedv	Design view of a page in Edit mode
pages_cs_cpparams	Parameters tab in the Component Properties dialog
pages_cs_cpdisplayopts	Display Options tab in the Component Properties dialog
pages_cs_cpstyle	Style tab in the Component Properties dialog

Table B–17 (Cont.) Oracle Composer Runtime Help Topics

Topic ID to be Used	Corresponding Oracle Composer UI
pages_cs_cpcontstyle	Content Style tab in the Component Properties dialog
pages_cs_othercss	Other CSS option on the Style tab in the Component Properties dialog
pages_cs_cpevents	Events tab in the Component Properties dialog
pages_cs_compcat	Catalog dialog
pages_cs_ppgen	Display Options tab in the Page Properties dialog
pages_cs_ppsecurity	Security tab in the Component Properties dialog
pages_cs_ pppgparams	Parameters tab in the Component Properties dialog
pages_cs_ cplayoutprops	Layout Customizable tab in the Component Properties dialog for a `Layout Customizable` component

B.5 Oracle Composer Components Style-Specific Properties

This section includes a series of tables that describe the style selectors associated with Oracle Composer components. Additionally, it describes how to use the `background` property to choose one of three skin-defined looks for these components.

This section includes the following topics:

- Section B.5.1, "Style Selectors for Oracle Composer Components"
- Section B.5.2, "Style Attributes"

B.5.1 Style Selectors for Oracle Composer Components

Use a style selector to describe an element's appearance by identifying the element and defining styles for it.

B.5.1.1 Global Style Selectors

Use global style selectors to define styles for multiple components within the application. Table B–18 lists and describes the global style selectors relevant to Oracle Composer components.

Table B–18 Global Style Selectors

Style Selector	Description
`.ComposerDark:alias`	Specifies the default color for toolbars, headers, and so on in Oracle Composer, with a color scheme of *dark*.
`.PEClickableImageAnchor:alias`	An alias for rendering a clickable icon in a table cell. Specifies the CSS properties needed to display an image which is clickable and which has inline-mode display. For example, you can specify this as a common property for all action icons on a `Show Detail Frame` header. By default, this alias is included in style selectors that use 16x16 images and have to display inline background-images for both IE and Firefox. To customize, you can include this alias in your selector and override its properties. For example, you can change font-size and padding-right for an image with different height and width, or you can set "display: block" with height, width attributes to display in block mode.

B.5.1.2 Page Customizable Style Selectors

Use the style selectors in Table B–19 to skin `Page Customizable` components.

Table B–19 Page Customizable Style Selectors

Style Selector	Description
`af|pageCustomizable`	Specifies the default size of a `Page Customizable` component when not being stretched by its parent container.
`af|pageCustomizable af|toolbox.ComposerToolbox`	Specifies the style for the main dark gray toolbar displayed in Edit mode. This toolbar contains a message about the page being edited and the various buttons.
`af|pageCustomizable af|toolbox.ComposerToolbox af|panelGroupLayout.ComposerConcurrency`	Specifies the style for the concurrency toolbar that appears below the main Oracle Compoesr toolbar when two or more users are editing a page at the same time in the same customization layer.
`af|pageCustomizable af|toolbox.ComposerToolbox af|menuBar af|menu::bar-item-text`	Specifies the style for the text on the View menu.
`af|pageCustomizable af|toolbox.ComposerToolbox af|menuBar af|menu::bar-item:highlighted`	Specifies the style for the View menu when the mouse hovers over it.
`af|pageCustomizable af|toolbox.ComposerToolbox af|menuBar af|menu::bar-item:depressed`	Specifies the style for the View menu when it is clicked.
`af|pageCustomizable af|toolbox.ComposerToolbox af|menuBar af|menu::bar-item-open-icon-style`	Specifies the style for the dropdown icon appearing on the View menu.
`af|pageCustomizable af|toolbox.ComposerToolbox af|menuBar af|menu:depressed af|menu::bar-item-open-icon-style`	Specifies the style for the dropdown icon when it is clicked on the View menu.

Table B–19 (Cont.) Page Customizable Style Selectors

Style Selector	Description
`af|pageCustomizable af|toolbox.ComposerToolbox af|menuBar af|menu:highlighted af|menu::bar-item-open-icon-style`	Specifies the style for the dropdown icon when the mouse hovers over it.
`af|pageCustomizable af|toolbox.ComposerToolbox af|commandToolbarButton::text` `af|pageCustomizable af|toolbox.ComposerToolbox af|commandToolbarButton:hover` `af|pageCustomizable af|toolbox.ComposerToolbox af|commandToolbarButton:depressed`	Specify the styles for the Oracle Composer toolbar buttons like Page Properties and Reset Page.
`af|dialog.ComposerDialog::content-start` `af|dialog.ComposerDialog::content-start:rtl` `af|dialog.ComposerDialog::content` `af|dialog.ComposerDialog::content:rtl` `af|dialog.ComposerDialog::content-end`	Specify the styles for the dialogs invoked using Oracle Composer toolbar buttons. Available alias is :rtl, which can be used when the component must be rendered right-to-left.

B.5.1.3 Layout Customizable Style Selectors

Use the style selectors in Table B–20 to skin `Layout Customizable` components.

Table B–20 Layout Customizable Style Selectors

Style Selector	Description
`af|layoutCustomizable::menu:edit`	Specifies the style for the Change Layout button in Edit mode.
`af|layoutCustomizable::menu:edit af|commandImageLink::text` `af|layoutCustomizable::menu:edit af|commandImageLink::text:hover`	Specify the styles for the text on the Change Layout button in Edit mode.

B.5.1.4 Panel Customizable Style Selectors

Use the style selectors listed in Table B–21 to skin `Panel Customizable` components.

In Table B–21, you may note that some style selectors have three color-scheme selections: *light*, *medium*, and *dark*. Using these you can define three distinct looks in your CSS and specify which one to use through the `background` property in the Oracle JDeveloper Property Inspector. Depending on which value is specified for a component instance's `background` property, the skin will apply the relevant style.

Table B–21 Panel Customizable Style Selectors

Style Selector	Description
`af|panelCustomizable`	Specifies the style for the root element of the component.
`af|panelCustomizable.PEStretched`	Specifies the style for the component, when it is stretched.
`af|panelCustomizable:edit`	Specifies the style for the container in Edit mode.
`af|panelCustomizable:edit:drop-target`	Specifies the style for this component when you drag another component on the page and hover over this component to drop it in.
`af|panelCustomizable::edit-mode-content-style`	Specifies the style for the area containing the edit and delete icons.

Table B–21 (Cont.) Panel Customizable Style Selectors

Style Selector	Description

`af|panelCustomizable::add-icon-style`
`af|panelCustomizable::add-icon-style:active`
`af|panelCustomizable::add-icon-style:hover`
`af|panelCustomizable::add-icon-style:rtl`
	Specify the styles for the Add Content button on the component.

`af|panelCustomizable::edit-icon-style`
`af|panelCustomizable::edit-icon-style:active`
`af|panelCustomizable::edit-icon-style:hover`
	Specify the styles for the edit icon on the component.

`af|panelCustomizable::delete-icon-style`
`af|panelCustomizable::delete-icon-style:active`
`af|panelCustomizable::delete-icon-style:hover`
	Specify the styles for the delete icon on the component.

`.PanelCustomizableDropProxy`	Specifies the style for the placeholder area inside this component when you drag another component on the page and hover over this component to drop it in.

B.5.1.5 Show Detail Frame Style Selectors

Use the style selectors listed in Table B–22 to skin portlets and `Show Detail Frame` components.

Note: In WebCenter applications, each portlet is rendered with portlet *chrome* (see Section 27.1.1, "Portlet Anatomy"). Portlet chrome shares the same chrome rendering mechanism as a `Show Detail Frame` component. Thus, the style and icon selectors that apply to `Show Detail Frame` also apply to portlet chrome. In other words, in addition to defining styles for `Show Detail Frame` components, use `Show Detail Frame` style and icon selectors to define styles for portlets.

In Table B–22, some `Show Detail Frame` style selectors have *light*, *medium*, and *dark* color scheme options. Using these you can define three distinct looks in your CSS and specify which one to use through the `background` property in the Oracle JDeveloper Property Inspector. Depending on which value is specified for a component instance's `background` property, the skin will apply the relevant style.

Table B–22 Show Detail Frame Style Selectors

Style Selector	Description
`af|showDetailFrame`	Specifies the style for the root element of the component.

`af|showDetailFrame::container:light`
`af|showDetailFrame::container:medium`
`af|showDetailFrame::container:dark`
	Specify the styles for the element containing the contents within the component.

`af|showDetailFrame.p_AFStretched`	Specifies the style for the root element when the component is stretched.

`af|showDetailFrame::content`
`af|showDetailFrame::content:ight`
`af|showDetailFrame::content:medium`
`af|showDetailFrame::content:dark`
	Specify the styles to render for the content region of the component.

Table B–22 (Cont.) Show Detail Frame Style Selectors

Style Selector	Description

`af|showDetailFrame::header`
`af|showDetailFrame::header:light`
`af|showDetailFrame::header:medium`
`af|showDetailFrame::header:dark`
	Specify the styles for the header element. This element surrounds the header text, icon, and actions regions.

`af|showDetailFrame::header-center`
`af|showDetailFrame::header-center:medium`
`af|showDetailFrame::header-center:dark`
	Specify the styles for the region on the component header containing the icons, title, and actions menu.

`af|showDetailFrame::header-start`
`af|showDetailFrame::header-end:rtl`
`af|showDetailFrame::header-start:dark`
`af|showDetailFrame::header-end:dark:rtl`
	Specify the styles for the small cap-like area at the start of the header. Available alias is :rtl, which can be used when the component must be rendered right-to-left.

`af|showDetailFrame::header-end,`
`af|showDetailFrame::header-start:rtl`
`af|showDetailFrame::header-end:dark`
`af|showDetailFrame::header-start:dark:rtl`
	Specify the styles for the small cap-like area at the end of the header. Available alias is :rtl, which can be used when the component must be rendered right-to-left.

`af|showDetailFrame::icon-style`	Specifies the style for the icon on the `Show Detail Frame` header, if the `icon` attribute is set on the component.
`af|showDetailFrame::header-text` `af|showDetailFrame::header-text:rtl`	Specifies the style for the title available on the component header. Available alias is :rtl, which can be used when the component must be rendered right-to-left.
`af|showDetailFrame::header-actions`	Specifies the style for icons like delete and actions menu on the component header.

`af|showDetailFrame::toolbar-container`
`af|showDetailFrame::toolbar-container:light`
`af|showDetailFrame::toolbar-container:medium`
`af|showDetailFrame::toolbar-container:dark`
	Specify styles for the container element of the floating toolbar. Available aliases are :light, :medium, and :dark.

`af|showDetailFrame::collapse-icon-style`
`af|showDetailFrame::collapse-icon-style:active`
`af|showDetailFrame::collapse-icon-style:hover`
	Specify styles for the collapse icon on the component header. Available aliases are :active and :hover that you can use to specify styles for the collapse icon when clicked or when the mouse hovers over it.

`af|showDetailFrame::disclose-icon-style`
`af|showDetailFrame::disclose-icon-style:active`
`af|showDetailFrame::disclose-icon-style:hover`
	Specify the styles for the disclose icon on the component header.

`af|showDetailFrame::actionmenu-icon-style`
`af|showDetailFrame::actionmenu-icon-style:active`
`af|showDetailFrame::actionmenu-icon-style:hover`
	Specify the styles for the action menu icon. This style has a background-image that you can override.

`af|showDetailFrame::remove-icon-style`
`af|showDetailFrame::remove-icon-style:active`
`af|showDetailFrame::remove-icon-style:hover`
	Specify the styles for the remove icon.

`af|showDetailFrame::edit-mode-content-style` `af|showDetailFrame::edit-mode-content-style:light`	Specify the styles for the area that contains the edit and remove icons.

`af|showDetailFrame::edit-mode-edit-icon-style`
`af|showDetailFrame::edit-mode-edit-icon-style:active`
`af|showDetailFrame::edit-mode-edit-icon-style:hover`
	Specify the styles for the edit icon on the component header.

Table B–22 (Cont.) Show Detail Frame Style Selectors

Style Selector	Description
af|showDetailFrame::edit-mode-remove-icon-style af|showDetailFrame::edit-mode-remove-icon-style:active af|showDetailFrame::edit-mode-remove-icon-style:hover	Specify the styles for the remove icon on the component header.
af|showDetailFrame::edit-text-link af|showDetailFrame::edit-text-link:rtl af|showDetailFrame::edit-text-link:hover	Specify the styles for the Edit Text link in Edit mode when the component includes a Rich Text Editor component as a child.
af|showDetailFrame::resize af|showDetailFrame::resize:rtl	Specifies the style for the resize handler on the component. Available alias is :rtl, which can be used when component must be rendered right-to-left.

B.5.2 Style Attributes

Table B–23 provides a quick reference for the style attributes available for WebCenter Customizable Components and Oracle Composer components through the Oracle JDeveloper Property Inspector. Note that portlets also use these style attributes.

For detailed information about style properties, see the CSS Specifications page at:

`http://www.w3.org/TR/CSS21/`

Table B–23 Style Attributes of Oracle Composer and WebCenter Customizable Components

Attribute	Description
`background-color`	The background color of the component. Choose from a list of colors, or click the Edit icon to select from a color palette.
`background-image`	The image that is displayed in the component background. Select to inherit from a parent component or to display no image, or click the Edit icon to select an image.
`background-repeat`	Specify whether and how the background image should repeat. Choose from: ■ `<none>`: Forgo repeating the image. ■ `no-repeat`: Forgo repeating the image. ■ `repeat`: Repeat the image to fill the container. ■ `repeat-x`: Repeat the image horizontally but not vertically. ■ `repeat-y`: Repeat the image vertically but not horizontally. ■ `inherit`: Inherit this setting from a parent component.
`border-color`	The color of the border that surrounds the component. Choose from a list of colors, or click the Edit icon to select from a color palette.

Table B–23 (Cont.) Style Attributes of Oracle Composer and WebCenter Customizable Components

Attribute	Description
`border-style`	The style of border to draw around the component. Choose from: ■ `<none>`: Apply no style to the border. ■ `none`: Do not display a border (use border-style:none on the `ContentStyle` property to turn off portlet. borders. Using this attribute with `InlineStyle` does not turn off portlet borders.) ■ `hidden`: Hide the border. ■ `dotted`: Display the border as a line of dots. ■ `double`: Display the border as a double line. ■ `groove`: Display the border as a grooved line. ■ `ridge`: Display the border as a ridged line. ■ `inset`: Display the border as a concave line. ■ `outset`: Display the border as a convex line. ■ `dashed`: Display the border as a line of dashes. ■ `solid`: Display the border as a solid line. ■ `inherit`: Inherit the border style from a parent component.
`border-width`	The thickness of the component border. Select `thick`, `medium`, `thin`, or `inherit`, to inherit border width from a parent component. Alternatively, click the Edit icon and define a thickness in your preferred unit of measurement.
`color`	The color of component text. Select from a list, or click the Edit icon to select from a color palette.
`font-family`	The font family (such as Arial, Helvetica, sans-serif) for component text. Enter this value manually.
`font-size`	The size of component text. Choose from: ■ `Choose Length`: Specify an absolute value for font-size in your preferred unit of measurement. ■ `Choose Percentage`: Specify font size as a percentage of normal text size defined for the browser. ■ `inherit`: Inherit font size from a parent component. ■ `large`: Select the next size larger than the font associated with the parent component. ■ `larger`: Increase the font size to larger than the font. associated with the parent component (`larger` is larger than `large`). ■ `medium`: Set the font size to the default font size of the parent component. ■ `small`: Set the font size to smaller than the default font size of the parent component. ■ `smaller`: Set the font size to smaller than the default font size of the parent component (`smaller` is smaller than `small`).
`font-style`	Specifies the font style of the component text. Choose from: ■ `<none>`: Apply no style to the font. ■ `normal`: The font is not italic. ■ `italic`: The font is italic. ■ `oblique`: The font is slanted to look italic, though the font family may not have a formal italic member. ■ `inherit`: Inherit font style from a parent component.

Table B–23 (Cont.) Style Attributes of Oracle Composer and WebCenter Customizable Components

Attribute	Description
`font-weight`	Set the thickness of component text. Choose from: ■ `<none>`: Specify no weight for component font. ■ `normal`: Apply same weight at the parent component's default font weight. ■ `bold`: Set component text in bold. ■ `bolder`: Set component text darker than bold. ■ `light`: Set component text lighter than normal. ■ `lighter`: Set component text lighter than light. ■ `100-900`: Specify a value for text darkness—`100` is lightest and `900` is darkest—`400` is normal weight; `700` is bold. ■ `inherit`: Inherit font weight from a parent component.
`height`	The spacing to apply between lines of continuous text (also known as *leading*). Specify a height by choosing from: ■ `Choose Length`: Specify an absolute value in your preferred unit of measurement. ■ `Choose Percentage`: Specify a value as a percentage of the value set for a parent component. ■ `auto`: Set line height automatically, usually about 2 points larger than font size. ■ `inherit`: Inherit line height from the parent component.
`list-style-image`	Specify an image to use as an indicator of a list item. Specify an image instead of defining a `list-style-type`. Choose from: ■ `inherit`: Inherit a list style image from a parent component. ■ `none`: Use no image as an indicator of a list item. Or click the Edit icon and select an image.
`list-style-type`	The type of indicator to use for items on a list. Use this instead of specifying a image through `list-style-image`. Select from among many styles offered on the list, including `<none>`, which means the browser's default style is used, and `inherit`, which means the `list-style-type` is inherited from a parent component.
`margin`	The border of space surrounding component content. Choose from: ■ `Choose Length`: Specify an absolute value for all margins. ■ `Choose Percentage`: Set a value as a percentage of the margin of a parent component. ■ `inherit`: Inherit margin value from a parent component. ■ `auto`: Set the value automatically according to browser defaults.
`padding`	The amount of space between the component and its margin or, if there is a border, between the component and its border. Choose from: ■ `Choose Length`: Specify an absolute value in your preferred unit of measurement. ■ `Choose Percentage`: Specify the value as a percentage of the padding set for a parent component. ■ `inherit`: Inherit the value from a parent component.

Table B–23 (Cont.) Style Attributes of Oracle Composer and WebCenter Customizable Components

Attribute	Description
`text-align`	The horizontal alignment of component text. Choose from: ■ `<none>`: Use browser default. ■ `left` ■ `right` ■ `center` ■ `justify`: Align text so that all lines start and end at the same point left and right. ■ `auto`: Apply a value automatically, either left or right. ■ `inherit`: Inherit alignment from a parent component.
`text-decoration`	Decorative value to apply to component text. Choose from: ■ `<none>`: Use the browser default for hyperlinks; otherwise, no text decoration. ■ `none`: No text decoration, including no underline for hyperlinks. ■ `underline`: Underline all component text. ■ `overline`: Draw a line over all component text. ■ `line-through`: Strike out all component text. ■ `blink`: Make component text blink. ■ `inherit`: Inherit text decoration from a parent component.
`vertical-align`	Vertical alignment of component text. Choose from: ■ `Choose Length`: Specify an absolute value in your preferred unit of measurement. ■ `Choose Percentage`: Specify a value as a percentage of the value set for a parent component. ■ `inherit`: Inherit the value from a parent component. ■ `baseline`: Set the text on the baseline of text row. ■ `middle`: Set the text in the middle of text row. ■ `text-bottom`: Set the text on the bottom of text row, lower than baseline. ■ `text-top`: Set the text at the top of the text row, above the baseline. ■ `sub`: Set the text as subscript. ■ `super`: Set the text as superscript.
`width`	The spacing between text. Specify a width by choosing from: ■ `Choose Length`: Specify an absolute value in your preferred unit of measurement. ■ `Choose Percentage`: Specify a value as a percentage of the value set for a parent component. ■ `auto`: Set line height automatically, usually about 2 points larger than font size. ■ `inherit`: Inherit line height from the parent component.

B.6 Customizable Components (HTML) Properties

Customizable Components (HTML) are similar in function to customizable components from the 10.1.3.2.0 release and can be used to add more customizable components to your migrated 10.1.3.2.0 application page. These customizable components are available from the Customizable Components (HTML) tag library in JDeveloper. When you add customizable components to the page, default values are assigned to certain attributes and you can define values for the remaining attributes as required.

Customizable components enable users to minimize or maximize, hide or show, or move any component on the page at runtime.

B.6.1 Panel Customizable (HTML) Component

Use the `Panel Customizable (HTML)` component as a container for page content that can be customized at runtime. Components added inside a `Panel Customizable (HTML)` can be maximized, minimized, or rearranged.

Use this component only to perform runtime customizations on child components. If you just want a container to arrange components at design time, then it is recommended that you use an ADF Faces container like `Panel Group Layout`.

Table B–24 describes the attributes of a `Panel Customizable (HTML)` component.

Table B–24 Attributes of a Panel Customizable (HTML) Component

Attribute	Type	Supports EL?	Description
Common Attributes			
id	String	No	The identifier for the component. The identifier must follow a subset of the syntax allowed in HTML: ■ Must not be a zero-length String. First character must be an ASCII letter (A-Za-z) or an underscore ('_'). ■ Subsequent characters must be an ASCII letter or digit (A-Za-z0-9), an underscore ('_'), or a dash ('-').
rendered	boolean	Yes	Specifies whether the component is rendered. When set to `false`, no output will be delivered for this component (the component will not in any way be rendered, and cannot be made visible on the client). The default value is `true`.

Table B–24 (Cont.) Attributes of a Panel Customizable (HTML) Component

Attribute	Type	Supports EL?	Description
layout	List of values `horizontal/vertical`	Yes	Specifies how the children of the `Panel Customizable` (HTML) must be laid out. The default value is `vertical`. Depending on the value of the layout attribute, child components are laid out as follows: If you select `vertical`, then the child components are displayed one below the other and can be moved either up or down within the layout. If you select horizontal, then the child components are displayed adjacent to each other and can be moved either to the left or right within the layout.
Style Attributes			
styleClass	String	Yes	A CSS style class to use for this component. The style class can be defined in your JSPX page or in a skinning CSS file, for example.
inlineStyle	String	Yes	The CSS styles to use for this component. This is intended for basic style changes. The `inlineStyle` is a set of CSS styles that are applied to the root DOM element of the component. If the `inlineStyle`'s CSS properties do not affect the DOM element you want affected, then you will have to create a skin and use the skinning keys which are meant to target particular DOM elements, like `::label` or `::icon-style`.
Advanced Attributes			

binding	String	Yes	An EL reference that will store the component instance on a bean. This can be used to give programmatic access to a component from a backing bean, or to move creation of the component to a backing bean.
			For example:
			`binding="#{yourManagedBean.`
			`Binding}"`

B.6.2 Show Detail Frame (HTML) Component

Add `Show Detail Frame (HTML)` components inside `Panel Customizable (HTML)` components to enable runtime personalizations like maximizing, moving, minimizing, restoring, and deleting child components. You can perform these customizations only if you add `Show Detail Frame (HTML)` components inside `Panel Customizable (HTML)` components.

Table B–25 describes the attributes of a `Show Detail Frame (HTML)` component.

Table B–25 Attributes of a Show Detail Frame (HTML) Component

Attribute	Type	Supports EL?	Description
Common Attributes			
id	String	No	The identifier for the component. The identifier must follow a subset of the syntax allowed in HTML: ■ Must not be a zero-length String. First character must be an ASCII letter (A-Za-z) or an underscore ('_'). ■ Subsequent characters must be an ASCII letter or digit (A-Za-z0-9), an underscore ('_'), or a dash ('-').
rendered	boolean	Yes	Specifies whether the component is rendered. When set to `false`, no output will be delivered for this component (the component will not in any way be rendered, and cannot be made visible on the client). The default value is `true`.
text	String	Yes	A title for the `Show Detail Frame` (HTML) component.

icon	String	Yes	If you decide to add an icon on the header of the `Show Detail Frame` component, then this specifies the URI for the image to be used.
			For example:
			`icon="http://source-pc/images/accessability.gif"`
			Note that an image that is stored at the document root does not require a full path. For example:
			`icon="detail.gif"`

Appearance Attributes			

Table B–25 (Cont.) Attributes of a Show Detail Frame (HTML) Component

Attribute	**Type**	**Supports EL?**	**Description**
shortDesc	String	Yes	The short description of the component. This text is commonly used by user agents to display tooltip help text, in which case the behavior for the tooltip is controlled by the user agent, for example, Firefox 2 truncates long tooltips. For form components, the `shortDesc` is displayed in a note window.
background	List of values `light/medium/dark`	Yes	Working with the skin CSS, provides a means of applying a different look and feel for this `Show Detail Frame (HTML)` instance. The default value is `medium`.
displayHeader	boolean	Yes	Indicates whether the header of the `Show Detail Frame` (HTML) is displayed. The default value is `true`. If you choose to set `displayHeader` to `false` and if you have exposed some actions on the component, then a toolbar is displayed when you move the mouse over the component area. The toolbar contains a drop down icon, which displays a menu of available options. This toolbar will display only if there are actions available on the component. The toolbar display is also affected by the `isSeededInteractionAvailable` attribute. As the default value for `isSeededInteractionAvailable` is `false`, the toolbar is not displayed. It can be displayed by setting `isSeededInteractionAvailable` to `true`.
expansionMode	List of values `maximized/minimized/normal`	Yes	The default state of the `Show Detail Frame (HTML).` The default display mode is `normal`.
Actions Attributes			
showMoveAction	List of values `menu/none`	Yes	Specifies whether the Move action must be displayed in the Actions menu. The default value is `menu`.
showMinimizeAction	List of values `chrome/none`	Yes	Specifies whether the minimize action must be displayed on the header. The default is `chrome`.

Table B–25 (Cont.) Attributes of a Show Detail Frame (HTML) Component

Attribute	Type	Supports EL?	Description
showMaximizeAction	List of values `menu/none`	Yes	Specifies whether the minimize action must be displayed on the header. The default value is `menu`.
Style Attributes			
contentStyle	String	Yes	The CSS style to apply to the `Show Detail Frame (HTML)` content area. Manually enter any style in compliance with CSS version 2.0 or later.
styleClass	String	Yes	A CSS style class to use for this component. The style class can be defined in your JSPX page or in a skinning CSS file, for example.
inlineStyle	String	Yes	The CSS styles to use for this component. This is intended for basic style changes. The `inlineStyle` is a set of CSS styles that are applied to the root DOM element of the component. If the `inlineStyle`'s CSS properties do not affect the DOM element you want affected, then you will have to create a skin and use the skinning keys which are meant to target particular DOM elements, like `::label` or `::icon-style`.
Behavior Attributes			
partialTriggers	String	Yes	The IDs of the components that should trigger a partial update. This component will listen on the trigger components. If a trigger component receives an event that will cause it to update in some way, this component will request to be updated too. Identifiers are relative to the source component (this component), and must account for NamingContainers. If your component is inside of a naming container, you can use a single colon to start the search from the root of the page, or multiple colons to move up through the NamingContainers - "`::`" will pop out of the component's naming container (or itself if the component is a naming container) and begin the search from there, "`:::`" will pop out of two naming containers (including itself if the component is a naming container) and begin the search from there.

Table B–25 (Cont.) Attributes of a Show Detail Frame (HTML) Component

Attribute	Type	Supports EL?	Description
disclosureListener	`javax.el.MethodExpression`	Yes	A method reference to a disclosure listener. A disclosure event is fired when the disclosure state changes.
Advanced Attributes			
binding	String	Yes	An EL reference that will store the component instance on a bean. This can be used to give programmatic access to a component from a backing bean, or to move creation of the component to a backing bean. For example: `binding="#{yourManagedBean.Binding}"`
attributeChangeListener	`javax.el.MethodExpression`	Yes	A method reference to an attribute change listener. Attribute change events are not delivered for any programmatic change to a property. They are only delivered when a renderer changes a property without the application's specific request. An example of an attribute change events might include the width of a column that supported client-side resizing.

The **Expression Builder** option available when setting these attributes enables you to bind the component instance to a managed bean property.

Show Detail Frame Facets

You can use the `Show Detail Frame (HTML)` facets to define and display custom actions on `Show Detail Frame (HTML)` components. Table B–26 describes the facets that enable you to display custom actions supported by the `Show Detail Frame (HTML)` component.

Table B–26 Show Detail Frame (HTML) Facets

Facet	Description
titleBarAction	Used if an action is to be associated with title of the `Show Detail Frame` (HTML) component.
additionalActions	Used if some additional actions are to be added to the list of actions available on the `Show Detail Frame (HTML)` header.

Oracle JDeveloper displays all facets available to the `Show Detail Frame (HTML)` component in the Structure window. However, only those that contain UI components appear activated.

B.6.3 Customizable Components (HTML) Style Selectors

The tables in this section describe the style selectors that you can use to skin WebCenter Customizable Components.

B.6.3.1 Panel Customizable (HTML) Style Selectors

Use the style selectors listed in Table B–27 to skin `Panel Customizable (HTML)` components. See Section B.6.3.4, "Icon Selectors" for icon selectors relevant to the `Panel Customizable` component.

Table B–27 Panel Customizable (HTML) Style Selectors

Style Selector	Description
`afh|panelCustomizable::no-header-content`	Specifies the style to render for all four component borders when the component header is turned off.

B.6.3.2 Show Detail Frame (HTML) Style Selectors

Use the style selectors listed in Table B–28 to skin the `Show Detail Frame (HTML)` components.

Note: In addition to defining styles for `Show Detail Frame` components, the `Show Detail Frame` style and icon selectors define styles for portlets.

Table B–28 Show Detail Frame (HTML) Style Selectors

Style Selector	Description

`afh|showDetailFrame::header-top-border-light`
`afh|showDetailFrame::header-top-border-medium`
`afh|showDetailFrame::header-top-border-dark`
	Specifies the style for the top and bottom border of a component header and, also the header background color.

`afh|showDetailFrame::container`	Specifies the style for the component's container.

`afh|showDetailFrame::header-light`
`afh|showDetailFrame::header-medium`
`afh|showDetailFrame::header-dark`
	Specifies the style for text in the component's header. The header is usually a banner of color that contains a title and links to menus and other types of actions.
	
	Define the header background color with the `af|showDetailFrame::header-top-border-[light,medium,dark]` style element. Use icon selectors to specify the icons to use in the component header and, also the shape of the header's left and right sides.

`afh|showDetailFrame::content-light`
`afh|showDetailFrame::content-medium`
`afh|showDetailFrame::content-dar`
	Specifies the style for the component's left, right, and bottom borders.

`afh|showDetailFrame::main-menu-container`	Specifies the style for the component's main menu container.
`afh|showDetailFrame::sub-menu-container`	Specifies the style for the component's submenu container.
`afh|showDetailFrame::menu-item`	Specifies the style for an individual item on the component's main menu.
`afh|showDetailFrame::menu-item:hover`	Specifies the style to render when a user pauses the mouse pointer over a component main menu item.
`afh|showDetailFrame::sub-menu-item`	Specifies the style for an individual item on the component's submenu.

Table B–28 (Cont.) Show Detail Frame (HTML) Style Selectors

Style Selector	Description
`afh|showDetailFrame::sub-menu-item:hover`	Specifies the style to render when a user pauses the mouse pointer over a component submenu item.
`afh|showDetailFrame::actions-image-separator`	Specifies the amount of padding to provide around the component's Actions, Minimize, and Restore icons. See also Section B.6.3.4, "Icon Selectors".
`afh|showDetailFrame::menu-item-separator`	Specifies the style for the line that separates a command or groups of commands on the component's Actions menu. In the default case, a separator appears to be a single thick line. This is achieved using `border-top` and `border-bottom` elements to style the separator. A user who creates a custom skin can style the separator differently. For example, a user can create a separator that displays as a rectangular bar with a colorful background.
`A.afh|showDetailFrame::title-clickable`	Specifies the style to render for the component's title when the title is a link.

`afh|showDetailFrame::no-header-content`
`afh|showDetailFrame::no-header-content-light`
`afh|showDetailFrame::no-header-content-medium`
`afh|showDetailFrame::no-header-content-dark`
	Specifies the style to render for all four component borders when the component header is turned off.

B.6.3.3 Property Keys

Use property keys to control the display of custom menu items and component action icons. Though you include property keys in your custom skin, they are not represented in the generated CSS that results from the skin.

To explain, skins go through a process that results in a generated CSS. In turn, the generated CSS is consumed by the application. Most style selectors for `Panel Customizable` and `Show Detail Frame` components are represented in the generated CSS. Property keys are the exception. Although they affect the application as much as any other component style selector, they are not represented in the final generated CSS.

Table B–29 lists and describes property keys relevant to `Panel Customizable` and `Show Detail Frame` components.

Table B–29 Property Keys of Panel Customizable and Show Detail Frame Components

Property Key	Description
`showDetailFrame {-ora-additional-actions-position-last:true}`	This property key positions additional actions relative to seeded actions on the component's Actions menu. Set to `false` to position additional actions before seeded actions. By default, additional actions are positioned after seeded actions. The default value is `true`.
`showDetailFrame {-ora-menu-icon-display:false}`	This property key controls the display of icons next to their related commands on a `Show Detail Frame` Actions menu. Set to `true` to display icons to the left of each action on the Actions menu. By default, no icons are displayed to the left of individual actions. The default value is `false`. For information about specifying the icons to use when this property key is set to `true`, see Section B.6.3.4, "Icon Selectors."

B.6.3.4 Icon Selectors

Icons are either displayed or not displayed depending on whether the component's `ora-menu-icon-display` property key is set to `true` or `false`. Property keys are described in Section B.6.3.3, "Property Keys."

Each icon selector has a *light*, *medium*, and *dark* scheme. Using these you can define three distinct looks in your CSS and specify which one to use through the `background` property in the Oracle JDeveloper Property Inspector. Depending on which value is specified for a component instance's `background` property, the skin will apply the relevant style.

For easy, error-free portability, store all application icons under the WebCenter application's root folder.

The selectors described in Table B–30 apply to the icons used with `Panel Customizable (HTML)` and `Show Detail Frame (HTML)` components.

Table B–30 Icon Selectors for WebCenter Customizable Components

Selector	Description

showDetailFrame
`showDetailFrame::light-ActionsIcon:alias`
`showDetailFrame::medium-ActionsIcon:alias`
`showDetailFrame::dark-ActionsIcon:alias`
panelCustomizable
`panelCustomizable::light-ActionsIcon:alias`
`panelCustomizable::medium-ActionsIcon:alias`
`panelCustomizable::dark-ActionsIcon:alias`
	This icon represents the Actions menu. The Actions menu lists the actions a user can perform on the component.
	In a WebCenter application, the Actions icon is rendered on the right corner of the component header.

showDetailFrame
`showDetailFrame::light-MinimizeIcon:alias`
`showDetailFrame::medium-MinimizeIcon:alias`
`showDetailFrame::dark-MinimizeIcon:alias`
panelCustomizable
`panelCustomizable::light-MinimizeIcon:alias`
`panelCustomizable::medium-MinimizeIcon:alias`
`panelCustomizable::dark-MinimizeIcon:alias`
	This icon represents the Minimize option. Minimize collapses the view of the component like a window shade.
	In a WebCenter application, the Minimize icon is rendered on the left side of the component header.
	See also, `showDetailFrame::light-ExpandIcon:alias`.

showDetailFrame
`showDetailFrame::light-MaximizeIcon:alias`
`showDetailFrame::medium-MaximizeIcon:alias`
`showDetailFrame::dark-MaximizeIcon:alias`
	This is applicable only for the `Show Detail Frame (HTML)` component. The rich variant of this component does not support the maximize action.
	This icon represents the Maximize option, which expands the `Show Detail Frame (HTML)` component to the dimensions of the `Panel Customizable (HTML)` component that contains it. Where multiple `Show Detail Frame (HTML)` components display in the same container, these are displaced while the maximized `Show Detail Frame (HTML)` remains maximized.
	In a WebCenter application, the Maximize icon is displayed to the left of the Maximize command on the component's Actions menu.

panelCustomizable
`panelCustomizable::light-MaximizeIcon:alias`
`panelCustomizable::medium-MaximizeIcon:alias`
`panelCustomizable::dark-MaximizeIcon:alias`
	This is applicable only for the `Panel Customizable (HTML)` component. The rich variant of this component does not povide any actions.
	This icon represents the Maximize option, which expands component display to the dimensions of the container. Where multiple components display in the same container, these are displaced by the maximized component.
	In a WebCenter application, the Maximize icon is displayed to the left of the Maximize command on the component's Actions menu.

showDetailFrame
`showDetailFrame::light-RestoreIcon:alias`
`showDetailFrame::medium-RestoreIcon:alias`
`showDetailFrame::dark-RestoreIcon:alias`
panelCustomizable
`panelCustomizable::light-RestoreIcon:alias`
`panelCustomizable::medium-RestoreIcon:alias`
`panelCustomizable::dark-RestoreIcon:alias`
	This icon represents the Restore option, which restores maximized views to their default display modes.
	In a WebCenter application, the Restore icon is rendered to the left of the Restore command on the component's Actions menu.

Table B–30 (Cont.) Icon Selectors for WebCenter Customizable Components

Selector	Description
showDetailFrame showDetailFrame::light-ExpandIcon:alias showDetailFrame::medium-ExpandIcon:alias showDetailFrame::dark-ExpandIcon:alias **panelCustomizable** panelCustomizable::light-ExpandIcon:alias panelCustomizable::medium-ExpandIcon:alias panelCustomizable::dark-ExpandIcon:alias	The Expand icon represents the action that expands a component that has been minimized. The Expand icon toggles with the Minimize icon. That is, when the component is minimized, the Expand icon is displayed; when the component is expanded, the Minimize icon is displayed. In a WebCenter application, the Expand icon is displayed on the left side of the component header.
showDetailFrame showDetailFrame::light-MoveIcon:alias showDetailFrame::medium-MoveIcon:alias showDetailFrame::dark-MoveIcon:alias **panelCustomizable** panelCustomizable::light-MoveIcon:alias panelCustomizable::medium-MoveIcon:alias panelCustomizable::dark-MoveIcon:alias	This icon represents the Move option, which enables rearrangement of a component's location in relation to the other components on the page. In a WebCenter application, the Move icon is displayed to the left of the Move command on the component's Actions menu.
showDetailFrame showDetailFrame::light-MoveLeftIcon:alias showDetailFrame::medium-MoveLeftIcon:alias showDetailFrame::dark-MoveLeftIcon:alias **panelCustomizable** panelCustomizable::light-MoveLeftIcon:alias panelCustomizable::medium-MoveLeftIcon:alias panelCustomizable::dark-MoveLeftIcon:alias	This icon represents the Move Left option on the component submenu. Move Left rearranges the component horizontally, one position closer to the left boundary of the page. For example, imagine three horizontally arranged components. You select Move Left on the right-most component. It becomes the middle component. In a WebCenter application, the Move Left icon is displayed to the left of the Move Left submenu item on the component's Actions menu.
showDetailFrame showDetailFrame::light-MoveRightIcon:alias showDetailFrame::medium-MoveRightIcon:alias showDetailFrame::dark-MoveRightIcon:alias **panelCustomizable** panelCustomizable::light-MoveRightIcon:alias panelCustomizable::medium-MoveRightIcon:alias panelCustomizable::dark-MoveRightIcon:alias	This icon represents the Move Right option on the component submenu. Move Right rearranges the component horizontally, one position closer to the right boundary of the page. For example, imagine three horizontally arranged components. You select Move Right on the left-most component. It becomes the middle component. In a WebCenter application, the Move Right icon is displayed to the left of the Move Right submenu item on the component's Actions menu.
showDetailFrame showDetailFrame::light-MoveUpIcon:alias showDetailFrame::medium-MoveUpIcon:alias showDetailFrame::dark-MoveUpIcon:alias **panelCustomizable** panelCustomizable::light-MoveUpIcon:alias panelCustomizable::medium-MoveUpIcon:alias panelCustomizable::dark-MoveUpIcon:alias	This icon represents the Move Up option on the component submenu. Move Up rearranges the component vertically in relation to the other components on the page. For example, imagine three vertically arranged components. You select Move Up on the middle component. It becomes the topmost component. In a WebCenter application, the Move Up icon is displayed to the left of the Move Up submenu item on the component's Actions menu.

Table B–30 (Cont.) Icon Selectors for WebCenter Customizable Components

Selector	Description
showDetailFrame showDetailFrame::light-MoveDownIcon:alias showDetailFrame::medium-MoveDownIcon:alias showDetailFrame::dark-MoveDownIcon:alias **panelCustomizable** panelCustomizable::light-MoveDownIcon:alias panelCustomizable::medium-MoveDownIcon:alias panelCustomizable::dark-MoveDownIcon:alias	This icon represents the Move Down option on the component submenu. Move Down rearranges the component vertically in relation to the other components on the page. For example, imagine three vertically arranged components. You select Move Down on the middle component. It becomes the bottommost component. In a WebCenter application, the Move Down icon is displayed to the left of the Move Down submenu item on the component's Actions menu.
showDetailFrame showDetailFrame::light-HeaderLeftIcon:alias showDetailFrame::medium-HeaderLeftIcon:alias showDetailFrame::dark-HeaderLeftIcon:alias **panelCustomizable** panelCustomizable::light-HeaderLeftIcon:alias panelCustomizable::medium-HeaderLeftIcon:alias panelCustomizable::dark-HeaderLeftIcon:alias	This icon provides an image for the top-left corner of the component header.
showDetailFrame showDetailFrame::light-HeaderRightIcon:alias showDetailFrame::medium-HeaderRightIcon:alias showDetailFrame::dark-HeaderRightIcon:alias **panelCustomizable** panelCustomizable::light-HeaderRightIcon:alias panelCustomizable::medium-HeaderRightIcon:alias panelCustomizable::dark-HeaderRightIcon:alias	This icon provides the image for the top-right corner of the component header.
showDetailFrame showDetailFrame::light-ToolbarLeftIcon:alias showDetailFrame::medium-ToolbarLeftIcon:alias showDetailFrame::dark-ToolbarLeftIcon:alias **panelCustomizable** panelCustomizable::light-ToolbarLeftIcon:alias panelCustomizable::medium-ToolbarLeftIcon:alias panelCustomizable::dark-ToolbarLeftIcon:alias	This icon provides the left portion of the component's FadeIn-FadeOut toolbar. The FadeIn-FadeOut toolbar comes into play when the `adfp:portlet` tag attribute `isSeededInteractionAvailable` is set to `true` and `displayHeader` is set to `false`. The toolbar contains the Actions menu that would otherwise be displayed on the header. To invoke the toolbar, users move their mouse over the component content area. If the page design is very simple, then the FadeIn-FadeOut toolbar may not display, even when `displayHeader` is set to `false` and `isSeededInteractionAvailable` is set to `true`.
showDetailFrame showDetailFrame::light-ToolbarRightIcon:alias showDetailFrame::medium-ToolbarRightIcon:alias showDetailFrame::dark-ToolbarRightIcon:alias **panelCustomizable** panelCustomizable::light-ToolbarRightIcon:alias panelCustomizable::medium-ToolbarRightIcon:alias panelCustomizable::dark-ToolbarRightIcon:alias	This icon provides the right portion of the component's FadeIn-FadeOut toolbar. See the description for `showDetailFrame::light-ToolbarLeftIcon:alias` for an explanation of the FadeIn-FadeOut toolbar.

Table B–30 (Cont.) Icon Selectors for WebCenter Customizable Components

Selector	Description

`showDetailFrame`
`showDetailFrame::light-ToolbarCenterIcon:alias`
`showDetailFrame::medium-ToolbarCenterIcon:alias`
`showDetailFrame::dark-ToolbarCenterIcon:alias`
`panelCustomizable`
`panelCustomizable::light-ToolbarCenterIcon:alias`
`panelCustomizable::medium-ToolbarCenterIcon:alias`
`panelCustomizable::dark-ToolbarCenterIcon:alias`
	This icon provides the center portion of the component's FadeIn-FadeOut toolbar.
	See the description for `showDetailFrame::light-ToolbarLeftIcon:alias` for an explanation of the FadeIn-FadeOut toolbar.

`showDetailFrame`
`showDetailFrame::light-EditIcon:alias`
`showDetailFrame::medium-EditIcon:alias`
`showDetailFrame::dark-EditIcon:alias`
`panelCustomizable`
`panelCustomizable::light-EditIcon:alias`
`panelCustomizable::medium-EditIcon:alias`
`panelCustomizable::dark-EditIcon:alias`
	This icon represents the Edit option on the component menu. In a WebCenter application, the Edit icon is displayed to the left of the Edit menu item on the component's Actions menu.

`showDetailFrame`
`showDetailFrame::light-HelpIcon:alias`
`showDetailFrame::medium-HelpIcon:alias`
`showDetailFrame::dark-HelpIcon:alias`
`panelCustomizable`
`panelCustomizable::light-HelpIcon:alias`
`panelCustomizable::medium-HelpIcon:alias`
`panelCustomizable::dark-HelpIcon:alias`
	This icon represents the Help option on the component menu. In a WebCenter application, the Help icon is displayed to the left of the Help menu item on the component's Actions menu.

C Resource Catalog Properties and Files

This appendix provides reference information about the configuration, location, and attributes of the default catalog definition file. This information is useful while performing the tasks described in Chapter 4, "Enabling Runtime Editing of Pages Using Oracle Composer".

This appendix includes the following sections:

- Section C.1, "Configuration and Location of Catalog Definitions"
- Section C.2, "Default Catalog Definition"
- Section C.3, "XML Schema"
- Section C.4, "Catalog Definition Attributes"

C.1 Configuration and Location of Catalog Definitions

The `rcv-config` section of your `adf-config.xml` defines the name of the default catalog. The `rcv-config` element is not available by default in your application's `adf-config.xml` file. You must add it if you want to change the name of the default catalog or specify a catalog selector class if you have chosen to implement multiple catalogs.

```
<rcv-config xmlns="http://xmlns.oracle.com/adf/rcs/viewer/adf-config">
  <catalog-selector class-name=""/>
  <default-catalog catalog-name="default-catalog"/>
</rcv-config>
```

The Resource Catalog service stores catalog definition files in MDS at the *`resource-catalog-root`* directory. The *`resource-catalog-root`* is defined as an MDS name space, hence the exact location depends on your MDS configuration. The default *`resource-catalog-root`* is defined by the `mds-package-prefix` attribute in the `adf-rcs-config` element in your `adf-config.xml`:

```
<adf-rcs-config xmlns="http://xmlns.oracle.com/adf/rcs/adf-config>
 <rcs-config mds-package-prefix="/oracle/adf/rc/metadata"/>
</adf-rcs-config>
```

The default Resource Catalog root is defined as:

```
<APPLICATION_ROOT>/mds/oracle/adf/rc/metadata
```

When you create or access catalogs using the Resource Catalog service APIs, the fully qualified MDS reference for the catalog (its storage location in the MDS repository) is dynamically constructed by appending the catalog identifier to the *`resource-catalog-root`* as follows:

```
<resource-catalog-root>/<catalog_id>.xml
```

If the value of *`resource-catalog-root`* is `/oracle/adf/rc/metadata` then all your catalog definitions must be stored within the `/oracle/adf/rc/metadata` MDS package.

If you create a catalog definition file and save it in MDS at `/oracle/adf/rc/metadata/myFirstCatalog.xml`, then that catalog's identifier would be `myFirstCatalog`. If this is to be the default catalog for your application, then `catalog-name` should be set to `myFirstCatalog` in `adf-config.xml`.

If your application requires multiple catalogs, you can organize them into sub packages. In this case, you might create a catalog definition file in MDS at `/oracle/adf/rc/metadata/hr/personalization.xml`. That catalog's identifier would be `hr/personalization`. If this is to be the default catalog for your application, then `catalog-name` should be set to `hr/personalization`.

C.2 Default Catalog Definition

Example C–1 shows the default catalog definition file to illustrate the most common elements.

Example C–1 Example of a Catalog Definition

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<catalogDefinition xmlns="http://xmlns.oracle.com/adf/rcs/catalog"
                   id="catalogDefinition"
                   name="Default Resource Catalog"
                   description="Default resource catalog definition containing
sample entries">
  <contents>
    <!-- *********************************************************************
    * Custom folder exposing ADF Faces components                           *
    * Comment out this element if you want to remove it from the catalog.   *
    ********************************************************************* -->
    <customFolder id="facesComponents" name="ADF Faces Components"
                  description="ADF Faces components you can add to application
pages"

factoryClass="oracle.adfinternal.view.page.editor.componentcatalog.adapter.Compone
ntObjectFactory"/>
    <!-- *********************************************************************
    * Dynamically include the portlets custom folder into the catalog at      *
    * runtime. The custom folder will only be included if the portlet runtime *
    * jar files are included in the application classpath.                    *
    ********************************************************************* -->
    <customContent id="portletContent"
 contentProviderClass="oracle.adf.rc.webcenter.WebCenterContentProvider"/>
    <!-- *********************************************************************
    * Uncomment the following <customFolder> element if you have configured   *
    * the Document Library service in your application and want to include    *
    * documents in the catalog.                                               *
    ********************************************************************* -->
<!--
    <customFolder id="doclibDocuments"
                  name="Documents"
                  description="Documents from the Document Library Service"

factoryClass="oracle.webcenter.content.model.rc.CustomFolderContextFactory"/>
```

```
-->
    <!-- *********************************************************************
    * Uncomment the following <resource> element if you have configured the   *
    * Document Library service in your application and want to include the    *
    * Document Library main view in the catalog.                              *
    *                                                                         *
    * NOTE: application.classpath is an implicitly defined "repository" for   *
    *       accessing ADF Libraries that are included in your classpath       *
    ********************************************************************* -->
<!--
    <resource id="doclibMainView"
              name="Document Library Task Flow"
              description="Main view of the Document Library Service"
              repository="application.classpath"
              path="doclib-service-view.jar/ADF_
TaskFlow/oracle+webcenter+doclib+view+jsf+taskflows+mainView.xml#doclib-document-l
ibrary"/>
-->
    <!-- *********************************************************************
    * To create a link to a task flow in an ADF Library:                      *
    * 1) add a <resource> tag to your catalog definition & set the            *
    *    "repository" attribute to "application.classpath"                    *
    *    (see doclibMainView above)                                           *
    * 3) set the "repository" attribute to the name of your file system       *
    *    connection.                                                          *
    * 4) set the "path" attribute for the task flow you are interested in.    *
    *    The path is of the following form:                                   *
    *                                                                         *
    *    path_to_jar/ADF_TaskFlow/task_flow_path                              *
    *                                                                         *
    * To obtain the task_flow_path, use the file system connection from (1)   *
    * and navigate to the task flow in your ADF Library. Mouse-over your task *
    * flow so its tool-tip is displayed. The tool-tip shows he fully qualified*
    * ID of the task flow. Take this value and replace all occurrances of "/" *
    * with "+". For example:                                                  *
    *                                                                         *
    * Tool-tip:

oracle/webcenter/collab/announcement/view/taskflows/main-view-definition.xml#annou
ncement-main-view
    * Task_flow_path:

oracle+webcenter+collab+announcement+view+taskflows+main-view-definition.xml#annou
ncement-main-view
    ********************************************************************* -->
   </contents>
</catalogDefinition>
```

C.3 XML Schema

Example C–2 provides the catalog definition XML schema for your reference.

Example C–2 XML Schema for Catalog Definition

```
<xsd:schema elementFormDefault="qualified"
            targetNamespace="http://xmlns.oracle.com/adf/rcs/catalog"
            xmlns:rcs="http://xmlns.oracle.com/adf/rcs/catalog"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:jaxb="http://java.sun.com/xml/ns/jaxb" jaxb:version="1.0">
  <xsd:annotation>
    <xsd:appinfo>
      <jaxb:schemaBindings>
        <jaxb:package name="oracle.adfinternal.rc.catalog.xml"/>
      </jaxb:schemaBindings>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexType name="descriptorType">
    <xsd:attribute name="attributeId" type="xsd:string" use="required"/>
    <xsd:attribute name="labelKey" type="xsd:string" use="required"/>
    <xsd:attribute name="shortLabelKey" type="xsd:string"/>
    <xsd:attribute name="searchable" type="xsd:boolean" default="true"/>
    <xsd:attribute name="multivalue" type="xsd:boolean" default="false"/>
    <xsd:attribute name="endUserVisible" type="xsd:boolean" default="true"/>
    <xsd:attribute name="resourceBundle" type="xsd:string"/>
  </xsd:complexType>
  <xsd:complexType name="schemaType">
    <xsd:sequence>
      <xsd:element name="descriptor" type="rcs:descriptorType"
       minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="resourceBundle" type="xsd:string"/>
  </xsd:complexType>
  <xsd:complexType name="attributeType">
    <xsd:attribute name="attributeId" type="xsd:string" use="required"/>
    <xsd:attribute name="value" type="xsd:string" use="required"/>
    <xsd:attribute name="isKey" type="xsd:boolean" use="required"/>
    <xsd:attribute name="resourceBundle" type="xsd:string"/>
  </xsd:complexType>
  <xsd:complexType name="attributesType">
    <xsd:sequence>
      <xsd:element name="attribute" type="rcs:attributeType"
       minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="resourceBundle" type="xsd:string"/>
  </xsd:complexType>
  <xsd:complexType name="itemType">
    <xsd:sequence>
      <xsd:element name="attributes" type="rcs:attributesType"
       minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string"/>
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
  <xsd:complexType name="resourceType">
    <xsd:complexContent>
      <xsd:extension base="rcs:itemType">
        <xsd:attribute name="repository" type="xsd:string"/>
        <xsd:attribute name="path" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="repositoryType">
    <xsd:complexContent>
      <xsd:extension base="rcs:itemType">
        <xsd:attribute name="repository" type="xsd:string"/>
        <xsd:attribute name="path" type="xsd:string"/>
        <xsd:attribute name="includeSubfolders" type="xsd:boolean"
         default="true"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="dynamicFolderSourceType">
    <xsd:attribute name="repository" type="xsd:string"/>
    <xsd:attribute name="path" type="xsd:string"/>
    <xsd:attribute name="includeSubfolders" type="xsd:boolean" default="true"/>
  </xsd:complexType>
  <xsd:complexType name="dynamicFolderType">
    <xsd:complexContent>
      <xsd:extension base="rcs:itemType">
        <xsd:sequence>
          <xsd:element name="source" type="rcs:dynamicFolderSourceType"
                       minOccurs="1" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="expression" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:group name="folderContentGroup">
    <xsd:choice>
      <xsd:element name="folder" type="rcs:folderType"/>
      <xsd:element name="repository" type="rcs:repositoryType"/>
      <xsd:element name="resource" type="rcs:resourceType" />
      <xsd:element name="dynamicFolder" type="rcs:dynamicFolderType" />
    </xsd:choice>
  </xsd:group>
   <xsd:complexType name="contentsType">
       maxOccurs="unbounded" />
      <xsd:group ref = "rcs:folderContentGroup" minOccurs="0"
   </xsd:complexType>
  <xsd:complexType name="folderType">
    <xsd:complexContent>
      <xsd:extension base="rcs:itemType">
        <xsd:sequence>
          <xsd:element name="contents" type="rcs:contentsType"
                       minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="catalogType">
    <xsd:complexContent>
      <xsd:extension base="rcs:folderType">
        <xsd:sequence>
          <xsd:element name="schema" type="rcs:schemaType" minOccurs="0"
           maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute name="contact" type="xsd:string"/>
        <xsd:attribute name="definitionFilter" type="xsd:string"/>
        <xsd:attribute name="resourceBundle" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="catalogDefinition" type="rcs:catalogType"/>
</xsd:schema>
```

C.4 Catalog Definition Attributes

The catalog definition schema provides a rich set of elements through which you can specify the structure and content of your resource catalogs.

- Section C.4.1, "catalogDefinition"
- Section C.4.2, "dynamicFolder"
- Section C.4.3, "folder"
- Section C.4.4, "attributes"
- Section C.4.5, "attribute"
- Section C.4.6, "resource"
- Section C.4.7, "schema"

For an example of creating a custom catalog definition file, see Section 6.3, "Creating a Custom Resource Catalog."

C.4.1 catalogDefinition

`catalogDefinition` element defines the overall characteristics of the catalog, such as the location of translated strings.

Example C–3 illustrates the use of `catalogDefinition`.

Example C–3 Example of catalogDefinition

```
<catalogDefinition xmlns="http://xmlns.oracle.com/adf/rcs/catalog"
                 id="PersonalSpaceCatalog"
                 name="Personal WebCenter Catalog"
                 description="Resource Catalog definiton for Personal WebCenter"
                 resourceBundle="oracle.webcenter.webcenterapp.internal.model.
                                rc.resource.PersonalSpaceCatalogBundle"
                 definitionFilter="oracle.webcenter.webcenterapp.internal.model.
                                  rc.PersonalSpaceCatalogFilter">
```

Table C–1 describes the keywords of the `catalogDefinition` element. For information about including `attributes` for elements, see Section C.4.4, "attributes."

Table C–1 catalogDefinition Keywords

Keyword	Description
`xmlns`	`http://xmlns.oracle.com/adf/rcs/catalog`
`id`	Is the identifier used when referring to this catalog definition.
`name`	Is the displayed name of the catalog definition.
`description`	Is text that describes the catalog definition.
`resourceBundle`	Is the name of the resource bundle to be used for obtaining translated attribute values, where `<attribute.... isKey="true"/>`. The `resourceBundle` attribute can also be used at lower levels in the document, such as on individual `attributes` elements. Refer to the XSD for further details.

Table C–1 (Cont.) catalogDefinition Keywords

Keyword	Description
`definitionFilter`	Is the fully qualified name of a Java class that implements the `oracle.adf.rc.spi.plugin.catalog.CatalogDefinitionFilter`. You can use a definition filter to dynamically exclude entries from a catalog at runtime. One common use of a definition filter is to provide different views of the same catalog to users with different roles. For example, Oracle WebCenter Spaces uses a definition filter to hide entries for services that the administrator has disabled.

C.4.2 dynamicFolder

`dynamicFolder` defines a folder that is dynamically populated based upon the results of a query.

Dynamic folders have `attributes` just like any other entry. For information about including `attributes` for elements, see Section C.4.4, "attributes."

Example C–4 Example of dynamicFolder

```
<dynamicFolder id="dynamicFolderId"
               name="dynamicFolderName"
               description="folderDescription"
               expression="jndiSearchExpression">
  <source repository="repositoryId" path="repositoryPath"
          includeSubfolders="true"/>
  ...
</dynamicFolder>
```

Table C–2 describes the keywords of the `dynamicFolder` element.

Table C–2 dynamicFolder Keywords

Keyword	Description
`id`	Is the identifier used when referring to this dynamic folder.
`name`	Is the displayed name of the dynamic folder.
`description`	Is text that describes the dynamic folder.
`expression`	Is a JNDI query expression. This query is executed against each of the source repositories and the results merged to form the dynamic folder contents.

source

`source` defines a repository or repository folder that should be included in a search as part of a dynamic folder. Dynamic folders can have one or more `source` repositories.

Example C–5 Example of source element

```
<source repository="repositoryId" path="repositoryPath" includeSubfolders="true"/>
```

Table C–3 source Keywords

Keyword	Description
`repository`	Is the identifier of a repository or connection you want to include in the search that generates the contents of the dynamic folder.

Table C–3 (Cont.) source Keywords

Keyword	Description
`path`	Is the path to the folder that should be included in the search.
`includeSubfolders`	Is a flag that indicates whether the search should be limited to the specified folder or include its subfolders as well.

C.4.3 folder

`folder` defines a catalog folder, used to organize content in the catalog. Folders can contain any number of entries including other folders. You can mix different types of catalog elements in the same folder.

Folders can have `attributes` just like any other entry. For information about including `attributes` for elements, see Section C.4.4, "attributes."

Example C–6 illustrates the use of the `folder` element.

Example C–6 Example of folder

```
<folder id="groupSpaceFolder">
 <attributes>
  <attribute value="GROUP_SPACE_FOLDER.TITLE" attributeId="Title" isKey="true"/>
  <attribute value="GROUP_SPACE_FOLDER.DESCRIPTION" attributeId="Description"
   isKey="true"/>
  <attribute value="GROUP_SPACE_FOLDER.KEYWORDS" attributeId="Subject"
   isKey="true"/>
 </attributes>
 <contents>
   ...
 </contents>
</folder>
```

Note: In addition to `folder` and `contents` elements, `customFolder` and `customContent` elements are displayed in the default catalog definition file. These elements are used to define content generated using Resource Catalog adapters.

You can comment out existing `customFolder` and `customContent` elements in your catalog definition file if required, but not add new ones. This is because you must use Resource Catalog APIs to define content in these elements and the required APIs are not exposed in this release.

Table C–4 describes the keywords of the `folder` element.

Table C–4 folder Keywords

Keyword	Description
`id`	Is the identifier used when referring to this folder.
`name`	Is the displayed name of the folder.
`description`	Is text that describes the folder.

C.4.4 attributes

Entries in a resource catalog have associated attributes that describe the entry. Values for some attributes can be obtained directly from the resources or from the `name` and `description` attributes on each XML element in the catalog definition. However, these values may not always be appropriate for displaying to end users, and often they are not available in different languages. Attributes defined using the `attributes` element in the catalog definition let you override those available from other sources. In addition, these attributes can be defined in terms of a key to a resource bundle so the actual values displayed reflect the current user's locale.

Within the `attributes` element you can insert any number of individual `attribute` elements that define the attributes you want to apply to a resource catalog entry.

Example C–8 illustrates the use of the `attributes` element.

Example C–7   Example of attributes and attribute Elements

```
<attributes>
 <attribute value="ANNOUNCEMENTS_MAIN_VIEW.TITLE" attributeId="Title"
  isKey="true"/>
 <attribute value="ANNOUNCEMENTS_MAIN_VIEW.DESCRIPTION" attributeId="Description"
  isKey="true"/>
 <attribute value="ANNOUNCEMENTS_MAIN_VIEW.KEYWORDS" attributeId="Subject"
  isKey="true"/>
 <attribute value="oracle.webcenter.collab.announcement"
  attributeId="WEBCENTER_SERVICE_ID" isKey="false"/>
</attributes>
```

Table C–5 describes the keywords of the `attributes` element.

Table C–5   attributes Keywords

Keyword	Description
`resourceBundle`	Is the name of the resource bundle to be used for obtaining translated attribute values.

C.4.5 attribute

`attribute` defines an attribute for a resource catalog entry that corresponds to a descriptor in the `schema` element. For information about attributes, see Section C.4.4, "attributes." Oracle Composer uses the following attributes:

- Title
- Description
- Subject
- IconURI
- ToolTip

You should include at least a `Title` and `Description` attribute for each catalog entry. A `Subject` attribute is also highly recommended to facilitate keyword searching of the resource catalog. The `Title` attribute overrides the `name` attribute on the parent element and is used to declare a display name that is translated through a resource bundle. The `Description` attribute overrides the `description` attribute on the parent element and is used to declare a description that is translated through a resource bundle.

You can declare your own attributes by including them in the `schema`. For more information, see Section C.4.7, "schema."

Example C–8 illustrates the use of the `attribute` element.

Example C–8 Example of attributes and attribute Elements

```
<attributes>
 <attribute value="ANNOUNCEMENTS_MAIN_VIEW.TITLE" attributeId="Title"
  isKey="true"/>
 <attribute value="ANNOUNCEMENTS_MAIN_VIEW.DESCRIPTION" attributeId="Description"
  isKey="true"/>
 <attribute value="ANNOUNCEMENTS_MAIN_VIEW.KEYWORDS" attributeId="Subject"
  isKey="true"/>
 <attribute value="oracle.webcenter.collab.announcement"
  attributeId="WEBCENTER_SERVICE_ID" isKey="false"/>
</attributes>
```

Table C–6 describes the keywords of the `attribute` element.

Table C–6 attribute Keywords

Keyword	Description
`attributeId`	Is the identifier of the attribute as declared in the `schema` element.
`value`	Is the value of the attribute.
`isKey`	Is a flag indicating whether the value is a literal value (`false`) or a key to a resource bundle (`true`). In most cases, attributes should be resource bundle keys and the values translated through a resource bundle. The resource bundle can be declared at the catalog level or on various lower level elements such as `attributes`.
`resourceBundle`	Is the name of the resource bundle to be used for obtaining translated attribute values.

For task flows included in your Resource Catalog, you can use the `attribute` element to pass task flow parameters and attributes of the enclosing `Show Detail Frame` component. You can do this by prefixing the `attributeId` values for the task flow parameters and `Show Detail Frame` attributes with **`parameter.`** and **`attr.`** respectively as shown in the following example:

```
<attributes>
  <attribute value="dark" attributeId="attr.background"
   isKey="false"/>
  <attribute value="#{myBean.myParam1}" attributeId="parameter.myParam1"
   isKey="false"/>
</attributes>
```

For more information, see Section 6.2.4, "How to Define Task Flow Parameters and Attributes of the Enclosing Show Detail Frames for Task Flows."

C.4.6 resource

`resource` is used to declare an individual resource, such as a task flow, in the catalog. Example C–9 illustrates the use of the `resource` element.

Example C–9 Example of resource

```
<resource id="announcements-main"  repository="application.classpath"
path="announcement-view.jar/ADF_
TaskFlow/oracle+webcenter+collab+announcement+view+taskflows+main-view-definition.
xml#announcement-main-view"/>
<attributes>
 <attribute value="ANNOUNCEMENTS_MAIN_VIEW.TITLE" attributeId="Title"
  isKey="true"/>
 <attribute value="ANNOUNCEMENTS_MAIN_VIEW.DESCRIPTION" attributeId="Description"
  isKey="true"/>
 <attribute value="ANNOUNCEMENTS_MAIN_VIEW.KEYWORDS" attributeId="Subject"
  isKey="true"/>
 <attribute value="oracle.webcenter.collab.announcement"
  attributeId="WEBCENTER_SERVICE_ID" isKey="false"/>
 </attributes>
</resource>
```

Table C–7 describes the keywords of the `resource` element. For information about including `attributes` for elements, see Section C.4.4, "attributes."

Table C–7 resource Keywords

Keyword	Description
`id`	Is the identifier for the element in the catalog. Identifiers are required and must be unique at the folder level.
`name`	Is the name of the element.
`description`	Is the text describing the element.
`repository`	Is name of the connection used to lookup the resource.

`path`	Is the path to the resource. For a task flow in an ADF Library, the path is of the following form:
	`path_to_jar``/ADF_TaskFlow/`*`task_flow_path`*
	To obtain the *`task_flow_path`*, create a file system connection in Application Connections and navigate to the task flow in your ADF Library. The tool-tip for the task flow shows the fully qualified ID of the task flow. Take this value and replace all occurrences of "/" with "+".
	For example:
	Tool-tip:
	`oracle/webcenter/collab/announcement/view/taskflows/mai n-view-definition.xml#announcement-main-view`
	Task_flow_path:
	`oracle+webcenter+collab+announcement+view+taskflows+mai n-view-definition.xml#announcement-main-view`

C.4.7 schema

`schema` contains one or more `descriptor` elements. The `descriptor` elements provide the resource catalog service with information about the attributes used within the catalog, such as whether they are searchable.

Example C–10 illustrates the use of the `schema` element.

Example C–10 Example of schema

```
<schema>
<descriptor searchable="true" labelKey="TITLE.PROMPT_KEY" endUserVisible="true"
```

```
 shortLabelKey="TITLE.SHORT_PROMPT_KEY" attributeId="Title" multivalue="false"/>
<descriptor searchable="true" labelKey="DESCRIPTION.PROMPT_KEY"
 endUserVisible="true" shortLabelKey="DESCRIPTION.SHORT_PROMPT_KEY"
 attributeId="Description" multivalue="false"/>
<descriptor searchable="true" labelKey="SUBJECT.PROMPT_KEY" endUserVisible="true"
 shortLabelKey="SUBJECT.SHORT_PROMPT_KEY" attributeId="Subject"
 multivalue="true"/>
<descriptor searchable="true" labelKey="WEBCENTER_SERVICE_ID.LABEL"
 endUserVisible="false" shortLabelKey="WEBCENTER_SERVICE_ID.SHORT_LABEL"
 attributeId="WEBCENTER_SERVICE_ID" multivalue="false"/>
<descriptor searchable="false" labelKey="TOOL_TIP.LABEL" endUserVisible="false"
 shortLabelKey="TOOL_TIP.SHORT_LABEL" attributeId="ToolTip" multivalue="false"/>
<descriptor searchable="false" labelKey="ICON_URI.LABEL" endUserVisible="false"
 shortLabelKey="ICON_URI.SHORT_LABEL" attributeId="IconURI" multivalue="false"/>
</schema>
```

Table C–8 describes the keywords of the `descriptor` element.

Table C–8 descriptor Keywords

Keyword	Description
`attributeId`	Is the identifier of the attribute being described.
`searchable`	(True or False) Indicates whether the attribute is included in resource catalog searches.
`multivalue`	(True or False) Indicates whether the attribute can have multiple values.
`endUserVisible`	(True or False) Indicates whether the attribute should be exposed to end users.
`resourceBundle`	Is the name of the resource bundle used to translate the label keys.
`labelKey`	Is the resource bundle key for the attribute's label.
`shortLabelKey`	Is the resource bundle key for the attribute's short label.

D Calling Oracle SES to Search Data

In custom WebCenter applications, it may be useful to provide users with the ability to search the data, or a portion of the data, presented to them on a page. To achieve this type of functionality, you can use a web service data control to call the searching function of Oracle Secure Enterprise Search (SES).

For more information on Oracle SES, see the Oracle Secure Enterprise Search documentation that comes with the product.

D.1 How to Call Oracle SES to Search Data

This appendix demonstrates how to create a web service data control to search SES data. This involves the following steps:

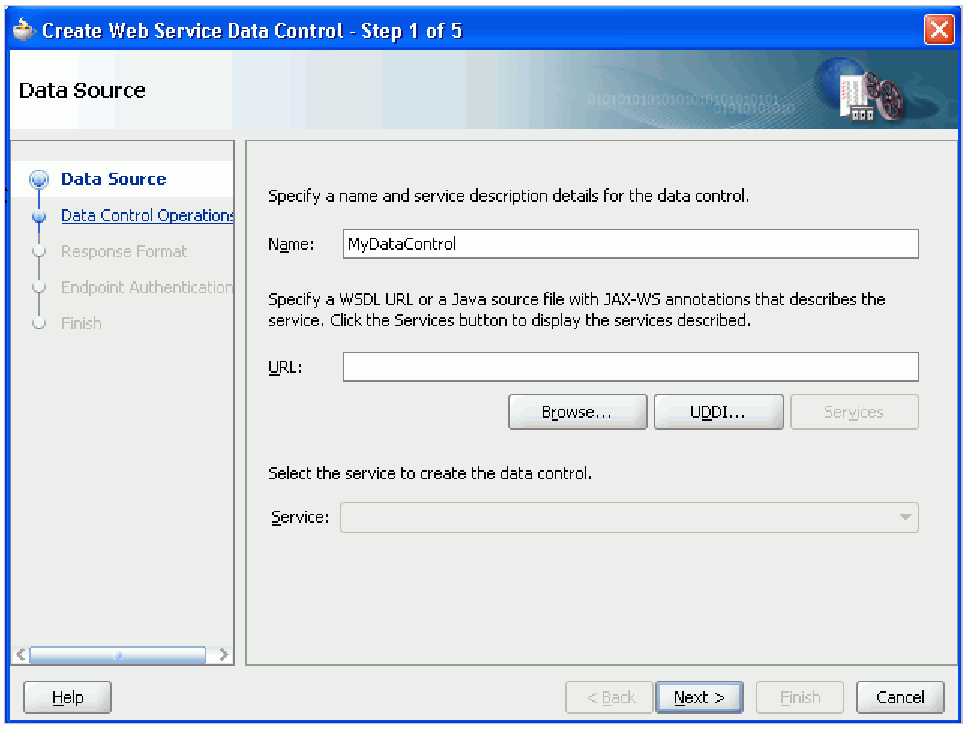
- Section D.1.1, "Create a Data Control"
- Section D.1.2, "Use the Data Control on a Page"
- Section D.1.3, "Format the Output"

D.1.1 Create a Data Control

To build a data control that uses Oracle Secure Enterprise Search:

1. If you have not already done so, install Oracle Secure Enterprise Search by referring to the installation guide for your platform.
2. If you have not already done so, in Oracle JDeveloper, create a custom WebCenter application.
3. Right-click the project where you want to incorporate searching and select **New** from the context menu.
4. In the New Gallery, make sure to filter by **All Technologies**.
5. Expand **Business Tier** and select **Web Services**.
6. In the **Items** list, select **Web Service Data Control** and click **OK**.
7. Enter a name for the data control. The wizard page should look similar to Figure D–1.

Figure D–1 Data Source Page of Web Service Data Control Wizard



Specify a name and service description details for the data control.

Name: MyDataControl

Specify a WSDL URL or a Java source file with JAX-WS annotations that describes the service. Click the Services button to display the services described.

URL:

Select the service to create the data control.

Service:

8. Enter a Web Services Description Language (WSDL) URL. The WSDL URL for Oracle Secure Enterprise Search has the following form:

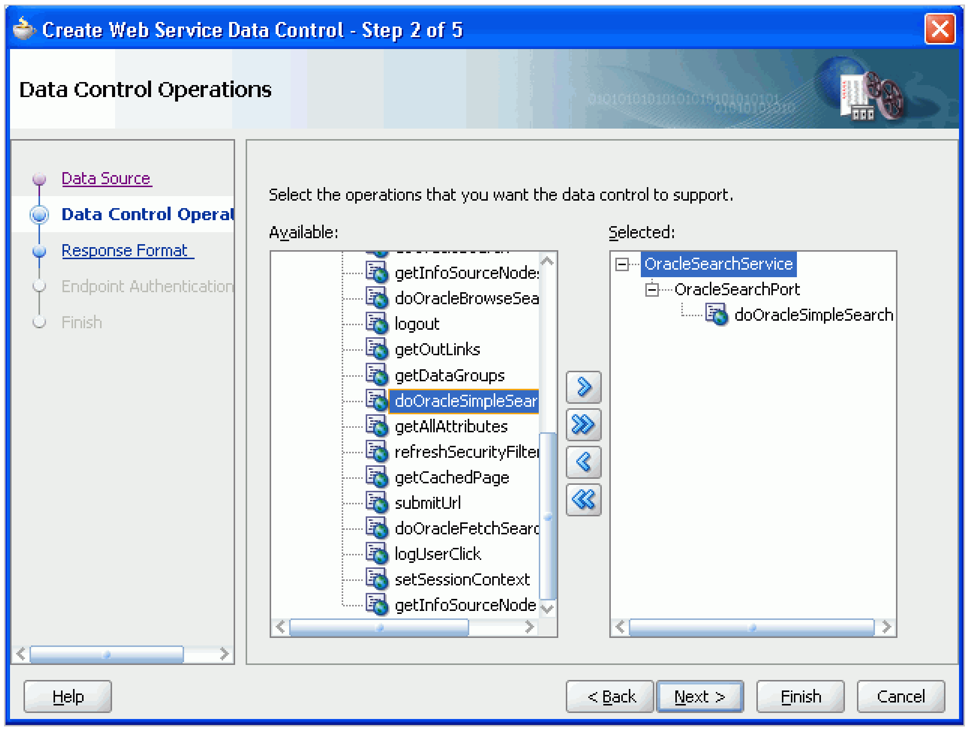
   ```
   http://host:port/search/query/OracleSearch?WSDL
   ```

   Note: When Oracle Secure Enterprise Search is installed, the endpoint listed in the WSDL points to a placeholder location. After the data control is created, you must correct the endpoint URL using the Structure panel. This step is described later in this procedure.

   ---

9. If the service is not automatically provided, then click **Services**.
10. Click **Next**.
11. From the Data Control Operations page, select the methods you want to expose by moving them from the **Available** list to the **Selected** list. For the purposes of this example, move **doOracleSimpleSearch** to the **Selected** list. When you are done, the page should look similar to Figure D–2.

Figure D–2 Data Control Operations Page of Web Service Data Control Wizard



Create Web Service Data Control - Step 2 of 5

Data Control Operations

Select the operations that you want the data control to support.

12. Click **Finish**. To review the other pages of the wizard first, click **Next** until you reach the end of the wizard, and then click **Finish**.

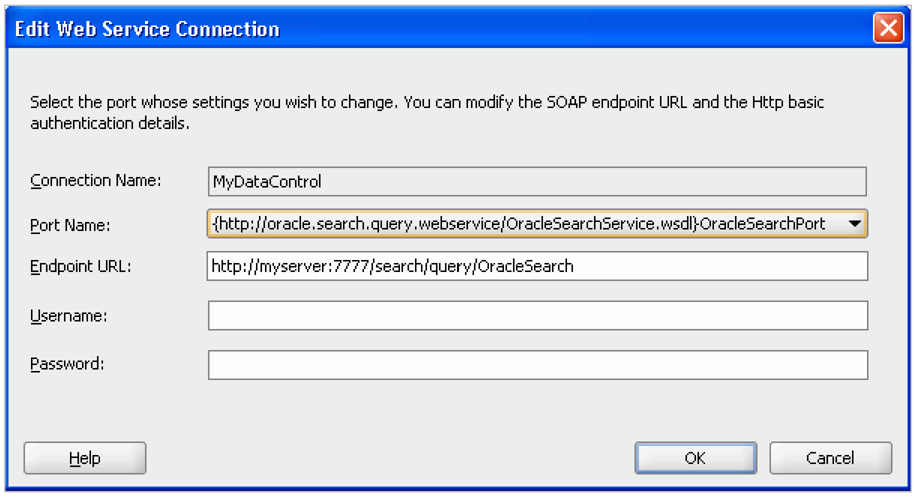
13. In the Application Navigator, expand **Application Sources** and then **view**.

14. Click **`DataControls.dcx`** and its structure will appear in the Structure panel below the Application Navigator.

15. Right-click your data control in the Structure panel and select **Edit Web Service Connection** from the context menu.

16. The Edit Web Service Connection dialog appears as shown in Figure D–3. As noted earlier, the endpoint listed in the WSDL now points to a placeholder location. Enter again the same endpoint URL.

Figure D–3 Edit Web Service Connection Dialog



Select the port whose settings you wish to change. You can modify the SOAP endpoint URL and the Http basic authentication details.

Connection Name:	MyDataControl
Port Name:	{http://oracle.search.query.webservice/OracleSearchService.wsdl}OracleSearchPort
Endpoint URL:	http://myserver:7777/search/query/OracleSearch
Username:	
Password:	

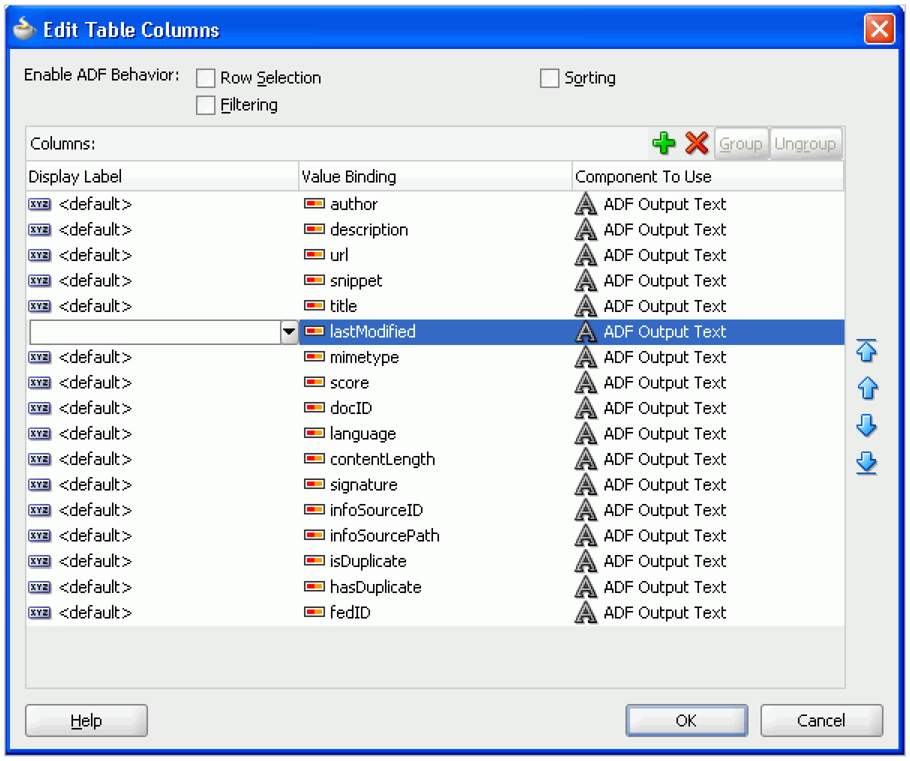
17. Click **OK**.

D.1.2 Use the Data Control on a Page

Now you can create a page on which to drop your new data control.

1. Right-click your project and select **New** from the context menu.
2. Under **Web Tier**, select **JSF**. Under **Items**, select **JSF Page**.
3. Click **OK**.
4. On the Create JSP Page, accept the default settings for everything except **File Name**. Select a meaningful file name.
5. Click **OK**.
6. You should see your data control listed under the Data Controls section of the Application Navigator. Expand the top level node of your data control and then expand **Return** and **Return**. Now you can see all the data that is returned by the data control.
7. For the purposes of this example, drag and drop **resultElements** onto your page.
8. Select **Tables**, **ADF Read-only Table** from the context menu that appears.
9. The Edit Table Columns dialog appears. (Figure D–4) Select **lastModified** and click the **Delete** icon. Click **OK**.

Figure D–4 Edit Table Columns Dialog



Display Label	Value Binding	Component To Use
<default>	author	ADF Output Text
<default>	description	ADF Output Text
<default>	url	ADF Output Text
<default>	snippet	ADF Output Text
<default>	title	ADF Output Text
	lastModified	ADF Output Text
<default>	mimetype	ADF Output Text
<default>	score	ADF Output Text
<default>	docID	ADF Output Text
<default>	language	ADF Output Text
<default>	contentLength	ADF Output Text
<default>	signature	ADF Output Text
<default>	infoSourceID	ADF Output Text
<default>	infoSourcePath	ADF Output Text
<default>	isDuplicate	ADF Output Text
<default>	hasDuplicate	ADF Output Text
<default>	fedID	ADF Output Text

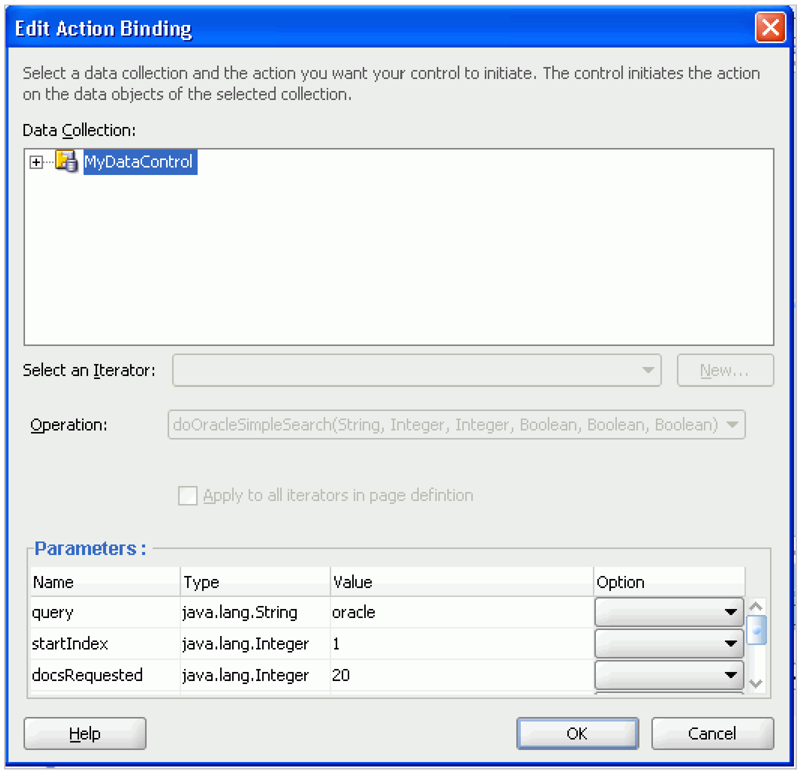
10. For the sake of simplicity, you can hard code the values for each parameter as shown in Table D–1.

Table D–1 Parameter Values for Action Binding Editor

Parameter	**Value**
`query`	The string you want to search for, for example, oracle
`startIndex`	1
`docsRequested`	20
`dupRemoved`	false
`dupMarked`	false
`returnCount`	false

11. The dialog should look like Figure D–5. Click **OK**.

Figure D–5 Action Binding Editor



Edit Action Binding

Select a data collection and the action you want your control to initiate. The control initiates the action on the data objects of the selected collection.

Data Collection:

MyDataControl

Select an Iterator:

Operation: doOracleSimpleSearch(String, Integer, Integer, Boolean, Boolean, Boolean)

Apply to all iterators in page defintion

Parameters :

Name	Type	Value	Option
query	java.lang.String	oracle	
startIndex	java.lang.Integer	1	
docsRequested	java.lang.Integer	20	

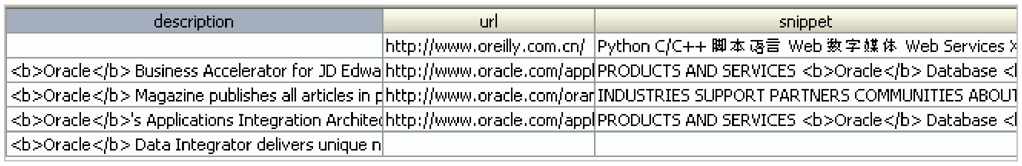
12. Click **OK**.

13. Now you can run your page. In the Application Navigator, right-click the page and select **Run** from the context menu.

D.1.3 Format the Output

At runtime, you can browse through your output and notice how occurrences of the word oracle are surrounded by `<b></b>` tags . Figure D–6 shows a small sample fragment of this output. Currently, the search hits are not formatted because you have not yet converted the view component that renders this column to `OutputFormatted`.

Figure D–6 Sample of Searched Output, Hits Unformatted



description	url	snippet
	http://www.oreilly.com.cn/	Python C/C++ 脚本语言 Web 数字媒体 Web Services X
<b>Oracle</b> Business Accelerator for JD Edwa	http://www.oracle.com/appl	PRODUCTS AND SERVICES <b>Oracle</b> Database <l
<b>Oracle</b> Magazine publishes all articles in p	http://www.oracle.com/orar	INDUSTRIES SUPPORT PARTNERS COMMUNITIES ABOUT
<b>Oracle</b>'s Applications Integration Architec	http://www.oracle.com/appl	PRODUCTS AND SERVICES <b>Oracle</b> Database <l
<b>Oracle</b> Data Integrator delivers unique n		

To format the search output:

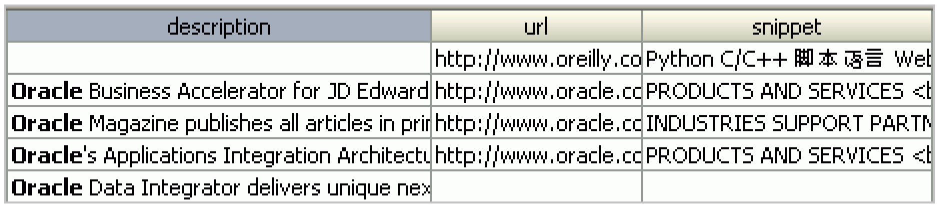
1. To format the output, return to Oracle JDeveloper and display the page containing the data control by clicking its tab in the Design view..

2. In the Structure panel, find and expand one of the columns that contains `oracle`; for example, the description column.

3. Right-click **`af:outputText - #{row.snippet}`** for that column.

4. In the Source view, change **`af:outputText`** to **`af:outputFormatted`**.
5. Run the page again to see the changes. You should see output similar to that in Figure D–7. Notice how the `<b></b>` around the word `oracle` is now interpreted so that search hits appear in bold.

Figure D–7 Sample of Searched Output, Hits Formatted



description	url	snippet
	http://www.oreilly.co	Python C/C++ 脚本语言 Wet
Oracle Business Accelerator for JD Edward	http://www.oracle.cc	PRODUCTS AND SERVICES <t
Oracle Magazine publishes all articles in prir	http://www.oracle.cc	INDUSTRIES SUPPORT PARTN
Oracle's Applications Integration Architectu	http://www.oracle.cc	PRODUCTS AND SERVICES <t
Oracle Data Integrator delivers unique nex		

6. Using the Structure panel, right-click and delete any extraneous columns of your table.
7. Repeat steps 1-5 for each of the remaining columns on your page.
8. Save and run the page again.

The rest of this section describes some other formatting features you can include.

1. You might see automatic pagination in the upper right corner of the table. The default number of rows for an Oracle ADF table is 10, but you chose to return 20 results.

   To enhance the page, you can provide an estimated count. From the Application Navigator , under Data Controls, drag and drop **estimatedHitCount** onto the page, before the table. Select **Texts**, **ADF Output Text w/ Label** from the context menu.
2. Another useful feature for this table would be a link to open the document. Drag a **Go Link** from under **ADF Faces** in the **Component Palette** and drop it right in front of **outputFormatted** of the **description** column. Note that you can perform this step a little more accurately in the Structure panel.
3. In the Property Inspector, change the **Text** attribute to **Open**. For the **Destination** attribute, enter the following: `#{row.url}`.
4. Click the **Source** tab. You should see something similar to the following in the source for your page:

```
<af:goLink text="Open" destination="#{row.url}"/>
```

5. You could also add an input box and a Submit button to enable users to perform their own custom searches. Click the **Design** tab.
6. From the Application Navigator , under Data Controls, expand the **Parameters** node under your data control.
7. Drag **query** and drop it just above the left corner of your table. Select **Texts**, **ADF Input Text w/ Label** from the context menu.
8. Drag **doOracleSimpleSearch** and drop it just below the parameter input box you just created. Select **Methods**, **ADF Button** from the context menu.
9. To avoid getting an error when someone first runs the page, you must set a default value for the query parameter. Go to the page definition, `PageDef.xml`.
10. In the Structure panel, expand **variables** under **executables** and select **doOracleSimpleSearch_query**.

11. In the Property Inspector, select the **DefaultValue** property and enter a default search; for example, **oracle**.

12. Run the page. Enter a different search term in the box and click the button you just created. The results should change.

Note: Because you previously limited the number of results returned, the hit count you added at the bottom of the page is now artificially limited. To get an accurate hit count, find the following line in your `PageDef.xml` file:

```
<NamedData NDName="returnCount" NDValue="false"
   NDType="java.lang.Boolean"/>
```

and change it to:

```
<NamedData NDName="returnCount" NDValue="true"
   NDType="java.lang.Boolean"/>
```

E Additional Portlet Configuration

This appendix discusses configuration information for some of the portlet technologies available with Oracle WebCenter.

This chapter includes the following sections:

- Section E.1, "Java Portlet Configuration Tips"
- Section E.2, "OmniPortlet Configuration Tips"
- Section E.3, "Web Clipping Portlet Configuration Tips"
- Section E.4, "Setting Up a Preference Store"
- Section E.5, "Portlet Preference Store Migration Utilities"

For more information on using the portlet technologies at design time, refer to their respective chapters throughout this guide:

- Chapter 29, "Creating Portlets with the Portlet Wizard"
- Chapter 30, "Coding Portlets"
- Chapter 31, "Creating Portlets with OmniPortlet"
- Chapter 32, "Creating Content-Based Portlets with Web Clipping"

For more information on using these technologies at run time, refer to their respective chapters in Oracle Fusion Middleware User's Guide for Oracle WebCenter.

E.1 Java Portlet Configuration Tips

This section contains configuration information for Java portlets.

Disabling a WSRP Test Page

To disable your WSRP test page, perform the following steps:

1. In Oracle JDeveloper, go to the Application Navigator and expand the **Web Content** and **WEB-INF** folders.
2. Double-click the `web.xml` file to open it.
3. In the Source view, look for the following element and comment it out:

```
<servlet-mapping>
      <servlet-name>WSRPTestPage</servlet-name>
      <url-pattern>/info</url-pattern>
   </servlet-mapping>
```

4. Save the `web.xml` file.

5. Deploy the portlet application.
6. Run your test page in a browser, an error occurs.

E.2 OmniPortlet Configuration Tips

This section contains configuration information for OmniPortlet. To learn more about the OmniPortlet wizard, see Chapter 31, "Creating Portlets with OmniPortlet." This section contains configuration information for the following areas:

- Section E.2.1, "Configuring the OmniPortlet Producer to Access Data Outside a Firewall"
- Section E.2.2, "Configuring the OmniPortlet Producer to Access Other Relational Databases"
- Section E.2.3, "Configure Portal Tools and Web Producers (Optional)"

Note: In this section, *OmniPortlet_WAR_DIR* indicates the directory where `omniPortlet.war` is unwrapped in the WebLogic Server. The exact path can vary depending on your installation. To determine this path, search for `omniPortlet/provider.xml` in the file system. For example, run the following command in UNIX:

```
> find <DOMAIN_DIR> -name "provider.xml" | grep -i omniportlet
```

In the Integrated WebLogic Server (Integrated WLS or Default Server), the path of OmniPortlet's `provider.xml` is located here:

```
JDEV_SYSTEM_DIRECTORY/DefaultDomain/servers/DefaultServer/tmp/_WL_
user/portalTools_11.1.1.1.0/RANDOMLY_GENERATED_
DIRECTORY/war/WEB-INF/providers/omniPortlet/provider.xml
```

In the Application Server 11*g* installation, the path of OmniPortlet's `provider.xml` is located here:

```
FMW_HOME/user_projects/domains/wc_domain/servers/WLS_Portlet/tmp/_
WL_user/portalTools_11.1.1.1.0/RANDOMLY_GENERATED_
DIRECTORY/war/WEB-INF/providers/omniPortlet/provider.xml
```

E.2.1 Configuring the OmniPortlet Producer to Access Data Outside a Firewall

If the OmniPortlet producer is inside your firewall, you must configure the proxy information so that OmniPortlet can access URLs of data (such as CSV, XML, or Web Services) located outside the firewall. To do so, you can either set the proxy information in the command line when you start your WebLogic server. Or, you can set up the proxy information in OmniPortlet's `provider.xml` file, located here: *`OmniPortlet_WAR_DIR`*`/WEB-INF/providers/omniPortlet/provider.xml`.

Note: For the Web Service data source, you must set the proxy information in both the `provider.xml` file and using the command line parameters.

- To set the proxy information in the command line when starting the WebLogic server, set the parameters as described in Table E–1, if you are using an HTTP Proxy Host, or Table E–2, if you are using an HTTPS Proxy Host.

Table E–1 HTTP Proxy Information Command Line Parameters

Parameter	Description
`http.proxyHost`	The host name of a proxy server if one is required to make a URL connection from the OmniPortlet producer to its data sources.
`http.proxyPort`	The port number for the HTTP Proxy Host.
`http.nonProxyHosts`	The name of any domain or hosts to which you can directly connect, bypassing a proxy server, such as your local machine: `localhost|localhost.localdomain` Hosts can be fully qualified host names or can be IP addresses.
`http.proxyUser`	The user to log into the proxy server if the proxy server requires authentication.
`http.proxyPassword`	The password to log into the proxy server if the proxy server requires authentication.
`http.proxyAuthType`	The authentication type of the proxy server. Acceptable values: `Basic | Digest`.
`http.proxyAuthRealm`	The name of the realm of the proxy server. If you do not know the name of the realm, contact the proxy server administrator.

Table E–2 HTTPS Proxy Information Command Line Parameters

Parameter	Description
`https.proxyHost`	The host name of a proxy server if one is required to make a URL connection from the OmniPortlet producer to its data sources.
`https.proxyPort`	The port number for the HTTPS Proxy Host.
`https.nonProxyHosts`	The name of any domain or hosts to which you can directly connect, bypassing a proxy server, such as your local machine: `localhost|localhost.localdomain` Hosts can be fully qualified host names or can be IP addresses.
`https.proxyUser`	The user to log into the proxy server if the proxy server requires authentication.
`https.proxyPassword`	The password to log into the proxy server if the proxy server requires authentication.
`https.proxyAuthType`	The authentication type of the proxy server. Acceptable values: `Basic | Digest`.
`https.proxyAuthRealm`	The name of the realm of the proxy server. If you do not know the name of the realm, contact the proxy server administrator.

The following are examples of three parameters and their values:

```
-Dhttps.proxyHost=myProxyServer.mycompany.com
-Dhttps.proxyPort=80
-Dhttps.nonProxyHosts=localhost|localhost.localdomain|127.0.0.1|
```

- To configure the proxy information in the `provider.xml` file, refer to Table E–3 for a list of parameters and their descriptions.

Table E–3 Provider.xml Tags

Parameter	Description
`httpProxyHost`	Enter the host name of a proxy server if one is required to make a URL connection from the OmniPortlet producer to its data sources.
`httpProxyPort`	Enter the port number for the HTTP Proxy Host.
`dontProxyFor`	Enter the name of any domain or hosts to which you can directly connect, bypassing a proxy server. Domain names are the part of a URL that contain the names of a business, or organization, or government agency, for example: `*.company.com, *.us.company.com` Hosts can be fully qualified host names or can be IP addresses.
`proxyUseAuth`	Acceptable values: `true | false` Enter true if your proxy server requires authentication. The authentication parameters will be specified by the following tags: `proxyType`, `proxyRealm`, `proxyUseGlobal`, `proxyUserName`, and `proxyPassword`.
proxyType	Acceptable values: `Basic | Digest` Choose the type of proxy server this provider. Note: For more information about basic or digest authentication, see `http://www.faqs.org/rfcs/rfc2617.html`.
`proxyRealm`	Enter the name of the realm of the proxy server that is accessed by the user according to the login information described later in the table. If you do not know the name of the realm, then contact the administrator of the proxy server.
`proxyUseGlobal`	Acceptable values: true | `false` If true, then the `<proxyUser>` and `<proxyPassword>` values are used for all users. Users do not see the Proxy Authentication section on the Source tab and Personalize screen. If false, then page designer must log in using the Proxy Authentication section on the Source tab when they define the portlet. The end user must log in using the Proxy Authentication section on the Personalize screen. If `<proxyUsername>` and `<proxyPassword>` are given, then they are only used for public users.
`proxyUserName`	Enter the username to log in to the proxy server.
`ProxyPassword`	Enter the password for the specified username. You must prefix ! before your plain password text. It will then be encrypted in the `provider.xml` file for protection once the producer starts.

The following is a basic example of using a proxy to access data outside a firewall:

```
<proxyInfo class="oracle.portal.provider.v2.ProxyInformation">
<httpProxyHost>www-proxy.mycompany.com</httpProxyHost>
<httpProxyPort>80</httpProxyPort>
<proxyUseAuth>false</proxyUseAuth>
</proxyInfo>
```

The following example requires a login and basic authentication for all users for the proxy server:

```
<proxyInfo class="mycompany.portal.provider.v2.ProxyInformation">
<httpProxyHost>stport823.mycompany.com</httpProxyHost>
<httpProxyPort>8080</httpProxyPort>
```

```
<proxyUseAuth>true</proxyUseAuth>
<proxyType>Basic</proxyType>
<proxyRealm>stport823</proxyRealm>
<proxyUseGlobal>false</proxyUseGlobal>
</proxyInfo>
```

E.2.2 Configuring the OmniPortlet Producer to Access Other Relational Databases

The OmniPortlet SQL data source is preconfigured to access Oracle databases using the Oracle JDBC drivers, and ODBC data sources using Sun Microsystem's JDBC-ODBC driver. Oracle allows developers to access other relational databases using DataDirect JDBC drivers.

See Also: For a list of supported databases, Certification Matrix for Oracle Application Server and DataDirect JDBC available on the Oracle Technology Network (`http://www.oracle.com/technology/index.html`).

This section contains the following steps:

- Installing DataDirect JDBC Drivers
- Registering DataDirect Drivers in OmniPortlet

E.2.2.1 Installing DataDirect JDBC Drivers

The following DataDirect JDBC drivers are included with the WebLogic Server installation:

- `wlinformix.jar`
- `wlsqlserver.jar`
- `wlutil.jar`
- `wldb2.jar`
- `wlresource.jar`
- `wlspy.jar`
- `wlbase.jar`

If you do not plan to use these DataDirect drivers, you can instead download DataDirect JDBC drivers to access the desired database. These drivers are packaged in a single ZIP, which you can download from the following location:

`http://www.oracle.com/technology/software/products/ias/htdocs/utilsoft.html`

To install DataDirect JDBC drivers, perform the following steps:

1. Unzip the contents of the ZIP file into a temporary directory, for example `/temp/datadirect`.
2. Copy the DataDirect JDBC drivers from the temporary directory to your WebLogic Server directory: `WLS_DOMAIN_DIRECTORY/lib`.

E.2.2.2 Registering DataDirect Drivers in OmniPortlet

OmniPortlet is implemented as a Web producer and all the configuration properties are stored in the `provider.xml` file. To use DataDirect JDBC drivers with OmniPortlet, you must register these drivers in the `provider.xml` file.

To register the new DataDirect JDBC drivers, perform the following steps:

1. Back up the file, *`OmniPortlet_WAR_DIRECTORY`*/WEB-INF/providers/omniPortlet/provider.xml, and then open the file.
2. Add the drivers that you want to use for the SQL data source configuration entry. To do this, perform the following:
   a. Search for the XML tag, `driverInfo`.
   b. Add a new entry after the last `driverInfo` tag.

   The following are examples of the driverInfo for WebLogic DataDirect drivers:

   - Microsoft SQL Server:

```
<driverInfo class="oracle.webdb.reformlet.data.jdbc.JDBCDriverInfo">
  <name>Microsoft SQL Server</name>
  <sourceDataBase>other</sourceDataBase>
  <subProtocol>weblogic:sqlserve</subProtocol>
  <connectString>mainProtocol:subProtocol://databaseName</connectString>
  <driverClassName>weblogic.jdbc.sqlserver.SQLServerDriver
  </driverClassName>
  <dataSourceClassName>com.oracle.ias.jdbcx.sqlserver.SQLServerDataSource
  </dataSourceClassName>
  <connHandlerClass>oracle.webdb.reformlet.data.jdbc.JDBCConnectionHandler
  </connHandlerClass>
  <connPoolSize>5</connPoolSize>
  <loginTimeOut>30</loginTimeOut>
</driverInfo>
```

   - Sybase:

```
<driverInfo class="oracle.webdb.reformlet.data.jdbc.JDBCDriverInfo">
  <name>Sybase</name>
  <sourceDataBase>other</sourceDataBase>
  <subProtocol>weblogic:sybase</subProtocol>
  <connectString>mainProtocol:subProtocol://databaseName</connectString>
  <driverClassName>weblogic.jdbc.sybase.SybaseDriver
  </driverClassName>
  <connHandlerClass>
  oracle.webdb.reformlet.data.jdbc.JDBCODBCConnectionHandler
  </connHandlerClass>
  <connPoolSize>5</connPoolSize>
  <loginTimeOut>30</loginTimeOut>
</driverInfo>
```

   - DB2:

```
<driverInfo class="oracle.webdb.reformlet.data.jdbc.JDBCDriverInfo">
  <name>DB2</name>
  <sourceDataBase>other</sourceDataBase>
  <subProtocol>weblogic:db2</subProtocol>
  <connectString>mainProtocol:subProtocol://databaseName</connectString>
  <driverClassName>weblogic.jdbc.db2.DB2Driver
  </driverClassName>
  <connHandlerClass>
  oracle.webdb.reformlet.data.jdbc.JDBCODBCConnectionHandler
  </connHandlerClass>
  <connPoolSize>5</connPoolSize>
  <loginTimeOut>30</loginTimeOut>
```

```
</driverInfo>
```

- Informix:

```
<driverInfo class="oracle.webdb.reformlet.data.jdbc.JDBCDriverInfo">
  <name>Informix</name>
  <sourceDataBase>other</sourceDataBase>
  <subProtocol>weblogic:informix</subProtocol>
  <connectString>mainProtocol:subProtocol://databaseName</connectString>
  <driverClassName>weblogic.jdbc.informix.InformixDriver
  </driverClassName>
  <connHandlerClass>
  oracle.webdb.reformlet.data.jdbc.JDBCODBCConnectionHandler
  </connHandlerClass>
  <connPoolSize>5</connPoolSize>
  <loginTimeOut>30</loginTimeOut>
</driverInfo>
```

Table E–4 describes the parameters in the `driverInfo` property.

Table E–4 Parameters in the driverInfo Property

Parameter	Description
`name`	Name of the database you want to use. This name will be used on the Source tab of the OmniPortlet wizard.
`sourceDataBase`	Internal value. Set the value to `other`.
`subProtocol`	JDBC subprotocol name used by OmniPortlet to create the connection string, for example `sqlserver`, `sybase`, or `db2`. To get the list of subprotocol names, see the DataDirect JDBC driver documentation using the links provided at the end of this section.
`connectString`	Description of the connect string format. For DataDirect drivers, the format is: `mainProtocol:subProtocol://databaseName`
`driverClassName`	Name of the driver class. To get the different values, see the DataDirect JDBC driver documentation using the links provided at the end of this section.
`dataSourceClassName`	Name of the data source class that implements connection pooling. This parameter is only available in OmniPortlet version 9.0.4.1 or later. See Table E–5 for the right data source class name for your driver.

`connHandlerClass`	Class used by OmniPortlet to manage the driver and connection pooling. The value is either of the following:
	▪ For OmniPortlet version 9.0.4.1 or later:
	`oracle.webdb.reformlet.data.jdbc.JDBCConnectionHandler`
	▪ For OmniPortlet versions before 9.0.4.1:
	`oracle.webdb.reformlet.data.jdbc.JDBCODBCConnectionHandler`

`connPoolSize`	Minimum number of connections that are opened by the connection pool.
`loginTimeOut`	Maximum time, in seconds, that this data source will wait while attempting to connect to a database.

Table E–5 lists the values for the `driverClassName` and `dataSourceClassName` properties for specific DataDirect JDBC drivers.

Table E–5 Parameters and Values for driverClassName and dataSourceClassName

DataDirect Drivers Supported	Properties
Microsoft SQL Server	Parameter: `driverClassName` Value: `weblogic.jdbc.sqlserver.SQLServerDriver`
Sybase	Parameter: `driverClassName` Value: `weblogic.jdbc.sybase.SybaseDriver`
DB2	Parameter: `driverClassName` Value: `weblogic.jdbc.db2.DB2Driver`
Informix	Parameter: `driverClassName` Value: `weblogic.jdbc.informix.InformixDriver`

3. Save the `provider.xml` file.
4. Stop and start the Oracle WebLogic managed server instance where your portlet producer was deployed. To do so, navigate to your WLS_HOME, then to the subdirectory `opmn/bin`.

Note: If you are using OmniPortlet in a multiple nodes configuration, for example, in a clustering or load-balancing environment, then you must manually copy the `provider.xml` file on each node.

See Also: For more information on how to use DataDirect JDBC drivers, refer to Chapter 31, "Creating Portlets with OmniPortlet."

E.2.3 Configure Portal Tools and Web Producers (Optional)

To ensure that the OmniPortlet and Web Clipping producers, locally built, and custom built Web producers work properly in the middle-tier environment, some additional configuration may be needed. If OmniPortlet or any other Web producers already have customization in the file system, then PDK-Java provides a Preference Store Migration/Upgrade Utility that can be used to migrate the existing customizations to the database and upgrade customizations from earlier releases. See Section E.5, "Portlet Preference Store Migration Utilities" for more information about the PDK Preference Store Migration Utility.

Configuring Portal Tools Producers in the Multiple Middle-Tier Environment

By default, the OmniPortlet producer uses the Database Preference Store. It can work in a multiple middle-tier environment without additional configuration.

You can find more information about configuring the database Preference Store in Section E.5, "Portlet Preference Store Migration Utilities"as well as the PDK article titled “Installing the DBPreferenceStore Sample (V2)”, (with the filename `installing.db.preference.store.v2.html`) located in the `pdksoftware10132.zip` file located on the Oracle Technology Network (`http://www.oracle.com/technology/products/webcenter/index.html`).

1. If you have already created an OmniPortlet instance with customizations in the file system, then you must migrate these customizations to the database using the Preference Store Migration Utility. To run the migration utility, perform the following steps:

   1. Navigate to the middle-tier Oracle home directory using the following command:

      ```
      cd ORACLE_HOME
      ```

   2. Run the following command to migrate OmniPortlet data from a file-based Preference Store (`FilePreferenceStore`) to the database Preference Store (`DBPreferenceStore`):

      ```
      java -classpath
      lib/dms.jar:jdbc/lib/ojdbc14dms.jar:portal/jlib/pdkjava.jar:portal/jlib/
      ptlshare.jar oracle.portal.provider.v2.preference.MigrationTool -mode
      filetodb -pref1UseHashing true -pref1RootDirectory
      portal/portletdata/tools/omniPortlet
      -pref2User <User_Name> -pref2Password <User_Password> -pref2URL
      jdbc:oracle:thin:@infra.host.com:1521:orcl
      ```

   See Section E.5, "Portlet Preference Store Migration Utilities" for more information about the PDK Preference Store Migration Utility.

2. Typically, you perform the HTTP Proxy configuration for OmniPortlet and Web Clipping before you configure the LBR. To do it after the LBR is configured, perform the following steps:

   a. The Portal Tools configuration information is stored in the `provider.xml` file on the middle-tier server. You must update the configuration directly on one middle tier (for example, **M1**) and then propagate it across all middle tiers front-ended by the LBR. Before you do this, you must shut down all middle tiers except **M1**.

   b. You can change the HTTP Proxy settings in the `provider.xml` file. For more information, see Section E.2.1, "Configuring the OmniPortlet Producer to Access Data Outside a Firewall".

   c. Propagate the changes made to the `provider.xml` file to middle tier **M2**:

      – Copy `OmniPortlet_WAR_DIR/WEB-INF/providers/omniPortlet/provider.xml` from **M1** to **M2**.

      – Copy `WebClipping_WAR_DIR/WEB-INF/providers/webClipping/provider.xml` from **M1** to **M2**.

3. Restart middle tier **M2**.

4. Update portlet producer registration in your WebCenter application. Change the first part of the producer registration URL from `http://m1.abc.com:7777/` to `http://lbr.abc.com/`.

5. Verify that OmniPortlet and the Web Clipping producers work properly through the LBR, by going to the test pages at the following URLs.

   ■ OmniPortlet Producer:
   `http://lbr.abc.com/portalTools/omniPortlet/producers/omniPortlet`

If you see the "No Portlets Available" message under the Portlet Information section in the OmniPortlet Producer test page, then you may not have configured OmniPortlet correctly in Step 1. If OmniPortlet is configured correctly, then the OmniPortlet and Simple Parameter Form portlets are available on the test page.

- Web Clipping Producer:
`http://lbr.abc.com/portalTools/webClipping/producers/webClipping`

Note: If you want to use the OracleAS Web Clipping portlet, or the Web Page Data Source for OmniPortlet, then you must also enable session binding in Oracle Web Cache.

E.3 Web Clipping Portlet Configuration Tips

Before you use Web Clipping, you must perform some administrative tasks, including the following:

- Web Clipping Repository Configuration
- HTTP or HTTPS Proxy Configuration
- Web Clipping Producer Security

E.3.1 Web Clipping Repository Configuration

Web clippings have definitions that must be stored persistently in an Oracle Metadata Services (MDS) store or an Oracle database.

You can view the Web Clipping repository configuration by accessing the Web Clipping Producer Test Page at:

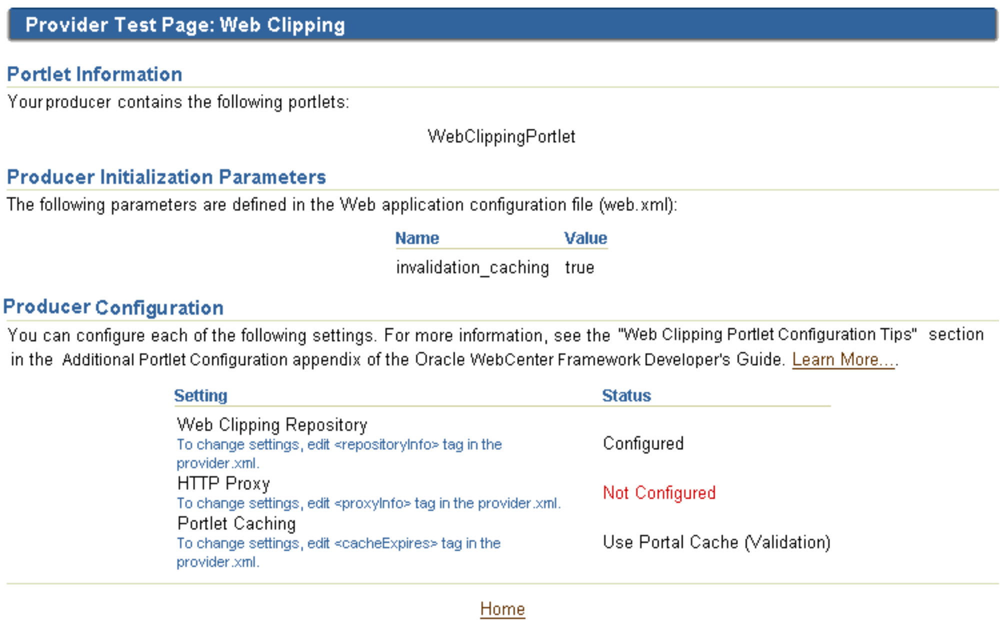
```
http://host:port/portalTools/webClipping/providers/webClipping
```

Where, *`host`* is the server to which your Web Clipping producer has been deployed and *`port`* is the port to which the server is listening for HTTP requests.

Note: Integrated WebLogic Server (WLS) and the `WLS_Portlet` managed server are deployed to different ports even if they may be available on the same system. By default, Integrated WLS is deployed to port 7101 and `WLS_Portlet` is deployed to port 8889.

The Provider Test Page automatically detects whether or not the Web Clipping producer is configured with a valid repository. If it is not, then the **Status** column for the Web Clipping Repository displays **Not Configured**. Figure E–1 shows the Provider Test Page of Web Clipping.

Figure E–1 Web Clipping - Provider Test Page



Provider Test Page: Web Clipping

Portlet Information

Your producer contains the following portlets:

WebClippingPortlet

Producer Initialization Parameters

The following parameters are defined in the Web application configuration file (web.xml):

Name	Value
invalidation_caching	true

Producer Configuration

You can configure each of the following settings. For more information, see the "Web Clipping Portlet Configuration Tips" section in the Additional Portlet Configuration appendix of the Oracle WebCenter Framework Developer's Guide. Learn More....

Setting	Status
Web Clipping Repository To change settings, edit <repositoryInfo> tag in the provider.xml.	Configured
HTTP Proxy To change settings, edit <proxyInfo> tag in the provider.xml.	Not Configured
Portlet Caching To change settings, edit <cacheExpires> tag in the provider.xml.	Use Portal Cache (Validation)

Home

You cannot change the Web Clipping configuration information by using the Provider Test Page. You can configure the Web Clipping repository by setting appropriate values in the `provider.xml` file. In this file, you use the `repositoryInfo` tag to specify the Web Clipping repository settings.

Note: To determine the path of `provider.xml`, search for `webClipping/provider.xml` in the file system. For example, run the following command in UNIX:

```
> find <DOMAIN_DIR> -name "provider.xml" | grep -i webClipping
```

In Oracle JDeveloper's Integrated WLS, Web Clipping's `provider.xml` is located here:

`JDEV_SYSTEM_DIRECTORY``/DefaultDomain/servers/DefaultServer/tmp/_WL_user/portalTools_11.1.1.1.0/`*`RANDOMLY_GENERATED_DIRECTORY`*`/war/WEB-INF/providers/webClipping/provider.xml`

In the Application Server 11*g* installation, Web Clipping's `provider.xml` is located here:

`FMW_HOME``/user_projects/domains/wc_domain/servers/WLS_Portlet/tmp/_WL_user/portalTools_11.1.1.1.0/`*`RANDOMLY_GENERATED_DIRECTORY`*`/war/WEB-INF/providers/webClipping/provider.xml`

E.3.1.1 Using Oracle Metadata Services (MDS) as the Web Clipping Repository

By default, in Oracle JDeveloper, the Web Clipping producer hosted on Integrated WLS, the default server, is configured to use MDS, which is file-based, as the Web Clipping repository.

Note: In a full Oracle Fusion Middleware installation, the Web Clipping portlet producer is also included within the `WLS_Portlets` managed server in the default domain. By default, this Web Clipping portlet producer is configured to use the Oracle database, which is installed as part of Oracle WebLogic Server, as the Web Clipping repository.

Example E–1 shows MDS as the default repository in `provider.xml`.

Example E–1 MDS as the Default Web Clipping Repository in provider.xml

```
<repositoryInfo class="oracle.portal.wcs.provider.info.MdsInformation">
  <mdsConfigLocation>mds-config.xml</mdsConfigLocation>
</repositoryInfo>
```

For an MDS repository, the `repositoryInfo` tag is set to the `MdsInformation` class. The `mdsConfigLocation` entry specifies the path to the `mds-config.xml` file, which contains the metadata store configuration information, including the path to the actual metadata store. In Oracle JDeveloper, the `mds-config.xml` file is located at the following path:

```
JDEV_SYSTEM_DIRECTORY/DefaultDomain/servers/DefaultServer/tmp/_WL_
user/portalTools_11.1.1.1.0/RANDOMLY_GENERATED_DIRECTORY/war/WEB-INF
```

The `mds-config.xml` file specifies the location of the repository in a property tag:

```
<property name="metadata-path" value="portletdata/tools/webClipping"/>
```

The location specified for `value` is relative to *`JDEV_HOME`*`/portal`. Any relative path specified is interpreted to be relative to *`JDEV_HOME`*`/portal`. If you want to use another location, for example, a location outside the Oracle JDeveloper home, then specify an absolute path, such as `c:\mds`.

Note: For a multiple middle tier deployment, change the metadata path to a shared file system.

E.3.1.2 Using an Oracle Database as the Web Clipping Repository

Although MDS is the default repository in Oracle JDeveloper, you can alternatively select to use a database schema for your Web Clipping repository. You can use either of the following database schemas for Web Clipping:

- The default `PORTLET` database schema created by RCU for Oracle WebLogic Server
- A user-defined database schema for Oracle 9*i* or later

E.3.1.2.1 Using the Database Schema Created by RCU For Web Clipping repository, you can use the Oracle database installed as part of Oracle WebLogic Server, through a JNDI data source named `jdbc/portletPrefs`. To access the database, the schema named `PORTLET` is used.

Note: The `PORTLET` schema must be created in the Oracle database configured for Oracle WebCenter. For more information, see the “Installing WebCenter" chapter in the *Oracle Fusion Middleware Installation Guide for Oracle WebCenter*.

E.3.1.2.2 Creating a Database Schema You can use any user-defined schema for an Oracle 9*i* or later database as the Web Clipping repository. To create a database schema for Web Clipping definitions and clippings, run the Java command described in Example E–2.

Example E–2 Java Command for Creating a Schema for Web Clipping Portlet Definitions and Clippings

```
ORACLE_HOME/jdk/bin/java -classpath ORACLE_HOME/lib/xmlparserv2.jar:
ORACLE_HOME/jdbc/lib/ojdbc14.jar:ORACLE_HOME/portal/jlib/wce.jar
oracle.portal.wcs.Installer -installSchema -username dbuser -password
dbpassword -dburl jdbc:oracle:thin:@//host:port/dbid
```

Where:

- *`ORACLE_HOME`* is the path to your Oracle home directory
- *`dbuser`* is the database user for the schema

  Consider using the same database user you use to create the WSRP and PDK-Java preference store database schema. If you do not use the same user, then you must create a new user and grant it connect and resource privileges.
- *`dbpassword`* is the specified user's password
- *`dburl`* is the URL for the database

  This is the database that contains the schema that you create for Web Clipping portlet definitions and clippings by using the following syntax:

  ```
  jdbc:oracle:thin:@//dbhost:dbport/service_name
  ```

  For example:

  ```
  jdbc:oracle:thin:@//shobeen:1521/sales_us
  ```

Note: The classpath in Example E–2 uses different separators for UNIX and Windows. On UNIX systems, the `classpath` uses a colon (:) separator. On Windows systems, the `classpath` uses a semicolon (;) separator.

E.3.1.3 Configuring Web Clipping Repository in provider.xml

To change the repository configuration for the Web Clipping producer deployed to Integrated WLS:

1. Open the `provider.xml` file in a text editor.
2. Specify one of the following settings for your Web Clipping repository:
   - Use the `PORTLET` schema referred by the JNDI data source created by RCU. Specify the entries shown in Example E–3.

Example E–3 Oracle Database as the Web Clipping Repository

```
<repositoryInfo class="oracle.portal.wcs.provider.info.JdbcDbInformation">
   <connectionName>jdbc/portletPrefs</connectionName>
   <useRAA>false</useRAA>
   <useASO>false</useASO>
</repositoryInfo>
```

For information about tag parameters, see Section E.3.1.4, "Attributes and Child Tags of the repositoryInfo Tag."

- Use the database schema, created for an Oracle database 9*i* or later, where you can manually specify connection information.

  To specify Oracle 9*i* or later database as the Web Clipping repository, specify database connection parameters in the entry shown in Example E–4.

Example E–4 Setting Oracle Database 9i or Later as Web Clipping Repository

```
<repositoryInfo class="oracle.portal.wcs.provider.info.DatabaseInformation">
   <useRAA>false</useRAA>
   <databaseHost>dbhost.mycompany.com</databaseHost>
   <databasePort>1521</databasePort>
   <databaseSid>iasdb</databaseSid>
   <databaseUsername>scott</databaseUsername>
   <databasePassword>!tiger</databasePassword>
   <useASO>false</useASO>
</repositoryInfo>
```

For information about tag parameters, see Section E.3.1.4, "Attributes and Child Tags of the repositoryInfo Tag."

If you require a secure database connection, then enable the Advanced Security Option (ASO) by setting the `useASO` entry to `true`. For more information about configuring ASO, see Section E.3.3, "Web Clipping Producer Security".

Note: Only one entry of `<repositoryInfo>` can exist in `provider.xml`. Depending on the Web Clipping repository you choose, you must uncomment only that entry.

3. Save the `provider.xml` file.
4. Restart Integrated WLS.

E.3.1.4 Attributes and Child Tags of the repositoryInfo Tag

Table E–6 lists and describes the attributes and child tags of the `repositoryInfo` tag.

Note: The attributes and child tags of the `repositoryInfo` tag are also described in the comments in the Web Clipping `provider.xml` file.

In the comments in the `provider.xml` file, the example provided for *Oracle9i Database or later using Oracle infrastructure database* is specific to Oracle Portal and its infrastructure database and application programming interfaces. That example should not be used for WebCenter application implementations.

Table E–6 Attributes and Child Tags of the repositoryInfo Tag

Attributes/Parameter	MDS/Database	Description
`class`	Both	The `class` attribute specifies the type of repository used to store Web Clipping definitions. The possible values for this attribute are: ■ `oracle.portal.wcs.provider.info.MdsInformation` This value signifies that MDS is used to store Web Clipping definitions and that MDS configuration is pushed to the `mds-config.xml` file. ■ `oracle.portal.wcs.provider.info.DatabaseInformation` This value signifies that an Oracle9*i* Database or later is used to store Web Clipping definitions and that the database connection details are included as children in the `repositoryInfo` tag. ■ `oracle.portal.wcs.provider.info.JdbcDbInformation` This value signifies that Oracle database installed as part of Oracle WebLogic Server is used to store Web Clipping definitions.
`mdsConfigLocation`	MDS	Use the `mdsConfigLocation` tag when the value for the `class` attribute indicates an MDS repository. It points to the MDS configuration file `mds-config.xml`, which specifies the actual MDS location. The `mds-config.xml` file is located at: *`ORACLE_HOME`*`/wlserver_10.3/wc_domain/servers/WLS_Portlet/tmp/_WL_user/portalTools_11.1.1.1.0/yyggl7/war/WEB-INF`
`connectionName`	Database	Specifies the JNDI name of the data source where the Web Clipping repository has been installed using the RCU. By default, the connection name is `jdbc/portletPrefs`, which points to the `PORTLET` schema in Oracle WebLogic Server 11g. In an interoperability scenario where an Oracle WebLogic Server 11*g* Middle Tier is paired with an Oracle Application Server 10*g* repository, the connection points to the `PORTAL` schema in the Oracle Application Server 10*g* repository.
`useASO`	Database	Set to `true` or `false`: ■ Specify `true` if you want to use ASO to encrypt the communication channel between the Web Clipping producer and the database. This is provided for introducing added security if case-sensitive data is contained in the clipped content. ■ Specify `false` to omit this option.
`useRAA`	Database	Set to `true` or `false`: ■ Specify `true` if Repository Access APIs will be used to access the database connection parameters. Specifying `true` is equivalent to making the Web Clipping producer use the default Oracle infrastructure database as the repository. Specifying `true` removes the need for other `repositoryInfo` child tags. ■ Specify `false` to omit this option.
`databaseHost`	Database	Specify the host name of the Oracle database. Use only version 9*i* or later. For example: `mycompany.dbhost.com`

Table E–6 (Cont.) Attributes and Child Tags of the repositoryInfo Tag

Attributes/Parameter	MDS/Database	Description
`databasePort`	Database	Specify the port number of the Oracle database listener. This is usually `1521`.
`databaseSid`	Database	Specify the Oracle SID of the database that hosts the Web Clipping repository.
`databaseUsername`	Database	Enter the user name used to log in to the database.
`databasePassword`	Database	Enter a plain text password for the specified database username. Prefix the password with an exclamation point (!) to allow the Web Clipping producer to encrypt the password once the producer starts. For example: `!AX3tR`

E.3.2 HTTP or HTTPS Proxy Configuration

Your HTTP or HTTPS proxy settings must be set to allow the Web Clipping Studio to connect to web sites outside of your firewall. You can specify the settings by manually editing the `provider.xml` file.

As the WebCenter Application administrator, you can set proxy settings manually according to your HTTP or HTTPS configuration. Edit the appropriate entries in the `provider.xml` file.

Example E–5 shows the relevant portion of `provider.xml`.

Example E–5 Proxy Settings

```
- <!--
 proxy information: Fill the following up if you have a proxy
 server between the provider and external sites.
   <proxyInfo class="oracle.portal.provider.v2.ProxyInformation">
      <httpProxyHost>proxy.mycompany.com</httpProxyHost>
      <httpProxyPort>80</httpProxyPort>
      <dontProxyFor>*.mycompany.com</dontProxyFor>
      <proxyUseAuth>true</proxyUseAuth>
      <proxyType>Basic</proxyType>
      <proxyRealm>realm1</proxyRealm>
      <proxyUseGlobal>false</proxyUseGlobal>
      <proxyUser>scott</proxyUser>
      <proxyPassword>!tiger</proxyPassword>
   </proxyInfo>

  -->
```

It is optional to specify values for the following tags: `<proxyUseAuth>`, `<proxyType>`, `<proxyRealm>`, `<proxyUseGlobal>`, `<proxyUser>`, and `<proxyPassword>`.

Table E–3, available earlier in this appendix, describes the proxy settings you specify in the `provider.xml` file. The descriptions in the table are applicable for Web Clipping producers also.

Note: For environments that use a proxy server to reach external web sites, you can use the `dontProxyFor` entry to specify the proxy exception list. Web Clipping uses the proxy exception list to restrict users from clipping content from unauthorized external web sites. Users attempting to reach a web site in one of the listed domains, from the Web Clipping Studio, will then receive an HTTP timeout error.

E.3.3 Web Clipping Producer Security

The preceding sections described the administrative tasks that must be performed before you are able to use the Web Clipping producer. The following sections describe some security configuration options that you must implement to enable the Web Clipping producer to access Secure Sockets Layer (SSL)-enabled web sites and encrypt the channel between itself and the database:

- Adding Certificates for Trusted Sites
- Configuring Oracle Advanced Security for the Web Clipping Producer

E.3.3.1 Adding Certificates for Trusted Sites

When a user navigates to a secure site, the web site typically returns a certificate, identifying itself to the user when asking for secure information. If the user accepts the certificate, then the certificate is placed into the list of trusted certificates of the browser so that a secure channel can be opened between the browser and that server. Like a web browser, the Web Clipping producer acts as an HTTP client to external web sites. To keep track of trusted sites, the Web Clipping producer uses the `cacerts` file, which is the Java keystore to store trusted certificates.

By default, the `cacerts` file stores various certificates including the Equifax, VeriSign, and Cybertrust certificates. However, it does not include all possible server certificates that exist on the web. For this reason, when a user navigates to a secure server using HTTPS, the user can get an SSL handshake failed exception in the Web Clipping Studio. To solve this problem, you must add the certificate of that site to the `cacerts` file.

To add a certificate to the `cacerts` file:

1. Download the certificate of the HTTPS web site and save the certificate in the PEM format.

   Tip: Mozilla Firefox 3.0 or later enable you to save a certificate file in the PEM format.

2. Locate the path to the `cacerts` file by using the following steps:

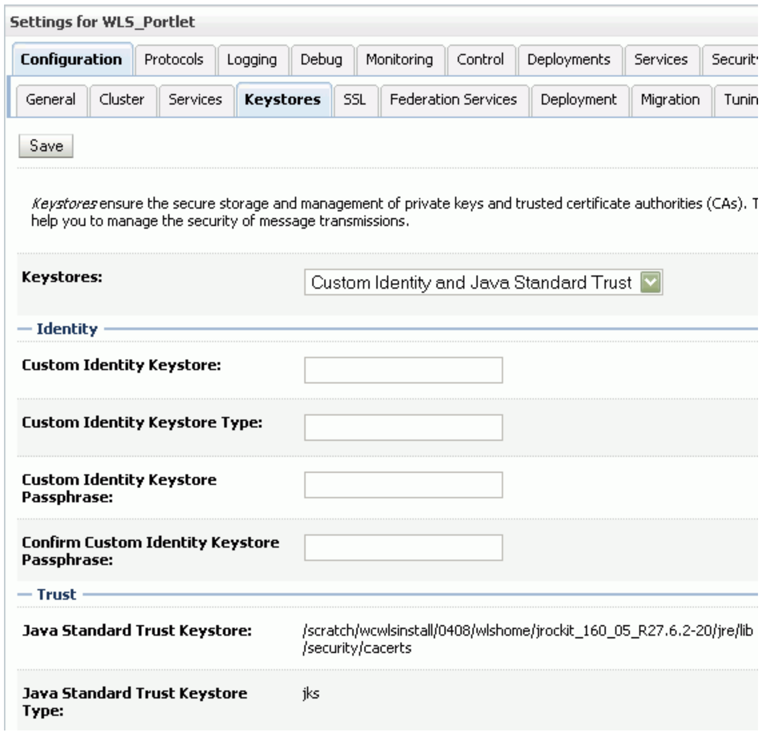
   a. Log on to the Oracle WebLogic Server Administration Console by using the following URL format:

      `http://`*`host:port`*`/console`

   b. Open the **Keystore** tab for the **WLS_Portlets** managed server.

   c. Note down the location of the `cacerts` file given in the **Java Standard Trust Keystore** field.

Figure E–2 Locating the cacerts file



Keystores ensure the secure storage and management of private keys and trusted certificate authorities (CAs). T
help you to manage the security of message transmissions.

Keystores: Custom Identity and Java Standard Trust

Identity

Custom Identity Keystore:

Custom Identity Keystore Type:

Custom Identity Keystore Passphrase:

Confirm Custom Identity Keystore Passphrase:

Trust

Java Standard Trust Keystore: /scratch/wcwlsinstall/0408/wlshome/jrockit_160_05_R27.6.2-20/jre/lib/security/cacerts

Java Standard Trust Keystore Type: jks

3. At the command prompt, navigate to the location of the `cacerts` file and run the following command to add the certificate:

   ```
   keytool -importcert -alias certi_alias -file certifi_name
   -keystore cacerts -storepass password
   ```

   Where, *`certi_alias`* refers to the alias used for the certificate, *`certifi_name`* refers to the certificate file name, and *`password`* refers to the password of the `cacerts` file. The default password is `changeit`.

   For example,

   ```
   keytool -importcert -alias stamf05 -file stamf05.crt
   -keystore cacerts -storepass changeit
   ```

   > **Tip:** It is advisable to import trusted certificate into the `cacerts` file by using an alias. It makes it easier for you to delete or list the entries in the keystore.

E.3.3.2 Configuring Oracle Advanced Security for the Web Clipping Producer

The Web Clipping producer can use Oracle Advanced Security Option (ASO) to secure and encrypt the channel between itself and the database that hosts the Web Clipping repository. This feature is available only if you have selected any Oracle database as the Web Clipping repository. This feature is disabled by default as Oracle Metadata Services is the default Web Clipping repository in Oracle JDeveloper. To enable ASO, perform the following steps:

1. Open `provider.xml` in a text editor.
2. Under the repository settings section in the file (shown in Example E–3), set the `useASO` entry to `true`.
3. Save the `provider.xml` file.

In addition, you must set the following ASO configuration parameters in the `sqlnet.ora` file to ensure that the database connections created between the Web Clipping producer and the database hosting the Web Clipping repository are using ASO.

- `SQLNET.AUTHENTICATION_SERVICES` -- This parameter is used to select a supported authentication method in making database connections with ASO. For more information about setting this parameter, see the *Oracle Advanced Security Administrator's Guide*.
- `SQLNET.CRYPTO_SEED` -- This parameter denotes the cryptographic seed value (FIPS 140-1 setting), used in making database connections with ASO.

  For more information about setting parameters and about the values to use for various possible combinations of parameters, see the *Oracle Advanced Security Administrator's Guide*.

> **Note:** When setting these parameters after the initial configuration (where the database parameters are already set up), database connections are assumed to be open. Because enabling the ASO affects all connections made to the database, it is advisable to restart Integrated WLS containing the Web Clipping producer to reset all the current connections to use ASO. You must also do this when disabling ASO.

E.4 Setting Up a Preference Store

The portlet preference store is used for persisting consumer registration handles and portlet preference data. Portlet producers can use one of two types of preference store: *File* and *Database*.

In a clustered environment, Oracle recommends the use of a database preference store. If you prefer to use a file-based preference store, then you must also use a shared file system.

If you configure the WebLogic Server to run portlet applications, then the database preference store is already configured for you. For more information, see the Oracle Fusion Middleware Installation Guide for Oracle WebCenter.

E.4.1 WSRP Producers

WSRP producers have a JNDI preference value that determines which preference store is used, and is set in the `web.xml` file of the portlet application. Table E–7 lists and describes the JNDI variables used to specify the preference store for WSRP producers.

Table E–7 WSRP Producer Database Preference Store-Related JNDI Variable

Variable Name	Variable Value	Description
`oracle/portal/wsrp/server/persistentStore`	`Database`	Determines which data store (File or Database) is used for persisting a portlet application's consumer registration handles and portlet preferences. Allowable values: {`File`, `Database`}
`oracle/portal/wsrp/server/fileStoreRoot`	`portletdata`	Defines the path to the root directory to be used by the file preference store. Absolute paths are interpreted relative to the file system root. Relative paths are interpreted relative to the `ORACLE_HOME\portal` directory. Note that all producers running within the same WebLogic Server must use the same path for this variable. Otherwise, you will get a `Portlet unavailable` error for some portlets.

The following examples show the entries in a producer's `web.xml` file.

Example E–6 Configuring web.xml to Use a Database Preference Store

```
<env-entry>
   <env-entry-name>oracle/portal/wsrp/server/persistentStore</env-entry-name>
   <env-entry-type>java.lang.String</env-entry-type>
   <env-entry-value>Database</env-entry-value>
</env-entry>
```

Example E–7 Configuring web.xml to Use a File-Based Preference Store

```
<env-entry>
  <env-entry-name>oracle/portal/wsrp/server/persistentStore</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>File</env-entry-value>
</env-entry>
<env-entry>
  <env-entry-name>oracle/portal/wsrp/server/fileStoreRoot</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>myPrefStore</env-entry-value>
</env-entry>
```

Once you have updated the relevant `web.xml` files, restart your WebLogic Server.

E.4.2 PDK-Java Producers

The type of preference store used for PDK-Java producers is determined by the `preferenceStore` tag in the `provider.xml` file. Table E–8 lists and describes the attributes and parameters used with the `preferenceStore` tag when a database preference store is specified.

Table E–8 Attributes and Parameters of the preferenceStore Tag

Attribute/Parameter	Description

`class`	This required attribute specifies the Java class that defines the location and other details of portlet preferences. For example, a file-based preference store might use:
	`<preferenceStore class="oracle.portal.provider.v2.preference.FilePreferenceStore">`
	A database preference store might use:
	`<preferenceStore class="oracle.portal.provider.v2.preference.DBPreferenceStore">`
	OmniPortlet uses its own class, for example:
	`<preferenceStore class="oracle.webdb.reformlet.ReformletFilePreferenceStore">`

`name`	This required parameter provides a name for the preference store. Use any value you choose. For example:
	`<preferenceStore class="oracle.portal.provider.v2.preference.DBPreferenceStore">`
	    `<name>MyPDKProducerPreferenceStore</name>`
	`</preferenceStore>`

`connection`	This required parameter for a database preference store points to a JNDI connection that connects with a schema containing the portlet preference store. For example:
	`<preferenceStore class="oracle.portal.provider.v2.preference.FCFDBPreferenceStore">`
	    `<name>MyPDKProducerPreferenceStore</name>`
	    `<connection>java:comp/env/jdbc/portletPrefs</connection>`
	`</preferenceStore>`

rootDirectory	This optional parameter specifies the location where the file-based preference store preferences are stored. When this parameter is not included with the `preferenceStore` tag, the default file-based preference store location is used. This is the same folder where the producer's provider definition file is stored

useHashing	This optional parameter is used when specifying a file based preference store, and accepts the value `true` or `false`. When it is set to `true`, each preference data file is stored in an extra subdirectory with a name determined by hashing the data file name. Using this parameter can improve file system performance by limiting the number of preference data files stored in a single directory.
	For example:
	`<preferenceStore class="oracle.portal.provider.v2.preference.FilePreferenceStore">`
	  `<name>PDKProducerPreferenceStore</name>`
	  `<useHashing>true</useHashing>`
	`</preferenceStore>`

E.5 Portlet Preference Store Migration Utilities

A preference store is a mechanism for storing information like user preference data, portlet and producer settings, or even portlet data. Preferences can be stored in a database, which is recommended for high availability configurations, or a file system. You can migrate the following stores for your WebCenter applications:

- JPS Portlet Preference Store - PersistenceMigrationTool

- PDK-Java Portlet Preference Store - Migration and Upgrade Utilities
- Web Clipping Repository

E.5.1 JPS Portlet Preference Store - PersistenceMigrationTool

A WSRP container preference store is a mechanism used for persisting consumer registration and portlet preference data. Currently, there are two preference store implementations, *database preference store* and *file-based preference store*. A database preference store persists data using a relational database. A file-based preference store persists data using the file system. The file-based preference store may be used for testing, since it removes the dependency on a database. But, for highly available configurations, Oracle recommends that you use the database preference store.

The WSRP container preference store migration utility, `PersistenceMigrationTool`, enables you to migrate existing data between different preference stores (for example, from a database preference store to a file preference store). This utility also enables upgrading users to ensure that their existing locale-specific portlet preference data uses a naming format compatible with the latest JPS release. You can also use this utility to migrate between source and destination stores of the same type. This enables data to be moved from one database store to another.

The syntax of the `PersistenceMigrationTool` is:

Note: You must set the classpath when running this tool to include `wsrp-container.jar.oracle.portlet-api.jar` and `ojdbc6.jar`. For example:

```
java  -classpath \
  ORACLE_HOME/webcenter/modules/oracle.portlet.server_
11.1.1/oracle-portlet-api.jar:ORACLE_
HOME/webcenter/modules/oracle.portlet.server_
11.1.1/wsrp-container.jar:$WL_HOME/server/lib/ojdbc6.jar \
  oracle.portlet.server.containerimpl.PersistenceMigrationTool
```

```
java oracle.portlet.server.containerimpl.PersistenceMigrationTool
-sourceType file | db
-destType file | db
{-sourcePath dir |
 -sourceUsername username -sourcePassword password -sourceDatabase db}
{-destPath dir | destUsername username -destPassword password -destDatabase db}
[-debug]
```

where

`sourceType` indicates whether the source store is in a file or database. You can have source and destination stores of the same type. Hence, you can migrate from one database to another or one file system to another.

`destType` indicates whether the destination store is in a file or database. You can have source and destination stores of the same type. Hence, you can migrate from one database to another or one file system to another.

`sourcePath` is the location of a file-based preference store. This argument is required when `sourceType` is `file`.

`sourceUsername` is the database user name for a preference store database. This argument is required when `sourceType` is `db`.

`sourcePassword` is the database password for a preference store database. This argument is required when `sourceType` is `db`.

`sourceDatabase` is the name of a preference store database. This argument is required when `sourceType` is `db`.

`destPath` is the location of a file-based preference store. This argument is required when `destType` is `file`.

`destUsername` is the database user name for a preference store database. This argument is required when `destType` is `db`.

`destPassword` is the database password for a preference store database. This argument is required when `destType` is `db`.

`destDatabase` is the name of a preference store database. This argument is required when `destType` is `db`.

`debug` turns on full logging through standard output to enable users to diagnose issues that arise when the tool runs.

Example E–8 demonstrates running the `PersistenceMigrationTool` utility. In this example, preferences from a database store are copied to a file store.

Example E–8 Running the PersistenceMigrationTool Utility

```
java  -classpath ORACLE_HOME/webcenter/modules/oracle.portlet.server_
11.1.1/oracle-portlet-api.jar:ORACLE_HOME/webcenter/modules/oracle.portlet.server_
11.1.1/wsrp-container.jar:WL_HOME/server/lib/ojdbc6.jar \
 oracle.portlet.server.containerimpl.PersistenceMigrationTool
-sourceType db \
-sourceUsername scott \
-sourcePassword tiger \
-sourceDatabase abc.mycompany.com:1521:yourdatabase \
-destType file \
-destRoot /data/prefs
```

E.5.2 PDK-Java Portlet Preference Store - Migration and Upgrade Utilities

PDK-Java has two `PreferenceStore` implementations, `DBPreferenceStore` and `FilePreferenceStore`. `DBPreferenceStore` persists data using a JDBC-compatible relational database and `FilePreferenceStore` persists data using the file system.

If you have already installed Oracle PDK, then you can manage the information stored in the preference store by using the Preference Store Migration and Upgrade Utility, which is included in the `pdkjava.jar` file. Note that you must run this tool from `ORACLE_HOME`. The syntax of the migration utility is:

Note: You must set the classpath when running this tool to include `pdk-java.jar.ptlshare.jar` and `ojdbc14dms.jar`. For example:

```
java -classpath ORACLE_
HOME/webcenter/modules/oracle.portlet.server_
11.1.1/pdkjava.jar:ORACLE_
HOME/webcenter/modules/oracle.portlet.server_
11.1.1/ptlshare.jar:ORACLE_HOME/ucm/shared/classes/ojdbc14.jar \
  oracle.portal.provider.v2.preference.MigrationTool
```

```
java oracle.portal.provider.v2.preference.MigrationTool
  -mode [file | db | filetodb | filetofile | dbtofile | dbtodb]
  [-remap language | locale]
  [-countries iso_country_code]
  [-pref1UseHashing true | false]
  {-pref1RootDirectory directory |
   -pref1User username -pref1Password password -pref1URL url}
  [-pref1UseHashing true | false]
  {-pref2RootDirectory directory |
   -pref2User username -pref2Password password -pref2URL url}
  [-upfixwpi filename]
```

where

`-mode` is the mode in which you want to run the Preference Store Migration and Upgrade Utility

- `filetodb`, `filetofile`, `dbtofile`, or `dbtodb` indicates that you want to run in migration mode. See Section E.5.2.1, "Migration Mode" for more information about this mode.
- `file` or `db` indicates that you want to run in upgrade mode. See Section E.5.2.2, "Upgrade Mode" for more information about this mode.

`-remap` is the `localePersonalizationLevel` (language or locale). Note that you only must use this option if you want to change `localePersonalizationLevel` as part of your upgrade or migration.

`-countries` specifies a prioritized list of ISO country codes, indicating your order of preference in a collision between remapped preferences for different countries. `-countries` is only meaningful if you also specified the `-remap` option.

`-pref1UseHashing` specifies whether you want to employ hashing on the source for this operation.

`-pref1RootDirectory` is the path of a source file system, for example, `j2ee/home/applications/jpdk/jpdk/WEB-INF/providers/sample`.

`-pref1User` is the user name for a source database.

`-pref1Password` is the password for a source database.

`-pref1URL` is the URL to a source database, for example, `jdbc:oracle:thin:@myserver.mydomain.com:1521:mysid`.

`-pref2UseHashing` specifies whether you want to employ hashing on the destination for this operation.

`-pref2RootDirectory` is the path of a destination file system, for example, `j2ee/home/applications/jpdk/jpdk/WEB-INF/providers/sample`.

`-pref2User` is the user name for a destination database.

`-pref2Password` is the password for a destination database.

`-pref2URL` is the URL to a destination database, for example, `jdbc:oracle:thin:@myserver.mydomain.com:1521:mysid`.

`- upfixwpi` indicates a log file for the operation.

E.5.2.1 Migration Mode

Use a migration mode to copy data from a source preference store to a target preference store. When the utility is run in this mode, the preference stores for all the portlet definitions are updated.

Table E–9 describes the migration modes in which you can run the utility.

Table E–9 Migration Modes in Which to Run the Utility

Mode	Description
`filetodb`	Use when data must be copied from a `FilePreferenceStore` to a `DBPreferenceStore`.
`filetofile`	Use when data must be copied from one `FilePreferenceStore` to another `FilePreferenceStore` that is in a different location
`dbtofile`	Use when data must be copied from a `DBPreferenceStore` to a `FilePreferenceStore`.
`dbtodb`	Use when data must be copied from one `DBPreferenceStore` to another `DBPreferenceStore` that is based on a different database table.

If the destination for the operation is a database, you must ensure that the destination WebLogic Server is created using the WebCenter portlet template and the appropriate schemas have been created using the RCU. For more information, see *Oracle Fusion Middleware Installation Guide for Oracle WebCenter*.

When using a migration mode, you can also use the `-remap` and `-countries` options to specify that the data should be upgraded in the course of being migrated. Specifically, you use these options to ensure that the locale-specific preferences are appropriately remapped.

You can use the other options accepted by the utility to specify the properties of the preference stores involved in the upgrade or migration process. These options must correspond to the tags you specify in `provider.xml` to describe the preference stores. For more information about the properties you can set on a preference store, see the "PDK-Java XML Provider Definition Tag Reference" document on Portal Studio:

`http://www.oracle.com/technology/products/ias/portal/html/javadoc/apidoc/oracle/portal/provider/v2/ProviderDefinition.html`

Properties beginning with the prefix `-pref1` correspond to properties of the source preference store (in an upgrade mode, this is the only preference store). For example, specifying `-pref1UseHashing true -pref1RootDirectory j2ee/home/applications/jpdk/jpdk/WEB-INF/providers/sample` would set the `useHashing` and `rootDirectory` properties of a source `FilePreferenceStore`.

When one of the migration basic modes is selected, properties beginning with the prefix `-pref2` correspond to properties of the target preference store. For example, specifying `-pref2User portlet_prefs -pref2Password portlet_prefs -pref2URL jdbc:oracle:thin:@myserver.mydomain.com:1521:mysid` would set the database connection details on a target `DBPreferenceStore`.

Example E–9 PDK-Java Migration Utility Command Line, Migration

```
java -classpath ORACLE_HOME/webcenter/modules/oracle.portlet.server_
11.1.1/pdkjava.jar:ORACLE_HOME/webcenter/modules/oracle.portlet.server_
```

```
11.1.1/ptlshare.jar:ORACLE_HOME/ucm/shared/classes/ojdbc14.jar \
oracle.portal.provider.v2.preference.MigrationTool \
-mode dbtofile \
-pref1User portlet_prefs \
-pref1Password portlet_prefs \
-pref1URL jdbc:oracle:thin:@myserver.mydomain.com:1521:mysid \
-pref2RootDirectory /mydirectory/preferences
```

E.5.2.2 Upgrade Mode

Use an upgrade mode to upgrade data in place, and to modify existing locale-specific preferences in the preference store so that the naming format used is compatible with the current version of Oracle Portal and a given `localePersonalizationLevel` setting.

Table E–10 describes the upgrade modes in which you can run the utility.

Table E–10 Upgrade Modes in Which to Run the Utility

Mode	Description
`file`	Use when data in a `FilePreferenceStore` must be upgraded.
`db`	Use when data in a `DBPreferenceStore` must be upgraded.

An upgrade mode can be used in the following scenarios:

- You have upgraded from Oracle PDK 9.0.4.0.0 or earlier and want to use existing portlets with the default `localePersonalizationLevel` setting of language (In earlier releases, the default setting was `locale`).
- You have upgraded from Oracle Portal 9.0.2.0.0 or earlier and want to use existing portlets with a `localePersonalizationLevel` setting of `locale` (Oracle Portal now uses different names for some locales and therefore some existing data must be remapped).
- You want to change the `localePersonalizationLevel` for an existing portlet from `locale` to `language` or vice-versa.

When using an upgrade mode, you must use the `-remap` option to specify the `localePersonalizationLevel` (`language` or `locale`) that you are upgrading to. You can also use the `-countries` option to specify a prioritized list of ISO country codes, indicating your order of preference in a collision between remapped preferences for different countries. For example, if you specify `-remap language -countries GB,US` in the command, then it means that if the utility comes across both US English and British English preferences (`en_US` and `en_GB`) in a given preference store, it will remap the British English preference to become the English-wide preference (`en`).

Example E–10 PDK-Java Migration Utility Command Line, Upgrade

```
java -classpath ORACLE_HOME/webcenter/modules/oracle.portlet.server_
11.1.1/pdkjava.jar:ORACLE_HOME/webcenter/modules/oracle.portlet.server_
11.1.1/ptlshare.jar:ORACLE_HOME/ucm/shared/classes/ojdbc14.jar \
  oracle.portal.provider.v2.preference.MigrationTool \
 -mode file -remap language
 -countries GB,US -pref1UseHashing true
 -pref1RootDirectory j2ee/home/applications/jpdk/jpdk/WEB-INF/providers/sample
```

E.5.3 Web Clipping Repository

Web Clipping does not have a preference store as such, but it stores Web Clipping definitions and associated metadata. By default, it uses MDS, which is file-based, for this purpose. But you can configure Web Clipping to use a database. To migrate this repository for WebCenter applications, you can use the Predeployment tool in export and import mode to go from MDS to a database, or vice versa. This procedure must be done for each application as follows:

1. Run the Predeployment tool in export mode on all WebCenter applications that use the Web Clipping producer. For more information, see Section 30.1.3, "How to Implement Export/Import of Customizations (WSRP 2.0)."
2. Update the producer to use a different repository. For information, see Section E.3.1.3, "Configuring Web Clipping Repository in provider.xml."
3. Run the Predeployment tool in `import` mode on all WebCenter applications that use the Web Clipping producer. For more information, see Section 30.1.3, "How to Implement Export/Import of Customizations (WSRP 2.0)."

E.5.4 Moving a Producer

In some situations, you may need to move a portlet producer. To do so, you must:

1. Install the new producer.
2. Make the preference store of the original producer available to the new producer by performing one of the following tasks:
   - Migrate the preference store using the Preference Store Migration and Upgrade Utilities (see Section E.5.2, "PDK-Java Portlet Preference Store - Migration and Upgrade Utilities" for more information), or
   - Configure the new producer to use the same preference store as the original producer. If the database preference store is being used, then you must point the WebLogic Server data source to the same database as the original producer.
3. Update the URL of the producer registration by using Enterprise Manager Fusion Middleware Control or the WLST commands: `setWSRPProducerRegistration` or `setPDKJavaProducerRegistration`.

F Reuse of Oracle Portal Components

This appendix describes how to reuse components from your existing Oracle Portal application in a custom WebCenter application built with Oracle WebCenter Framework.

This appendix includes the following sections:

- Section F.1, "Introduction to Oracle Portal Components"
- Section F.2, "Reusing Portlets"
- Section F.3, "Reusing Items"

F.1 Introduction to Oracle Portal Components

The two most basic building blocks of Oracle Portal are as follows:

- Portlets are the fundamental building blocks of a portal page. Portlets can be deployed to Oracle Portal through three producer (known in Oracle Portal as provider) types:
  - **Web (or Oracle PDK-Java) producers** use open standards, such as XML, SOAP, HTTP, or J2EE for deployment , definition, and communication with applications. The Oracle PDK-Java provides a framework that simplifies the task of building Web producers. For more information, see Section 27.2.3.1, "PDK-Java Producers."
  - **WSRP producers** serve as the containers for standards-based (JSR 168) portlets. For more information, see Section 27.2.3.2, "WSRP Producers."
  - **Database producers** own one or more database portlets. Examples of database portlets include, portlets built using the Oracle PDK-PL/SQL, Portlet Builder portlets (charts, forms, reports, and so on), and Oracle Portal page portlets).

  In Oracle WebCenter Framework, you can make Web and WSRP portlets available by registering their producers with the application. For more information, see Section F.2.1, "How to Reuse JSR 168 and Oracle PDK-Java Portlets."

  You can also make database portlets available to custom WebCenter applications by using the Federated Portal Adapter. For more information, see Section F.2.8, "How to Use the Federated Portal Adapter to Reuse Database Portlets."

- Items are individual pieces of content (text, hyperlink, image, and so on) that reside 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 Oracle 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.

Because items are stored in a content repository, you can retrieve them with Java Content Repository (JCR) data controls. A default adapter for the Oracle Portal content repository is provided with Oracle WebCenter Framework. For more information, see Section F.3, "Reusing Items."

Given the technical differences between Oracle Portal and WebCenter, there is no direct upgrade available across the platform. However, key components of your portal can be exposed within WebCenter.

F.2 Reusing Portlets

You can make the portlets used in your portal available to your custom WebCenter applications.

F.2.1 How to Reuse JSR 168 and Oracle PDK-Java Portlets

To reuse JSR 168 and Oracle PDK-Java portlets in your custom WebCenter application, all you need to do is register the producers as you would any other producer.

To reuse JSR 168 and Oracle PDK-Java portlets:

1. In Oracle JDeveloper, access the WSRP or Oracle PDK-Java Producer Registration Wizard.
2. Step through the wizard providing information about the producer, including the URL endpoint.
3. Open the page to which you to add the portlet.
4. Drag the portlet from the registered producer to the appropriate part of the page.

For more detailed information about consuming portlets in a custom WebCenter application, see Chapter 9, "Consuming Portlets."

F.2.2 What You May Need to Know About Events

Oracle PDK-Java events are not supported in Oracle WebCenter Framework. If you have Oracle PDK-Java portlets that use events and you deploy them in an Oracle WebCenter Framework environment, then the events are ignored.

F.2.3 What You May Need to Know About Mobile Portlets

Mobile portlets are not supported in Oracle WebCenter Framework. If you have Oracle PDK-Java mobile portlets, then they do not work in a Oracle WebCenter Framework environment.

F.2.4 What You May Need to Know About the Portlet Chrome

In Oracle WebCenter Framework, the portlet does not provide the chrome. The consuming application provides the chrome. This behavior conforms to the WSRP convention, but it is a change from Oracle Portal, where Oracle PDK-Java provided the chrome for portlets. Hence, the Oracle PDK-Java portlet header is filtered out in the Oracle WebCenter Framework environment.

As part of this filtering process, Oracle WebCenter Framework parses the title area of PDK-Java portlets to preserve the title and add the necessary portlet mode links. The filtering algorithm is as follows:

- If the portlet has multiple tables, then the title area is considered to be the first table. If the portlet has only one table, then the title area is considered to be the first row in that table.
- The first text in the title area is considered to be the title. If the portlet has no tables or a discernible title, then the title is taken from the `adfp:portlet` component.
- Any link that contains the fragment `_mode=` *`parameter`* is considered to be a portlet mode link. These links are moved into the portlet menu by Oracle WebCenter Framework.

If you want to bypass the rendition of the header section of your portlet chrome and reuse its original, Oracle PDK-Java header, then you can set the `displayHeader` attribute of the `adfp:portlet` tag to `false` and `retainPortletHeader` as `true` in the associated portlet bindings of the page definition XML. Setting `retainPortletHeader=true` in the page definition portlet binding retains the portlet header in the portlet response. Setting `displayHeader=false` in the `adf:portlet` tag suppresses the application's header for the portlet chrome, and hides the icons and links normally displayed in the header.

F.2.5 What You May Need to Know About Personalizations and Customizations

Portlet customizations and personalizations that you or your users applied to your portlets in Oracle Portal are not carried over to your custom WebCenter application. In a custom WebCenter application, your Oracle Portal portlets are treated as new instances. Hence, previous personalizations and customizations are not available.

F.2.6 What You May Need to Know About Oracle Portal System Resources

Portlets built with a target consumer of Oracle Portal in mind tend to make invalid assumptions about the runtime environment. For example, the portlet might assume the presence of Oracle Portal resources, such as images or JavaScript functions. In Oracle WebCenter Framework, these resources are not available. Hence, you must bundle the resources the portlet needs with the producer to ensure that the portlet runs properly in the Oracle WebCenter Framework environment.

Another common issue with portlets designed for Oracle Portal consumption is the portlet assuming the Oracle Portal URL format and the presence of Oracle Portal parameters. The page URL of an Oracle ADF application is different from that of Oracle Portal. Therefore, portlet developers cannot make assumptions about the page URL and must prepare their Oracle PDK-Java portlets for execution in the Oracle WebCenter Framework environment by using the proper portlet APIs.

F.2.7 What You May Need to Know About Partner and External Applications

Partner applications are not supported. External applications are supported. Automatic login is supported through an external application automated login servlet, which is automatically configured in your application when you register a producer.

F.2.8 How to Use the Federated Portal Adapter to Reuse Database Portlets

The Federated Portal Adapter is a component of Oracle Portal that enables Oracle Portal instances to share their database portlets through the Web portlet interface. The Federated Portal Adapter receives the SOAP messages for a Web producer, parses the SOAP and then dispatches the messages to a database producer as PL/SQL procedure calls. In effect, the Federated Portal Adapter makes a database producer behave in the same way as a Web producer. You can make Oracle Portal database portlets, including

PL/SQL portlets, Portlet Builder portlets, and page portlets, available for use in your custom WebCenter applications.

Note: You can also expose Oracle Portal pages in WebCenter through the Federated Portal Adapter by publishing them as portlets in Oracle Portal.

Implementing the Federated Portal Adapter for your portal exposes the Oracle Portal portlets through the Oracle PDK-Java Web producer protocol making it possible to register them as PDK-Java producers. Registering the producer with a custom WebCenter application causes the portlets to appear in the Component Palette, and you can then drag and drop them onto pages as you would any other portlet. Without the Federated Portal Adapter, you cannot access these producers from a custom WebCenter application.

For more information about the Federated Portal Adapter, see the chapter "Using the Federated Portal Adapter" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

To register a Federated Portal Adapter producer with Oracle WebCenter:

1. Set up the Oracle Portal environment for the Federated Portal Adapter. For more information, see the section "Setting Up the Environment to Use the Federated Portal Adapter" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.
2. To register an instance of Oracle Portal through the Federated Portal Adapter, the portal must have a user named "PORTAL" in Oracle Internet Directory. To create this user:
    a. Log in to the Oracle Portal instance as the portal administrator user.
    b. In the **User** portlet, click **Create New Users.**

       By default, the **User** portlet is on the **Administer** tab of the **Portal Builder** page, on the **Portal** subtab.
    c. In the **Last Name** field, enter `PORTAL`.
    d. In the **User ID** field, enter `PORTAL`.
    e. In the **Password** and **Confirm Password** fields, enter an appropriate password.
    f. In the **Email Address** field, enter an appropriate email address.
    g. Click **Submit,** then **Done.**
    h. In the **Portal User Profile** portlet, enter `PORTAL` in the **Name** field, and click **Edit.**

       By default, the **Portal User Profile** portlet is on the **Administer** tab of the **Portal Builder** page, on the **Portal** subtab.
    i. In the **Default Group** field, enter `PORTLET_PUBLISHERS`.
    j. Click **OK.**
3. If you want to display Oracle Portal pages in your custom WebCenter applications, you must first publish each of those pages as portlets. To display Oracle Portal pages, edit the page properties and select **Publish As Portlet** on the

Optional tab. For more information, see the section "Placing One Page Onto Another" in the *Oracle Fusion Middleware User's Guide for Oracle Portal*.

4. Confirm that the users of the custom WebCenter application have appropriate privileges on the portlets and pages exposed through the Federated Portal Adapter, and that they map to the users of the Oracle Portal instance. The user name for a person on the custom WebCenter application should map to the user name of the same person in Oracle Portal. For example, JSMITH should represent the same person in both the custom WebCenter application and Oracle Portal.

5. In Oracle JDeveloper, register the producer as an Oracle PDK-Java producer using the Federated Portal Adapter URL and Service ID:

   ```
   http://host:port/adapter/dad/schema
   ```

   To verify the URL, enter it in a browser. If the URL is correct you should see the following message:

   ```
   Congratulations - you got to the adapter test page
   ```

   If you are logged in you should see a list of producers, their service IDs, and the portlets each provides. If you are logged in as an Oracle Portal administrator, you should also see additional details about page portlets and Oracle Portal HMAC registrations.

   When you register the producer, ensure that you select the **Establish Producer Sessions** check box on the Specify Connection Details page of the wizard. This is so that the WebCenter user information is correctly propagated to Oracle Portal.

   For more information, see Section 9.2.3, "How to Register an Oracle PDK-Java Portlet Producer."

6. You can drag and drop the producer's portlets onto a page.

7. To secure communication between the Oracle Portal instance and Oracle WebCenter, provide a shared key using Hash Message Authentication Code (HMAC):

   a. Go to SQL*Plus and log in as the portal schema owner.

   b. Run:

      ```
      @wwc/proadssr.sql http://www.oracle.com/adapter/portal
      ```

      Note that, for custom WebCenter applications, the adapter URL is always the same:

      ```
      http://www.oracle.com/adapter/portal
      ```

   c. Enter a shared key code.

   d. In Oracle JDeveloper, enter this shared key code into the **Shared Key** field on the Specify Additional Registration Details step of the Create/Edit Oracle PDK-Java Portlet Producer wizard or dialog.

      Note: To remove the shared key from the Oracle Portal instance use:

      ```
      @wwc/proadsdr.sql
      ```

8. Certain portlets built in Oracle Portal, such as page portlets and several sample database portlets, assume that the set of images shipped in the portal middle tier are available in the path `/images` in the Web server of the portal page. These

images are not available by default for your custom WebCenter applications. For these images to display correctly:

a. Download the following zip file:

   `http://download.oracle.com/otndocs/tech/portal/files/portal-images.zip`

b. Extract the contents of the zip file and mount the contents under the URL `/images`.

9. You can now run the page that contains the portlet.

F.2.9 What You May Need to Know About Troubleshooting the Federated Portal Adapter

The following information may help when trying to resolve issues with the Federated Portal Adapter in Oracle WebCenter.

F.2.9.1 On Registration

Error occurred trying to register producer

The shared key has not been set up in the Oracle Portal instance or it does not match that given in Oracle JDeveloper, or the service ID has not been entered.

Could not Contact Producer

Either the portal registration URL is not correct, the service ID given in Oracle JDeveloper does not match a provider in the Oracle Portal instance, the Oracle Portal instance could not be contacted (for example, it is behind a firewall or is not available), or the users in Oracle JDeveloper do not match those in Oracle Portal. The details of the message should give further information about the cause:

- 404 Not Found: The Oracle Portal instance could not be contacted or has timed out. Try increasing the time out period.
- 406 Not Acceptable: The provider with the given service ID could not be found in this Oracle Portal instance or HMAC authentication has either failed or been incorrectly configured.
- 503 Service Unavailable: the PORTAL user could not be found in Oracle Internet Directory. The Oracle Portal administrator may not have added the PORTAL user to Oracle Internet Directory.

F.2.9.2 During Runtime

Could not Contact Producer

Either the portal registration URL is not correct, the service ID given in Oracle JDeveloper does not match a provider in the Oracle Portal instance, the Oracle Portal could not be contacted (for example, if it is behind a firewall or not available), or the users in Oracle JDeveloper do not match those in Oracle Portal. The details of the message should give further information about the cause:

- 404 Not Found: The Oracle Portal instance could not be contacted or has timed out. Try increasing the time out period.
- 406 Not Acceptable: The provider with the given service ID could not be found in this Oracle Portal instance or HMAC authentication has either failed or been incorrectly configured.

- 503 Service Unavailable: the PORTAL user could not be found in Oracle Internet Directory. The Oracle Portal administrator may not have added the PORTAL user to Oracle Internet Directory.

Portlet does not render properly

- If images are missing or appear as links, see the instructions for downloading the images zip file provided in Section F.2.8, "How to Use the Federated Portal Adapter to Reuse Database Portlets."
- If navigation does not work properly, try editing the page and setting `RenderPortletInIframe` to `true.` Some portlets work better in their own iframe.
- If popups, lists of values, and so on do not work in the portlet, it may be because JavaScript exceptions are raised in an attempt to avoid XSS attacks. The popup must have the same server URL as the custom WebCenter application.
- If a portlet renders a blank screen for Customize or Personalize, it may be that the user does not have permission to edit the portlet. Check the Portal log files.

F.2.10 What You May Need to Know About Limitations of the Federated Portal Adapter

The following limitations apply when using the Federated Portal Adapter with Oracle WebCenter:

- Deep linking may not work. In some cases, putting the portlet in an iframe helps (setting `RenderPortletInIframe` to `true`).
- Links rendered by the portlet must be absolute URLs.
- If a page portlet displays a page with tabs it should be placed into an iframe.
- Refresh, Collapse, and Remove portlet links do not work in portlets displayed in page portlets.
- Not all objects are returned by the Federated Portal Adapter, and thus are not visible in WebCenter. For a full list, see "Oracle Portal - Limitations" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

F.2.11 How to Reuse Oracle PDK-Java Producers from Earlier Oracle Application Server Versions

If you have a producer that you designed and deployed on Oracle Application Server Release 2 (10.1.2) or earlier, then you can reuse it in either one of two ways:

- You can consume a portlet from a producer running on an Oracle Application Server Release 2 (10.1.2.0.2) middle tier. In this case, the producer's code is running on Release 2 (10.1.2.0.2) while the application consuming the producer is running on Release 11.
- You can redeploy the producer application as an EAR file on Oracle WebLogic Server and consume its portlets in other custom WebCenter applications. In this case, both the producer's code and the consuming application run on Release 11.

In either of these cases, you must take some additional steps to get the producer's portlets to operate correctly. Portlets built for Oracle Portal expect certain boilerplate images to be present on the page assembly servlet and they break if those images are not available. These images were included in the middle-tier servlet by default in Oracle Application Server Release 2 (10.1.2.0.2), but for Oracle WebCenter Framework you must manually ensure that the images can be found by portlets.

F.2.11.1 Consuming a Portlet from Oracle Portal

In Oracle WebCenter, to consume a portlet running on a page in an Oracle Portal instance, you can do either of the following:

- Obtain the Oracle Portal images zip file from the Oracle Technology Network and unzip it inside the ADFP servlet.
- Properly configure the resource servlet as follows:

1. Add an initialization parameter to the Web servlet as shown is Example F–1.

Example F–1 Web Servlet Initialization Parameters

```
<servlet>
 <servlet-name>SOAPServlet</servlet-name>
 <display-name>SOAPServlet</display-name>
 <description>Extended Portal SOAP Server</description>
 <servlet-class>oracle.webdb.provider.v2.adapter.SOAPServlet</servlet-class>
 ...
 <init-param>
   <param-name>resourceServletMapping</param-name>
   <param-value>/pdkresource</param-value>
 </init-param>
</servlet>
```

2. Add a servlet definition for the resource servlet as shown in Example F–2.

Example F–2 Servlet Definition for Resource Servlet

```
<servlet>
 <servlet-name>ResourceServlet</servlet-name>
  <display-name>ResourceServlet</display-name>
  <description>Image resource servlet</description>
  <servlet-class>oracle.webdb.provider.v2.adapter.ResourceServlet</servlet-class>
</servlet>
```

3. Add a servlet mapping for the resource servlet as shown in Example F–3.

Example F–3 Servlet Mapping for Resource Servlet

```
<servlet-mapping>
 <servlet-name>ResourceServlet</servlet-name>
 <url-pattern>/pdkresource/*</url-pattern>
</servlet-mapping>
```

4. Now you can register this PDK-Java producer as you would any other PDK-Java producer. For more information about registering PDK-Java producers, see Section 9.2.3, "How to Register an Oracle PDK-Java Portlet Producer."

F.2.11.2 Redeploying PDK-Java Producers from Oracle Portal

If you choose to take a PDK-Java producer built for Oracle Portal and redeployed on Oracle WebLogic Server, then you must enable the resource servlet as follows:

1. Add an initialization parameter to the Web servlet as shown in Example F–4.

Example F–4 Web Servlet Initialization Parameter

```
<servlet>
 <servlet-name>SOAPServlet</servlet-name>
 <display-name>SOAPServlet</display-name>
 <description>Extended Portal SOAP Server</description>
 <servlet-class>oracle.webdb.provider.v2.adapter.SOAPServlet</servlet-class>
 ...
 <init-param>
   <param-name>resourceServletMapping</param-name>
   <param-value>/pdkresource</param-value>
 </init-param>
</servlet>
```

2. Add a servlet definition for the resource servlet as shown in Example F–5.

Example F–5 Servlet Definition for Resource Servlet

```
<servlet>
 <servlet-name>ResourceServlet</servlet-name>
  <display-name>ResourceServlet</display-name>
  <description>Image resource servlet</description>
  <servlet-class>oracle.webdb.provider.v2.adapter.ResourceServlet</servlet-class>
</servlet>
```

3. Add a servlet mapping for the resource servlet as Example F–6.

Example F–6 Servlet Mapping for Resource Servlet

```
<servlet-mapping>
 <servlet-name>ResourceServlet</servlet-name>
 <url-pattern>/pdkresource/*</url-pattern>
</servlet-mapping>
```

4. Now you can register this PDK-Java producer as you would any other PDK-Java producer. For more information about registering PDK-Java producers, see Section 9.2.3, "How to Register an Oracle PDK-Java Portlet Producer."

F.3 Reusing Items

Items from Oracle Portal have no equivalent in Oracle WebCenter Framework. As such, creating, maintaining, and publishing items is not supported in Oracle WebCenter Framework. To replicate the behavior of items with Oracle WebCenter Framework, you can use JCR data controls to include content from Oracle Portal. Oracle Portal is a content repository that Oracle WebCenter Framework supports out of the box in the Data Control Wizard in Oracle JDeveloper. For more information about integrating content from Oracle Portal, see Chapter 8, "Integrating Content."

Glossary

About mode

A **portlet mode** that typically displays information such as copyright, version, and author of the portlet.

ADF

Application Development Framework. A range of technologies aimed at making **Java EE** application development faster and simpler for developers while at the same time taking advantage of proven software patterns to ensure that the developed application is scalable, performant, and the like.

administrator

In WebCenter Spaces there are two types of administrator:

- Fusion Middleware administrator: Also referred to as systems administrator. A user with complete administrative capabilities. This administrator can perform the complete range of security-sensitive administrative duties, and all installation, configuration, and audit tasks.
- WebCenter Spaces administrator: A WebCenter Spaces user who is responsible for customizing WebCenter Spaces out of the box, managing and granting application roles, and maintaining the application when it is in use.

Ajax

A combination of asynchronous JavaScript, dynamic HTML (DHTML), XML, and XmlHttpRequest communication channel that allows requests to be made to the server without fully re-rendering the page. Ajax allows rich client-like applications to use standard internet technologies.

Announcements service

A WebCenter Web 2.0 service that offers a quick, convenient way to create and widely distribute messages instantly or at a specific time.

API

Application Programming Interface. A set of exposed data structures and functions that an application can use to invoke services on an application object, such as a **portlet**.

Application Development Framework

See **ADF**.

application lifecycle

The process of creating and testing an application in a design time environment, deploying it to a production system, and then performing routine maintenance, such as monitoring performance and migrating customization data. The lifecycle of an application also includes performing further enhancements, restaging, and then redeploying the application to the production system.

Application Programming Interface

See **API**.

application role

Roles that are specific to a particular application and are stored in an application-specific stripe of the policy store.

application skin

Specifies the WebCenter Spaces application background color, screen fonts, and, with some skins, the shapes and images used for application buttons and icons. The WebCenter Spaces administrator chooses the default application skin. WebCenter users may change the application skin on the General tab of the Preferences dialog box.

Applications pane

An area of the WebCenter Spaces Sidebar that provides convenient access to your frequently used applications.

authenticated user

A user who is logged into a **WebCenter application**. By default, an authenticated user can access public and secured information, such as pages and **portlet**s.

Contrast with **public user**s, who are not logged in, and can access public content only.

authentication

Identification of a user through an identity management system. You can require ADF authentication to enforce credentials for users to access the WebCenter application only (all ADF resources in the application remain accessible), or authentication *and* authorization to enforce credentials for users to access the WebCenter application and any ADF resources that have been secured in the application.

authorization

The policies that define the access rights of an individual or group to a secured resource. This resource may be a page or component within a page.

authorized user

An individual who has access to a secured resource. For non-public resources, this individual is also an **authenticated user**.

blog page

A page that provides a personal record of an individual user's experience and opinions. There are two kinds of blog: personal blogs are written by an individual; group blogs are written by several users.

Box layout component

An Oracle Composer layout component. A container that enables the placement of content on a WebCenter Spaces page. In Oracle Composer, a Box is rendered as a rectangle comprised of dashed lines. For designers of custom WebCenter applications, this is the runtime equivalent of a Panel Customizable component.

BPEL

Business Process Execution Language. An XML-based markup language for composing a set of discrete Web services into an end-to-end process flow.

business role page

A page, created by the WebCenter Spaces administrator, specifically provided for a given role in an organization. Business role pages provide a targeted environment for users of a particular role, by delivering information that is timely and relevant to individual roles without the noise of irrelevant information from other lines of business. Business role pages appear in the personal spaces of users classified under the specified role.

caching

The act of storing frequently accessed information, typically Web pages, in a location where it can be accessed quickly to avoid frequent content generation.

See also **expiry-based caching** and **validation-based caching**.

Change Mode Button component

A component provided in the Oracle Composer tag library that lets users change from the View mode of a page to Edit mode, in which they can edit the page using Oracle Composer.

Change Mode Link component

A component provided in the Oracle Composer tag library that lets users change from the View mode of a page to Edit mode, in which they can edit the page using Oracle Composer.

check out/check in

A mechanism that enables a user to lock information, by checking it out, so that other users cannot modify that same piece of information. This prevents users from overwriting each other's changes. After making modifications, the user releases it by checking it back in, making it available again for other users to modify.

chrome

Visual elements surrounding a portlet or task flow that provide an access point for actions, such as those on the Actions menu and those embedded in the chrome itself, such as the minimize icon or resize handles.

Community of Interest group space

A group space created using the Community of Interest template. This type of group space provides an optimal structure for supporting communities of people, joining together to learn more about a subject area through the sharing of expertise, ideas, and content.

component

An individual piece of an application, for example, a task flow, portlet, page, or layout element such as a box or image.

Component Catalog

A dialog, accessed from Oracle Composer, that provides access to all the content you can add to a WebCenter application page.

component developer

The developer who builds components (such as portlets, **JavaServer Faces** components, and Web services).

Component Properties

A dialog, accessed from Oracle Composer, that provides access to a component's parameters, display options, style settings, and associated events.

container

An application program or subsystem in which the program building block, known as a component, is run.

content integration services

Services provided by **Oracle WebCenter** to enable developers to display content from a **content repository**, such as by creating **data control**s.

content repository

A specialized storage and management mechanism, such as author-based versioning, full textual searching, content categorization and attribution, and is optimized for storing unstructured information, which differentiates it from a data repository.

content repository data control

A **data control** sourced though a content repository. In a **WebCenter application**, you can create content repository data controls for the following content repositories: **Oracle Portal**, **Oracle Universal Content Management**, and third-party repositories supporting the Java Content Repository (JCR) standard, or your local file system.

credential provisioning page

A **JSF** (`*.jspx`) page used for authenticating to an **external application**. At run time, the Credential Provisioning page displays login data fields consisting of the data fields specified through external application registration. Login information is passed to the producer, which in turn passes the login values to the external application. The application provides the producer with the requested portlets.

After authentication, the user's login credentials are preserved in a **credential store**, which subsequently supplies that information at future sessions. Unless his information changes, the user supplies his credentials only one time.

credential store

A storage area that preserves the login credentials a user provides for authentication to an **external application**.

CSS

Cascading Style Sheet. A simple mechanism for ensuring a consistent look and feel or adding style, such as fonts, colors, and spacing, to Web documents.

custom action

Icons or menu items that are displayed on the header or in the Actions item of a Show Detail Frame component that surrounds a task flow. These actions can control actions defined in the task flow itself, enabling task flows to represent internal actions as options on the chrome.

custom attribute

Specifies group space information in addition to that provided by the built-in attributes. Custom attributes can be used to determine the content of the components in a group space based on the parameter passed in. For example, a component can display data for a specific customer by passing in the customer ID. A custom attribute is simply a name value pair ; for example customerId=400, orderId=11, userName=Smith, and so on. Custom attributes are stored within the group space template.

custom page

Any page created by a user rather than one provided out of the box.

custom resource catalog

A resource catalog that has been customized to control the components that are visible to specific users.

Contrast with **default resource catalog**.

custom role

A user role, created by an administrator or a group space moderator, to meet a specific personal space or group space requirement.

Customize mode

A **portlet mode** that enables users to set the default values for portlet preferences for all users.

customizable component

A WebCenter component that can be added to a page at runtime to enable end users to perform personalizations such as move, minimize, restore, or remove on content within those components. Customizable components are the **Panel Customizable component** and the **Show Detail Frame component**.

customization

An update that affects all users.

data control

A mechanism that provides an abstraction of the business service's data model. The ADF data controls provide a consistent mechanism for clients and Web application controllers to access data objects, collections, methods, and operations.

See also **content repository data control**.

default language

A display language that is used when users log in to WebCenter Spaces. The default language can be overridden temporarily by the session language. The WebCenter Spaces administrator sets the default language on the General tab of the Administration page. Individual users can set their own default language on the General tab of the Preferences dialog box.

default resource catalog

The resource catalog that is provided by default for an application. It contains all of the Oracle ADF components and portlets available to the application.

Contrast with **custom resource catalog**.

Default Server

See **Integrated WLS**.

deployment profile

A file used in application deployment that specifies the following types of information:

- The source files, deployment descriptors, and other auxiliary files that are packages
- The type and name of the archive file to be created
- Dependency information
- Platform-specific instructions
- Other information

Oracle WebCenter Services provides a special deployment profile, the **WebCenter application** WAR deployment profile, that includes an option to export project metadata.

Design view (JDeveloper)

A view, in **Oracle JDeveloper**, that provides a WYSIWYG representation of a file.

See also **Source view (JDeveloper)**.

Design view (WebCenter Spaces)

A view, in Oracle Composer, that provides a WYSIWYG representation of a page and its components.

See also **Source view (WebCenter Spaces)**.

discoverable group space

A group space that can be found by anyone logged into WebCenter Spaces, for example through a search. A group space is made discoverable when the group space moderator enables the Discoverable setting. Discoverable group spaces are listed in My Group Spaces; users wishing to join the group space can request membership through self-subscription (if enabled) or by contacting the group space moderator.

Discussions service

A WebCenter Web 2.0 service that provides a means of creating and participating in text-based discussions with members of a particular group space.

display language

Controls the language in which application user interface elements, such as buttons, field labels, and screen text, are rendered in the browser.

Document List Viewer task flow

A Documents service task flow that exposes a list of documents and optionally folders defined by the listing of a specific folder or the results of a document search. Include on a page by selecting All Documents, Group Space Documents, or Personal Documents from the Oracle Composer catalog.

Documents task flow

A Documents service task flow that exposes all the folders and files available from the default content repository connection and default folder. Include on a page by selecting Documents from the Oracle Composer catalog. Use to create, upload, and manage library content; to manage file versions; and to check files out and in. Equivalent to Document Library task flow in WebCenter Services Catalog in Oracle WebCenter Framework.

Documents page

A predefined page provided in every WebCenter Spaces group and personal space that includes the **Documents task flow** for managing content.

Documents service

A WebCenter Web 2.0 service that provides features for accessing, adding, and managing files; creating and managing file folders; configuring file and folder properties; and searching file and folder content.

domain

Any tree or subtree within the Domain Name System (DNS) namespace. Domain most commonly refers to a group of computers whose host names share a common suffix, the domain name.

dynamically-generated page

A page that displays as the result of a user action, such as a search or a click on a tag. As the name suggests, dynamically-generated pages are not stored, but rather are created as and when needed.

EAR

Enterprise Archive file. A **Java EE** archive file that is used in deploying applications on a **Java EE** application server. **WebCenter application**s are deployed using both a generic EAR file containing the application and the respective run-time customization and a targeted EAR file containing only the application for deployment to the application server. EAR files simplify application deployment by reducing the possibility of errors when moving an application from development to test, and test to production.

See also **JAR** and **WAR**.

ECMA-262 specification

A standardization of scripting programming languages, such as **ECMAScript** and **JavaScript**.

ECMAScript

A scripting programming language, standardized by Ecma International according to the **ECMA-262 specification**. Frequently referred to as **JavaScript** or JScript, which are both extensions of the ECMA-262 specification.

Edit Defaults mode

(**JSR 168** portlets only.) A **portlet mode** that enables personalization of a JSR 168 portlet. Edit Defaults mode is a display mode for the JSR 168 portlet's properties. In a **WebCenter application**, the Edit Defaults mode displays on the portlet's Actions menu as the Customize command.

See also **Edit mode**.

Edit mode

A **portlet mode** that enables personalization of the portlet for each user, for each instance.

See also **Edit Defaults mode**.

edit mode

A view mode that enables users to modify the content, style, and layout of a page. Access edit mode by choosing Edit Page from the Page Actions menu.

EL

Expression Language. Provides a short-hand way of working with Web application data by providing operators for retrieving and manipulating application data residing in a **Java EE** Web container. In a **WebCenter application**, EL expressions are encapsulated in the characters "#{" and "}" and typically come in the form #{object.data} where *object* represents any Java object or **ADF** component placed in the **Java EE** Web container's page, request, session, or application's scope.

Enterprise Archive file

See **EAR**.

enterprise mashup

An application that enables users to bring all sorts of content and services together in a single place.

Events service

A WebCenter Web 2.0 service that provides group calendars, which you can use to schedule meetings, appointments, and so on. This service is available only in WebCenter Spaces, and not in custom WebCenter applications.

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. When there is a request for the item beyond the retention period, it is refreshed in the cache.

See also **validation-based caching**.

Expression Language

See **EL**.

Extensible Markup Language

See **XML**.

external application

Applications that do not delegate authentication to the single sign-on server. Instead, they display HTML login forms that ask for application user names and passwords. At the first login, users can choose to have the single sign-on server retrieve these credentials for them. Thereafter, they are logged in to these applications transparently.

farm

A collection of components managed by Fusion Middleware Control. A farm can contain a Managed Server domain and other Oracle Fusion Middleware system components that are installed, configured, and running on the domain.

favorites

A personal list of links to favorite WebCenter Spaces pages and external Web sites.

Federated Portal Adapter

See **FPA**.

FOD

Fusion Order Demo. An enterprise application built using Oracle Fusion Middleware, including Oracle WebCenter, used to provide examples of WebCenter functionality.

FPA

Federated Portal Adapter. A component of **Oracle Portal** that enables Oracle Portal instances to share their database portlets through the Web portlet interface. Using the FPA, Oracle Portal database portlets, including PL/SQL portlets, Portlet Builder portlets, and page portlets can be made available for use in WebCenter applications.

Full Screen Mode (WebCenter Spaces)

A view mode that opens the group space to occupy the entire screen, thus maximizing the display space. The Sidebar is not displayed in Full Screen Mode.

Full Screen mode (Portlets)

(**PDK-Java** portlets only.) A **portlet mode** that provides more content than can be shown in the portlet when it is sharing a page with other portlets.

Fusion Middleware Control

A browser-based management application that is deployed when you install Oracle WebCenter. From Fusion Middleware Control Console, you can monitor and administer a **farm** (such as Oracle WebCenter).

Fusion Order Demo (FOD)

See **FOD**.

Group Project group space

A group space created using the Group Project template. This type of group space provides an optimal structure for supporting a core project team where each member might come from a different department but all members contribute toward meeting a common goal.

group space

A work area within WebCenter Spaces that supports a group of people of any size that is organized around an area of interest or a common goal.

group space icon

An image displayed alongside group space names on the Group Spaces page in My Group Spaces to help other users with identification and location.

group space logo

An image displayed on the group space Home page to provide a visual identity for the group space. Group space logos also display alongside the group space name at the top of the page in Full Screen Mode.

group space member

A user who is participating in a group space. Members can be added or invited to a group space, or they can subscribe to a group space themselves if self-registration is enabled.

group space owner

A user who initially created a group space. The group space owner is automatically also a moderator of the group space.

group space template

A starting point for group space creation. WebCenter Spaces includes three templates to get you started: Group Project, Community of Interest, and Blank, but you can turn any group space into a template to use it as the starting point for other similar group spaces.

Group Space Unavailable page

A predefined page that displays when a group space member tries to open a group space that is temporarily offline. Moderators can customize this page.

HA

High Availability. A collection of solutions to ensure that your applications meet the required availability to achieve your business goals, eliminating single points of failure with no or minimal outage in service.

Help mode

A **portlet mode** that displays usage information about the functionality of the portlet.

High Availability

See **HA**.

HTML

Hypertext Markup Language. A format for encoding hypertext documents that may contain text, graphics, and references to programs and other hypertext documents.

HTML Markup layout component.

An Oracle Composer layout component. A simple HTML component that renders raw HTML and JavaScript mark-up inline on the page.

HTTP

Hypertext 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.

Hyperlink layout component

An Oracle Composer layout component. A link to an internal or external Web page. For designers of custom WebCenter applications, this is the runtime equivalent of a Go Link component.

Hypertext Markup Language

See **HTML**.

Hypertext Transfer Protocol

See **HTTP**.

IDE

Integrated Development Environment. A visual application development tool containing editors, debuggers, screen painters, object browsers, and the like. **Oracle JDeveloper** is an example of an IDE.

Image layout component

An Oracle Composer layout component. An illustration that can include a hyperlink. For designers of custom WebCenter applications, this is the runtime equivalent of an Image Link component.

IMP service

See **Instant Messaging and Presence service**.

initialization parameters

The parameters initialized upon the start-up of a standard JSR 168 portlet. Initialization parameters provide an alternative to JNDI (Java Naming and Directory Interface) variables. Use initialization parameters instead of JNDI to configure the behavior of all of the different components of the portlet—for example, servlets and other portlets—in a compatible way. In **Oracle WebCenter**, initialization parameters are entered into the `portlet.xml` file.

Instant Messaging and Presence service

A WebCenter Web 2.0 service that enables users to observe the presence status of other authenticated users and provides instant access to interaction options, such as instant messages, emails, and phone calls.

Integrated Development Environment

See **IDE**.

Integrated WLS

Integrated WebLogic Server. A WLS instance used as a platform for pretesting WebCenter application deployments on a local computer. Integrated WLS also contains preconfigured portlet producers and several useful prebuilt portlets.

JAAS

Java Authentication and Authorization Service (JAAS) is a Java package that enables applications to authenticate and enforce access controls upon users. JAAS is designed to complement Java 2 security and implements a Java version of the standard Pluggable Authentication Module (PAM) framework. This enables an application to remain independent from the authentication service, and supports the use of custom authentication modules.

JAAS extends the access control architecture of the Java 2 Security Model to support subject-based authorization. It also supports declarative security settings, in deployment descriptors, instead of being limited to code-based security settings.

JAR

A Java archive file. JAR files contain the class, image, and sound files for a Java application or applet. JAR files may also be compressed.

See also **EAR** and **WAR**.

Java Authentication and Authorization Service

See **JAAS**.

Java Content Repository

See **JCR 1.0**.

Java EE

Also known as Java EE 5. Java Enterprise Edition 5 Platform. A platform that enables application developers to develop, deploy, and manage multitier, server-centric, enterprise-level applications. The Java EE 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 Java EE portlets and expose them through Web producers.

Java Enterprise Edition 5 Platform

See **Java EE**.

Java Portlet Specification

Standardizes how components for portal servers are to be developed. This specification defines a common portlet **API** and infrastructure that provides facilities for personalization, presentation, and security. Portlets using this **API** and adhering to the specification are product-agnostic, and can be deployed to any portal product that conforms to the specification. See also **JSR 168**.

Java Specification Request

See **JSR 168**.

JavaScript

A scripting language developed by Netscape that enables generation of **portlet**s that introduce dynamic behavior in otherwise static HTML. This language is compliant with the European Computer Manufacturing Association's **ECMA-262 specification** (ECMA-262 standard). An alternative name for this EMCA-262 language is **ECMAScript**.

JavaServer Faces

See **JSF**.

JavaServer Page

See **JSP**.

JCR 1.0

Java Content Repository 1.0. Also known as JSR 170. It proposes a standard access and interaction **API** for content repositories, much like JDBC does for databases.

JDeveloper

See **Oracle JDeveloper**.

JSF

JavaServer Faces. A standard Java framework for building Web applications. It simplifies development by providing a component-centric approach to developing Java Web user interfaces. JSF offers rich and robust **API**s that provide programming flexibility and ensures that applications are well designed with greater maintainability by integrating the Model-View-Controller (**MVC**) design pattern into its architecture. As JSF is a Java standard developed through Java Community Process, development tools like **Oracle JDeveloper** are fully empowered to provide easy to use, visual, and productive development environments for JSF.

JSF JSP

JavaServer Faces JavaServer Page. JSF JSPs differ from plain JSPs through their support of **Oracle ADF Faces** components for the user interface and JSF technology for page navigation. JSF JSP pages leverage the advantages of the Oracle **Application Development Framework** (Oracle ADF) by using the ADF Model binding capabilities for the components in the pages.

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**.

JSP tags

Tags that can be embedded in **JSP**s 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.

JSR 168

Java Specification Request (JSR) 168. Defines a set of **API**s 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 170

See **JCR 1.0**

JSR 301

See **Oracle JSF Portlet Bridge**.

keystore

A file that provides information about available public and private keys that are used for authentication and data integrity. User certificates and the trust points needed to validate the certificates of peers are also stored securely in the keystore

layout box

A container that enables placement of content on a WebCenter Spaces page.

layout component

An object for enhancing the usefulness and appearance of a given page. Layout components include layout boxes, a rich text editor, images, hyperlinks, and so on.

Layout Customizable component

A component provided in the Oracle Composer tag library that enables users to select from a set of predefined layouts (for example, two column, three column, two row, and so on) and apply it to the page. Users can apply these layouts to a particular area of the page or to the entire page.

LDAP

Lightweight Directory Access Protocol. A standard, extensible directory access protocol. It is a common language that LDAP clients and servers use to communicate.

The framework of design conventions supporting industry-standard directory products, such as the Oracle Internet Directory.

lifecycle

See **application lifecycle**.

Lightweight Directory Access Protocol (LDAP)

See **LDAP**.

Links service

A WebCenter Web 2.0 service that provides a means of creating a bidirectional association between two objects, thus setting up easy access between those objects.

List Manager

A task flow of the Lists service that provides access to all the tools for creating and revising lists and list content and to all of a group space's current lists.

Lists page

A predefined page that displays the group space's current lists.

Lists service

A WebCenter Web 2.0 service for creating, publishing, and managing lists. Uses for lists include tracking issues, capturing project milestones, publishing project assignments, and so on. This service is available only in WebCenter Spaces, and not in custom WebCenter applications.

Lists Viewer

A task flow of the Lists service that provides a means of placing a particular list on a group space page.

Mail service

A WebCenter Web 2.0 service for exposing familiar mail functionality in WebCenter applications.

Managed Server

In a production environment, a Managed Server hosts applications and the resources needed by those applications. A domain, which is a logically related group of Oracle WebLogic Server resources, can have any number of Managed Servers. An Administration Server manages these servers.

mashup

A Web application that enables end users to pull information from different sources to create a personalized application that exactly meets their individual requirements.

MDS

Oracle Metadata Services. A core technology of the **Application Development Framework**. MDS provides a unified architecture for defining and using metadata in an extensible and customizable manner.

Model-View-Controller

See **MVC**.

moderator

A WebCenter Spaces user who is responsible for managing a particular group space. A group space moderator can add and remove members, invite new members, enable self registration, provide and update group space metadata, and manage the services available to the group space.

Movable Box layout component

An Oracle Composer layout component. A container that enables the placement of content on a WebCenter Spaces page and also enables the container (rather than just the content) to be moved around on the page. For designers of custom WebCenter applications, this is the run time equivalent of Show Detail Frame component.

MVC

Model-View-Controller. A classic design pattern often used by applications that need the ability to maintain multiple views of the same data. The MVC pattern hinges on a clean separation of objects into one of three categories: models for maintaining data, views for displaying all or a portion of the data, and controllers for handling events that affect the model or views. Because of this separation, multiple views and controllers can interface with the same model. Even new types of views and controllers that never existed before, such as portlets, can interface with a model without forcing a change in the model design.

My Group Spaces page

A predefined page that displays a list of all the group spaces and group space templates available to the currently logged in user. This includes group spaces of which the user is a member, group spaces marked as discoverable, and group spaces that are public and available to everyone.

navigation parameter

Parameters in a **WSRP** container that map to the render parameters with the same name in **JSR 168** portlet code. Navigation parameters are exposed by the portlet to the consumer. The consumer stores and manages parameter values and sends them on every invocation to the portlet. Navigation parameters are a WSRP version 2 feature.

Notes service

A WebCenter Web 2.0 service that provides useful features for writing personal notes and reminders. This service is available only in WebCenter Spaces, and not in custom WebCenter applications.

OAM

See **Oracle Access Manager (OAM)**.

OHS

See **Oracle HTTP Server (OHS)**.

OmniPortlet

A component of **Oracle WebCenter** that enables you to inject portal-like capabilities, such as portlets, content integration, and customization, into your **Oracle ADF Faces** applications.

Oracle Access Manager (OAM)

Part of Oracle's enterprise class suite of products for identity management and security, Oracle Access Manager provides a wide range of identity administration and security functions, including several single sign-on options for WebCenter Spaces and WebCenter custom applications. OAM is the recommended single sign-on solution for Oracle WebCenter 11g installations.

Oracle ADF Faces

Oracle **ADF** Faces is a rich set of user interface components based on the new **JavaServer Faces** JSR (JSR 127). Oracle ADF Faces provide various user interface components with built-in functionality, such as data tables, hierarchical tables, and color and date pickers, that can be customized and reused in an application.

Oracle Composer

A seamlessly integrated environment for populating, revising, and configuring WebCenter application pages. It enables users to easily build or revise page layout and content. It also provides the means of adding different components, such as task flows, portlets, content, and other objects, onto a page and then linking those components for a more relevant or personalized view of the information.

Oracle Content Server

Software for building secure business libraries with check in and check out, revision control, and automated publishing in web-ready formats. Current information is available to authorized users anytime, anywhere.

Oracle Enterprise Manager

A component that enables administrators to manage Oracle Fusion Middleware services through a single environment. The Fusion Middleware administrator uses Enterprise Manager to configure, manage, and monitor WebCenter applications.

Oracle HTTP Server (OHS)

Software that processes Web transactions that use the Hypertext Transfer Protocol (HTTP). Oracle uses HTTP software developed by the Apache Group.

Oracle Internet Directory

Oracle's LDAP V3 compliant LDAP server. It is used as the default repository provisioning users and groups.The repository for storing **Oracle Portal** user credentials and group memberships. By default, the **Oracle Single Sign-On** 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.

Oracle JDeveloper

Oracle JDeveloper is an integrated development environment (**IDE**) for building applications and Web services using the latest industry standards for Java, XML, and SQL. Developers can use Oracle JDeveloper to create Java portlets.

Oracle JSF Portlet Bridge

Based on and conforming to JSR 301, the Oracle JSF Portlet Bridge enables application developers to expose a JSF application or task flow as a JSR 168 portlet for consumption in another application.

Oracle Metadata Services

See **MDS**.

Oracle Portal

A component used for the development, deployment, administration, and configuration of enterprise class **portal**s. Oracle Portal incorporates a portal building framework with self-service publishing features to enable you to create and manage information accessed within your portal.

Oracle SES

Oracle Secure Enterprise Search (SES) provides an easy-to-use, Internet-search-like user experience for public and secure sources. Based on crawling agents, the search can include structured and unstructured, public and secure content. Oracle Secure Enterprise Search is included with **Oracle WebCenter**.

Oracle Single Sign-On

A component that enables users to log in to all features of the Oracle Fusion Middleware product suite, and to other Web applications, using a single user name and password.

Oracle SOA Suite

A middleware component of Oracle Fusion Middleware. Oracle SOA Suite enables services to be created, managed, and orchestrated into SOA composite applications. Composites enable you to easily assemble multiple technology components into one SOA composite application. Oracle SOA Suite plugs into heterogeneous infrastructures and enables enterprises to incrementally adopt SOA.

Oracle Technology Network

See **OTN**.

Oracle Universal Content Management

A consolidated content management application that provides multisite Web content management, document management, digital asset management and records management.

Oracle WebCenter

A suite of services that enables you to build custom WebCenter applications. Oracle WebCenter reduces the front-end labor historically required to bring necessary business components to the user by capitalizing on the notion of Service Oriented Architecture (SOA). The suite includes a wide range of plug-and-play products, tools, and services that make it easy to build the applications your users need. Oracle WebCenter includes:

- **Oracle WebCenter Services**
- **Oracle WebCenter Framework**
- **content integration services**
- **ADF**
- **Secure Enterprise Search**
- Mobile Services
- Portlet Pack

Oracle WebCenter Framework

A set of features provided by **Oracle WebCenter** that augments the Java Server Faces (JSF) environment by providing additional integration and run-time customization options It is the basis of Oracle WebCenter and supports the creation and execution of context-rich applications, which can come in the form of human interaction, files and documents, or a clear representation of where the user is within a complex work process. It includes such features as:

- Portlet support
- **content integration services**
- **Oracle JSF Portlet Bridge**
- Search framework
- customizable components

Oracle WebCenter Services

A suite of services included in **Oracle WebCenter** that enables you to enhance your **Oracle ADF Faces** applications with WebCenter application capabilities, such as portlets, content integration, and customization. Includes design time extensions to **Oracle JDeveloper** to help to build **WebCenter application**s. The services include:

- **Oracle Universal Content Management**
- **Secure Enterprise Search**
- communication services

Oracle WebLogic Communications Services (OWLCS)

A comprehensive platform designed to integrate communication services with enterprise services and applications. It includes easy to consume services to support interactions with key communication channels.

Oracle WebLogic Server Administration Console

A browser-based, graphical user interface to manage a WebLogic Server domain. Use to:

- Configure, start, and stop WebLogic Server instances
- Configure WebLogic Server clusters
- Configure WebLogic Server services, such as database connectivity (JDBC) and messaging (JMS)
- Configure security parameters, including creating and managing users, groups, and roles
- Configure and deploy your applications
- Monitor server and application performance
- View server and domain log files
- View application deployment descriptors
- Edit selected run-time application deployment descriptor elements

Oracle Wiki and Blog Server

Provides web services that enable interaction between your application and the wiki.

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/`.

OWLCS

See **Oracle WebLogic Communications Services (OWLCS)**.

Page Customizable component

A component provided in the Oracle Composer tag library that defines the editable area of a page at runtime. Within this area, users can edit properties for a component, add content to the page, arrange content, and so on.

page parameter

A parameter that enables your page to take values through its URL. Page parameters are defined using the `<parameter>` tag at the top of your `PageDef.xml`. You can bind page parameters to your **page variable**s.

page parameter

A parameter associated with a page that can be used to store values that can then be passed to the components on the page

Page Properties

A dialog, accessed from Oracle Composer, that provides access to a page's display options, security settings and parameters.

page scheme

Determines the background image used in the page. WebCenter Spaces provides several default page schemes and an option for specifying a custom page scheme.

Page service

A service for creating new pages and task flows in your application at runtime.

page style

Determines the initial page structure, for example one column or two column. Some default page styles also include the task flows, components, and page properties useful for a particular purpose. For example, a page created using the Text page style includes a Text layout component.

page variable

A variable that binds your public portlet parameter to the page. Page variables are defined within the `<variableIterator>` of your `PageDef.xml`. One page variable can be bound to multiple public portlet parameters.

Panel Customizable component

A component provided in the Oracle Composer tag library that provides a container region for a group of Oracle ADF components and portlets that are customizable at runtime. Any Show Detail Frame components and portlets added as child components to a Panel Customizable component can be moved or maximized with the Panel Customizable component.

parameter

A variable that controls the default behavior of task flow content and facilitates the wiring of a task flow to page parameters and page definition variables.

participant

A WebCenter Spaces user who can manipulate the content of a group space. A participant can upload and share documents, initiate and take part in chats with other members, create discussion topics, modify due dates of tasks assigned to them, create new or view existing lists.

PDK-Java

Java Portlet Developer Kit. The development framework used to build and integrate Web content and applications with **Oracle WebCenter**. 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.

personalization

An update that affects only the user who made it.

personal page

A page created by a user in his or her personal space. Personal pages are viewable by other users only if specifically granted access by the user who created the page.

personal profile

A page that displays a user's personal information such as email address, phone number, office location, department, manager, direct reports, and so on.

personal space

A work area within WebCenter Spaces that provides individual users with a private space for storing personal content, keeping notes, viewing and responding to assignments, maintaining a list of online buddies, and performing many other tasks relevant to their unique working day. Users can also extend this environment by creating additional personal pages and custom content.

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.

Portal Developer Kit

See **PDK-Java**.

portlet

A reusable Web component that can draw content from many different sources. Portlets can 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. Because different portlets can be placed on a common page, the user receives a single-source experience, even though the content may be derived from multiple sources. Portlet resources include the many prebuilt portlets available out of the box from many sources, programmatic portlets built through WebCenter's JSR 168 and PDK-Java Portlet wizards, and through other portlet building tools.

portlet mode

The ways by which a **portlet** can be called to display information. These methods include:

- **Shared Screen mode** or **View mode**
- **Edit mode** or **Edit Defaults mode**

- **Customize mode**
- **Help mode**
- **About mode**
- **Full Screen mode (Portlets)** or **Show Details Page mode**

Portlet Producer Application template

An application template, provided by JDeveloper, for creating an application with the recommended projects and technology scopes required for developing portlets. The Portlet Producer Application template consists of a single project scoped for portlet creation (Portlets).

See also **WebCenter Application template**.

predefined page

A page created by WebCenter Spaces to perform a specific function. Examples of predefined pages include, Welcome pages, Search pages and Documents pages.

Predeployment Tool

A utility for **WebCenter application**s that helps you configure your target system with the new producer registrations you have added to your application in Oracle JDeveloper. You must run this utility before deploying your application. You can also use this utility after deployment to migrate metadata from stage to production, such as for exporting and importing your customizations. This tool also enables you to define the **MDS** repository location to allow run-time customizations to be migrated.

pretty URL

A shortened version of a page's URL that hides the complexity of the real Web address.

private parameter

A portlet parameter that is known only to the portlet itself and has no connection to the page on which the portlet resides.

Contrast with **public parameter**.

producer

A producer communication link between portlet consumers (such as a **WebCenter application** or a **portal**). When a consumer application renders a portlet, it calls the producer of that portlet, which in turn executes the portlet and returns the results in the form of portlet content. A producer can contain one or more portlets. A portlet can be contained by only one producer.

Oracle WebCenter supports two types of producers:

- Oracle **PDK-Java** producers: Deployed to a **Java EE** application server, which is often remote and communicates through Simple Object Access Protocol (SOAP) over HTTP.
- **Web Services for Remote Portlets** (WSRP): A Web services standard that enables 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. A portlet (regardless of language) deployed to a WSRP-enabled container can be rendered in any application that supports this standard.

programmatic portlets

Portlets constructed in a non-declarative manner using **APIs**. Also referred to as *hand-* or *manually coded* portlets.

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 computers, while allowing Web access.

public group space

A group space that is available to all users, even those who are not logged in to WebCenter Spaces.

public page

A page within WebCenter Spaces that is available to all users, even those who are not logged in to WebCenter Spaces.

public parameter

A portlet parameter that is known to the page and bound to it by way of a page variable.

Contrast with **private parameter**.

public user

A user who can access, but is not logged into, a **WebCenter application**. A public user can view any page that has been marked as public, but cannot personalize or edit any content, or view pages that have any form of access control.

Contrast with **authenticated user**.

Recent Activities service

A WebCenter Web 2.0 service that provides a means of tracking recent activities in a WebCenter application.

Recent Documents task flow

A Documents service task flow that exposes the files most recently modified in some way. Include on a page by selecting Recent Documents from the Oracle Composer catalog.

resize handle

A user interface element in a task flow chrome increasing or decreasing the height of the task flow.

Resource Action Handling framework

Enables services that expose custom resources to be viewed, searched, and tagged.

Resource Catalog

A catalog that provides a federated view of one or more otherwise unrelated repositories in a unified search and browse user interface. Resources are created and published in their source repository and are then exposed to the developer in

JDeveloper's Resource Palette and to the end user in the Resource Catalog Viewer. Resource catalogs can contain layout components, Oracle ADF components, portlets, service task flows, and documents.

Reverse Proxy Server

A server process that hides the physical location of internal servers by exposing the servers as a single public site. Requests to the public site are routed to the appropriate internal server.

Rich Text portlet

A portlet, based on the **WSRP** standard, offering browser-based rich text editing at run time on a deployed Oracle ADF **JavaServer Faces** JSP.

RSS service

A WebCenter Web 2.0 service that provides a means of publishing content from other services as news feeds. The RSS service supports both RSS 2.0 and Atom 1.0 formats.

Search page

A predefined page for running searches, creating and managing saved searches, and viewing and refining search results.

Search service

A WebCenter Web 2.0 service that enables the discovery of information and people in a WebCenter application, returning only the results users are authorized to see

Secure Enterprise Search

See **Oracle SES**.

secured application page

A page created by a user that has not been made available to public users.

Self-Registration page

A predefined page where users can register with WebCenter Spaces, thus creating themselves an LDAP login account. Administrators can customize certain aspects of this page.

Self-Subscription page

A predefined page where users can register to become members of a group space. Moderators can customize certain aspects of this page.

service ID

A PDK-Java producer's unique identifier. PDK-Java enables you to deploy multiple producers under a single adapter servlet. Different producers are identified by their unique service IDs. A service ID is required only when a service ID/producer name is not appended to the URL endpoint.

Service Oriented Architecture

See **SOA**.

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 language

A display language specified by the user that remains in effect for the life of the session cookie (usually from the time the user logs on until he logs off). If the user clears browser cookies, the display language reverts to the default language or, if a default language if not specified, the application display language. Set the session language in the Change Language pop-up, accessible from the Welcome page.

Shared Screen mode

A **portlet mode** that renders the body of the portlet and enables you to display a portlet on a page that can contain other portlets. Every portlet must have at least a Shared Screen mode.

See also **View mode**.

Show Detail Frame component

A component provided in the Oracle Composer tag library that renders a border or chrome around the child component. It provides a header with an Actions menu and thereby providers user interface (UI) controls to customize the display of the child component. However, to customize the display of the child component, the Show Detail Frame component must be included inside a Panel Customizable component.

Show Details Page mode

A **portlet mode** that provides full-browser display of the portlet. For example, a portlet in **Show Page mode** could be limited to displaying only the ten most recently submitted expense reports, while the same portlet in Show Details Page mode could show all submissions.

Contrast with **Show Page mode**.

show modes

Types of **portlet mode**s encompassing **Show Page mode** and **Show Details Page mode**.

Show Page mode

A **portlet mode** that provides a smaller portlet display to allow space for additional portlets and other objects in the browser window. For example, a portlet in Show Page mode could be limited to displaying only the ten most recently submitted expense reports, while the same portlet in Show Details Page mode could show all submissions.

Contrast with **Show Details Page mode**.

Sidebar

A panel in WebCenter Spaces that provides quick access to tools and information essential to personal productivity, including mail, personal contacts, and so on.

skin

A style sheet based on the CSS 3.0 syntax specified in one place for an entire application. Instead of providing a style sheet for each component, or inserting a style sheet on each page, you can create one skin for the entire application.

SOA

Service Oriented Architecture. A design methodology aimed at maximizing the reuse of application services.

Source view (JDeveloper)

A view, in **Oracle JDeveloper**, that provides a way to directly edit the source code of a file.

Source view (WebCenter Spaces)

A view, in Oracle Composer, that provides a selectable structural representation of a page and its components.

See also Design view (WebCenter Spaces).

struts

A development framework for Java servlet applications based upon the **MVC** design paradigm.

style properties

Used to override the style information from the skin CSS to set specific instances of component display.

Tags service

A WebCenter Web 2.0 service that enables users to apply their own terms to application objects, making it possible to search for those objects using personally meaningful terms.

task flow

A set of ADF Controller activities, control flow rules, and managed beans that interact to allow a user to complete a task. Task flows provide a modular approach for defining control flow in an application. Instead of representing an application as a single JSF page flow, developers can break it up into a collection of reusable task flows.

task flow header

An area at the top of a task flow that displays the task flow name and various tools for interacting with the task flow.

template

See group space template.

Text layout component

An Oracle Composer layout component. A rich text editor for providing static page text. For designers of custom WebCenter applications, this is the runtime equivalent of a Rich Text Editor component.

Unauthorized Access page

A predefined page that displays when someone tries to open a page without access permissions. Moderators can customize the default content of this page.

URL

Uniform Resource Locator. A compact string representation of the location for a resource that is available through the Internet.

URL parameter

See **private parameter**.

validation-based caching

A **caching** method that uses a validation check to determine if the cached item is still valid.

Contrast with **expiry-based caching**.

viewer

A WebCenter Spaces user who can look at the information in a group space but cannot contribute any of their own.

View mode

(**JSR 168** portlets only.) A **portlet mode** that enables you to display a JSR 168 portlet on a page that can contain other portlets. It is the only required mode for JSR 168 portlets.

See also **Shared Screen mode**.

WAR

Web application archive file. This file is used in deploying applications on a **Java EE** application server. WAR files encapsulate in a single module all of the components necessary to run an application. WAR files typically contain an application's **servlet**, **JSP**, and **JSF JSP** components.

See also **EAR** and **JAR**.

Web 2.0

Technologies, such as wiki, RSS, and blogs, that enable the construction of highly interactive Web applications.

See also **WebCenter Web 2.0 service**.

Web Application Archive file

See **WAR**.

Web clipping

A feature that 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.

Web Clipping portlet

A browser-based declarative tool that enables you to integrate any Web application with your **WebCenter application**. It is designed to give you quick integration by leveraging the Web application's existing user interface. You can drag and drop Web Clipping portlets on to a `*.jspx` page.

Web Page layout component

An Oracle Composer layout component. A means of embedding another Web site, wiki, or blog within the context of a WebCenter Spaces page. For designers of custom WebCenter applications, this is the equivalent of an Inline Frame component.

Web server

A program that delivers Web pages.

Web Services for Remote Portlets

See **WSRP**.

WebCenter

See **Oracle WebCenter**.

WebCenter application

An ADF application that combines Web content, portlets, and collaborative services for the end user. Administrators can customize the **WebCenter application** based on their roles and skill levels in the organization.

WebCenter application administrator

The administrator responsible for maintaining the **WebCenter application**. This administrator performs tasks such as implementing the branding for the WebCenter application, making new content available, modifying pages, and granting and revoking privileges.

Contrast with Fusion Middleware Administrator who is responsible for setting up and configuring WebCenter Spaces, and performing on-going administrative tasks for WebCenter Spaces and other WebCenter components.

WebCenter application developer

The developer who plans, builds, and maintains a **WebCenter application** using the Oracle Application Development Framework, **Oracle JDeveloper**, and the **Oracle WebCenter**.

WebCenter application end user

The WebCenter application end user is the run time user of the **WebCenter application**, who accesses pages, portlets, and content, and personalizes portlets (assuming the appropriate privileges).

WebCenter Application template

An application template, provided by JDeveloper, for creating an application with the recommended projects and technology scopes required for developing a WebCenter application. The WebCenter Application template consists of a project for the data model (Model) and a project for consuming portlets, components, and data controllers (ViewController).

See also **Portlet Producer Application template**.

WebCenter Extension for Oracle JDeveloper

An extension available through the Oracle JDeveloper Update Wizard that installs the necessary libraries, templates, wizards, and dialogs needed to build and deploy **WebCenter applications** in **Oracle JDeveloper**.

WebCenter Framework

See **Oracle WebCenter Framework**.

WebCenter Services

See **Oracle WebCenter Services**.

WebCenter Spaces

A Web-based application that offers the very latest technology for social networking, communication, collaboration, and personal productivity. WebCenter Spaces uses services and applications to bring everything together that users require to exchange ideas with others, keep track of personal and work-related tasks, interact with critical applications, and zero in on projects and interests; all within a single integrated environment.

WebCenter Spaces application administrator

See administrator.

WebCenter Spaces RSS reader

An RSS reader provided with WebCenter Spaces that incorporates public news feeds from external sources onto application pages. This RSS reader is available only in WebCenter Spaces, and not in custom WebCenter applications.

WebCenter systems administrator

See administrator.

WebCenter Web 2.0 service

A service that provides Web 2.0 functionality in support of personal and team objectives. WebCenter provides the following services:

- Announcements service
- Discussions service
- Documents service
- Events service
- Instant Messaging and Presence service
- Links service
- Lists service
- Mail service
- Notes service
- Recent Activities service
- RSS service
- Search service
- Tags service
- Worklist service

WebLogic Server

See **WLS**.

Welcome page

There are two types of Welcome page:

- Public Welcome page: A predefined page that users encounter before logging in to WebCenter Spaces.
- Personal Welcome page: A predefined page that introduces users to their personal space.

wiki page

A page that provides in-place editing using HTML or a simple mark-up language. Any user with sufficient privileges can add, revise, and remove information.

WLS

WebLogic Server. A scalable, enterprise-ready Java Platform, Enterprise Edition (Java EE) application server. The WebLogic Server infrastructure supports the deployment of many types of distributed applications and is an ideal foundation for building applications based on Service Oriented Architectures (SOA).

See also **Integrated WLS**

WLST

WebLogic Scripting Tool. A command line tool for managing Oracle Fusion Middleware components, such as Oracle WebCenter.

Worklist service

A WebCenter Web 2.0 service that provides access to notifications, alerts, and BPEL tasks assigned to the current user.

WSRP

Web Services for Remote Portlets (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 based on a particular language (such as JSR 168, .NET, Perl) and any WSRP portal. A portlet (regardless of language) deployed to a WSRP-enabled container can be rendered on any portal that supports this standard.

XML

Extensible Markup Language (XML) is an open standard for describing data using a subset of the SGML syntax.

XSL

Extensible Stylesheet Language (XSL) is the language used within style sheets to transform or render **XML** documents.

Index