ORACLE®

Oracle® Fusion Middleware

Developing WebCenter Portal Assets and Custom Components
with Oracle JDeveloper

12*c* (12.2.1.1)

E71587-01

June 2016

Documentation for developers that explains how to develop
and customize portal assets and components for Oracle
WebCenter Portal using Oracle JDeveloper.

ORACLE®

Oracle Fusion Middleware Developing WebCenter Portal Assets and Custom Components with Oracle JDeveloper, 12c (12.2.1.1)

E71587-01

Contents

Part III   Working with Portlets

11   Introduction to Portlets

12   Creating Portlets from JSF Applications Using the Oracle JSF Portlet Bridge

Part IV   Working with Pagelets

14   Introduction to Pagelets

15   Creating Pagelet Producer Objects

16   Working with Pagelets and Gadgets

Part V   Additional WebCenter Portal Customizations

20   Developing Shared Libraries

21   Localizing Portals

B   WebCenter Portal Accessibility Features

C   Using the WebCenter Portal REST APIs

Preface

This guide explains how to build WebCenter Portal assets, extensions to WebCenter Portal Server, and shared libraries using JDeveloper. This guide provides in-depth information for all of the following tasks:

- How to set up and prepare your development environment

- How to build and deploy WebCenter Portal assets

- How to build and deploy extensions to WebCenter Portal

- How to build and deploy shared libraries that can be used in WebCenter Portal

> **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 is written for a developer who provides support for WebCenter Portal applications. The developer is primarily responsible for developing assets (such as page templates, resource catalogs, skins, portlets, and task flows), which are published and leveraged by knowledge workers and application specialists. A developer primarily works with JDeveloper and creates WebCenter Portal Asset application and leverages one of three asset-related templates: the WebCenter Portal Asset template, the WebCenter Portlet Producer Application template, or the WebCenter Portal Server Extension template.

The chapters in Working with WebCenter Portal Assets explain how to develop assets such as page templates, resource catalogs, skins, portlets, and task flows. Several other parts of this guide also contain chapters devoted to working with assets such as portlets, pagelets, and shared libraries.

Developer Tasks

Tasks that are typical of a developer include:

- Developing custom assets, like page templates and for portals in WebCenter Portal

- Developing Java portlets

- Modifying existing WebCenter Portal assets

- Developing and deploying task flows, managed beans, and other custom components

- Maintaining the source control system

- Maintaining a build system

This guide also assumes that the audience has already read the guide Developing Fusion Web Applications with Oracle Application Development Framework and is familiar with the following technologies:

- 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

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle Fusion Middleware 12*c* (12.2.1.1) documentation set or on Oracle Technology Network (OTN) at `http://www.oracle.com/technology/index.html`.

- *Administering Oracle WebCenter Portal*

- *Building Portals with Oracle WebCenter Portal*

- *Using Oracle WebCenter Portal*

- *Error Messages*

- *Developing Fusion Web Applications with Oracle Application Development Framework*

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.

Who's Who

The WebCenter Portal documentation is organized so that the tasks in a particular guide address a specific user *persona*. Each persona is associated with a set of skills required to work with WebCenter Portal, from basic to advanced. For example, this guide is aimed at the *Developer* persona.

This preface introduces you to the WebCenter Portal personas and describes the ways in which they might interact with WebCenter Portal. Each persona is assigned a default role provided out-of-the-box with WebCenter Portal. The default roles are given a unique set of permissions appropriate for the work that each persona will typically do. Note that you can modify these default roles or configure new roles to meet the unique needs of your organization.

The people who interact with WebCenter Portal typically work together as a team that is comprised of the following personas:

- Knowledge Worker
- Application Specialist
- Web Developer
- Developer
- System Administrator

Knowledge Worker

Karen is a *knowledge worker* who typically uses WebCenter Portal to contribute and review content, participate in social interactions, and leverage the Home portal to manage her own documents and profile.

At the application level, Karen has permissions such as those granted to the default `Authenticated-User` role, which may be customized for the specific needs of the organization. At the portal level, the portal manager will likely assign Karen a role that includes `View Pages` and `Customize Pages` permissions.

For more information about roles and permissions, see About Roles and Permissions for a Portal in *Building Portals with Oracle WebCenter Portal*.

Knowledge Worker Tasks in WebCenter Portal

Tasks that are typical of a knowledge worker like Karen include:

- Editing and updating pages for which she has been assigned content contribution permissions

- Connecting to and collaborating with other WebCenter Portal users by sharing information, files, and links; and by interacting through instant messaging, mail, message boards, discussions, wikis, and blogs

- Uploading, sharing, and managing documents stored in Content Server

- Joining a team or project portal

- Keeping up with changes in WebCenter Portal by receiving notifications when content is updated, viewing the activities of the portals she is a member of and users she's connected to, viewing announcements, participating in discussions, and monitoring WebCenter Portal RSS feeds

- Staying organized through the use of favorites, notes, calendars, lists, links to portal objects, and tags

As Karen becomes more familiar with the functionality available in WebCenter Portal, she may begin to perform more advanced tasks, such as creating portals. As a more advanced knowledge worker, her role may evolve to overlap with application specialist tasks.

Information targeted for knowledge workers like Karen is in Using Oracle WebCenter Portal. Advanced tasks that overlap with those of an application specialist are covered in Building Portals with Oracle WebCenter Portal.

Application Specialist

Ari is an *application specialist* who works in WebCenter Portal to create and administer portals, their structure (hierarchy of pages, navigation, security), and their content (components on a page, layout, behavior, and so on). In a typical project, Ari

coordinates the efforts of Karen (knowledge worker), Wendy (web developer), and Dave (developer).

At the application level, Ari has permissions such as those granted to the default `Application Specialist` role, which may be customized for the specific needs of the organization. In a portal that Ari creates, he performs actions available to the `Portal Manager` role to manage the portal.

For more information about roles and permissions, see About Roles and Permissions for a Portal in *Building Portals with Oracle WebCenter Portal*.

Application Specialist Tasks in WebCenter Portal

Tasks that are typical of an application specialist like Ari include:

- Planning and creating new portals

- Editing and administering the portals he owns

- Creating and building portal pages using the page editor and the resource catalog to add and configure page components

- Creating and managing portal assets, tools, and services

- Managing shared assets and portal templates across all portals

Information targeted for application specialists like Ari is in Building Portals with Oracle WebCenter Portal. To work with his personal view of the Home portal, Ari will also refer to Using Oracle WebCenter Portal.

Web Developer

Wendy is a *web developer* who focuses on delivering a consistent, branded look and feel to all portals. Wendy provides graphics designs and HTML markup from which Ari (application specialist in WebCenter Portal) or Dave (developer in JDeveloper) can create content or page style templates, skins, and so on. Once these assets are created, Ari can leverage them to create portal pages. Wendy typically does not interact with WebCenter Portal directly.

Web Developer Tasks in WebCenter Portal

Tasks that are typical of a web developer like Wendy include:

- Developing a corporate portal look and feel

- Designing new page templates

Information targeted for web developers like Wendy is in Creating a Look and Feel for Portals in *Building Portals with Oracle WebCenter Portal*.

Developer

Dave is a *developer* who is primarily responsible for developing components (such as task flows, page templates, and content templates), which are published and leveraged by Ari (the application specialist). Dave works with JDeveloper to develop and extend assets for use in WebCenter Portal.

Developer Tasks

Tasks that are typical of a developer like Dave include:

- Developing custom assets such page templates and resource catalogs for portals in WebCenter Portal

- Developing Java portlets

- Developing and deploying task flows, managed beans, and other custom components

- Developing custom personalization components

- Maintaining the source control system

- Maintaining a build system

Information targeted for developers like Dave is in Developing WebCenter Portal Assets and Custom Components with Oracle JDeveloper.

System Administrator

Syed is a *system administrator* who fields requests from IT employees and business users to set up new machines; clone or back up existing applications systems and databases; install patches, packages, and applications; and perform other administration-related tasks. As the system administrator, Syed works with other tools

such as Fusion Middleware Control and command line tools. He leverages Enterprise Manager to configure portal settings, and also configures integrations such as WebCenter Content and other Fusion Middleware products and Oracle applications.

In WebCenter Portal, he has permissions such as those granted to the default `Administrator` role, which provides exclusive access to administer and set global options for all portals (including the Home portal).

For more information about application level roles and permissions, see About Application Roles and Permissions in *Administering Oracle WebCenter Portal*.

System Administrator Tasks

Tasks that are typical of a system administrator like Syed include:

- Uses WebCenter Portal administration to administer all portals (including import and export of portals) and security site-wide

- Uses WebCenter Portal administration to manage site-wide system pages, business role pages, and personal pages

- Leads security, taxonomy, metadata, workflow, governance

- Uses the management console for administrative functions

- Executes command line utilities for administrative functions

- Installs and configures production versions of developers' efforts

- Performs patching of the production versions and the operating system

- Creates clones and backups of the production versions

- Performs restores of production versions

- Monitors the operating system for issues with the production version

- Deploys and redeploys applications

Information targeted for system administrators like Syed is in Administering Oracle WebCenter Portal and WLST Command Reference for WebLogic Server.

Part I

Getting Started

This part contains the following chapters:

- Introduction to Developing for Oracle WebCenter Portal

- Setting Up a Development Environment

1

Introduction to Developing for Oracle WebCenter Portal

This chapter introduces this guide, *Developing WebCenter Portal Assets and Custom Components with Oracle JDeveloper*, and briefly describes the major tasks that WebCenter Portal developers may be expected to perform.

This chapter includes the following topics:

- What Is the Purpose of This Guide?
- What Is My Role as a WebCenter Portal Developer?

1.1 What Is the Purpose of This Guide?

This chapter describes the intended audience and basic purpose of *Developing WebCenter Portal Assets and Custom Components with Oracle JDeveloper*.

This guide discusses developing WebCenter Portal assets, shared libraries and extensions to WebCenter Portal using Oracle JDeveloper. Many of the tasks described in this guide involve activities that require Java, CSS, Application Development Framework (ADF), Expression Language (EL), and related experience.

Major activities described in this guide include:

Setting Up Your Environment

See the chapters on setting up your environment and your team environment in Getting Started.

Working with Assets

The chapters in Working with WebCenter Portal Assets explain how to develop WebCenter Portal assets such as layouts, page styles, page templates, skins, Content Presenter templates, and visualization templates. Several other parts of this guide also contain chapters devoted to working with other types of assets such as portlets, pagelets, task flows, and shared libraries.

Working with Portlets

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 several different portlets can be placed on a single page, users benefit from a single-source experience even though, in reality, the content may be derived from multiple sources. See the chapters of Working with Portlets for detailed information.

Working with Pagelets

Pagelets are sub-components of a web page accessed through Pagelet Producer that can be injected into any proxied application. Any application on a Pagelet Producer

resource that returns markup can be registered as a pagelet, which can then be displayed in WebCenter Portal, or any other web application. See the chapters of Working with Pagelets for detailed information.

Working with Additional WebCenter Portal Customizations

You can provide additional functionality within WebCenter Portal by extending ADF assets, such as task flows, data controls, and managed beans, as a shared library, or provide page layout more suitable to mobile devices such as smartphones and tablets. For more information on these and other ways in which to customize WebCenter Portal, see the chapters in Additional WebCenter Portal Customizations.

1.2 What Is My Role as a WebCenter Portal Developer?

In some cases, WebCenter Portal may require custom components that do not exist out-of-the-box or cannot easily be created with WebCenter Portal's runtime. For example, design requirements might call for page templates, page styles, and skins that are branded and organized in a precise way. In this case, you may need to use ADF Faces and CSS to achieve the desired look and feel, and JDeveloper is well-suited to such development. Other use cases may include developing custom components that allow for interaction between portal elements and data (for example, using ADF task flows or portlets with event handling). For example, as a WebCenter Portal developer, you may be asked to create and configure:

- Portal infrastructure components like page templates and page styles

- Custom portal extensions like task flows, data controls, and backing beans

- WebCenter Portlet Producer applications

- Pagelet Producer pagelets

2

Setting Up a Development Environment

This chapter helps you get started using JDeveloper to develop assets and extensions to WebCenter Portal, providing guidance and tips on how to structure your development environment.

This chapter includes the following topics:

- Introduction to Setting up a Development Environment
- Basic Setup Tasks
- Setup Tasks Specific to WebCenter Portlet Producer Applications

2.1 Introduction to Setting up a Development Environment

This chapter distinguishes between tasks that are general and tasks that are application-specific. General tasks apply no matter what kind of application you are developing. For instance, installing JDeveloper and the WebCenter Portal extensions are general setup tasks. Other tasks described in this chapter apply only if you are developing specific kinds of applications, like WebCenter Portal server extensions or WebCenter Portlet Producer applications.

2.2 Basic Setup Tasks

This section describes setup tasks that do not depend on the kind of application you are developing.

- Installing Oracle JDeveloper
- Installing the WebCenter Portal Extensions for JDeveloper
- Setting the User Home Directory Environment Variable
- Managing the Integrated WebLogic Server
- Creating Application Resource Connections

2.2.1 Installing Oracle JDeveloper

Oracle JDeveloper provides an integrated development environment (IDE) for developing portals and custom portal components. For information on obtaining and installing Oracle JDeveloper, see the Oracle JDeveloper page on OTN at:

```
http://www.oracle.com/technetwork/developer-tools/jdev/overview/
index.html
```

2.2.2 Installing the WebCenter Portal Extensions for JDeveloper

The WebCenter Portal extensions is an add-in that provides JDeveloper with the complete set of WebCenter Portal capabilities and features. To install the WebCenter Portal extensions:

1. Start Oracle JDeveloper.

2. If the Select Default Roles dialog displays, select **Default Role** to enable all technologies, and click **OK**.

3. If a dialog opens asking if you want to migrate settings from an earlier version, click **No**.

4. From the **Help** menu, select **Check for Updates**.

    > **Note:**
    >
    > If you are behind a firewall, you may need to configure a proxy server to access the extensions. From the Updates wizard, a dialog appears in which you can enter your HTTP Proxy Server settings. Click the **Help** button in any dialog for more information. For detailed information, see the "Proxy Settings and JDeveloper" topic in the Oracle JDeveloper online help.

5. On the Source page, select **Search Update Centers**.

6. Check **Oracle Fusion Middleware Products** and **Official Oracle Extensions and Updates** and click **Next**.

7. From the generated list, search for the **WebCenter Core Design Time** and **WebCenter Framework and Services Design Time** extensions, select them, and then click **Finish**.

8. When prompted, restart JDeveloper.

See also *Installing and Configuring Oracle WebCenter Portal*.

2.2.3 Setting the User Home Directory Environment Variable

Oracle strongly recommends that you set an environment variable for the user home directory that is referenced by JDeveloper. By setting this variable, you can avoid receiving long pathname errors that are known to occur in some circumstances.

For detailed instructions on setting this variable on Windows, Linux, UNIX, and Mac OS X operating systems, see Setting the User Home Directory in *Installing Oracle JDeveloper*.

2.2.4 Managing the Integrated WebLogic Server

The following topics describe the Integrated WebLogic Server used by JDeveloper, and how to deploy and configure it:

*   What is the Integrated WebLogic Server?

*   Starting and Stopping Integrated WebLogic Server

- Configuring the JVM for the Integrated WebLogic Server

2.2.4.1 What is the Integrated WebLogic Server?

The Integrated WebLogic Server is a runtime service that references an instance of Oracle WebLogic Server and is bundled with JDeveloper. The Integrated WebLogic Server allows developers to run, test, and debug WebCenter Portal portlet producer and consumer applications from within JDeveloper. Note that you will need to publish and test assets within WebCenter Portal to determine their suitability within a particular portal and/or page.

2.2.4.2 Starting and Stopping Integrated WebLogic Server

For detailed information managing the Integrated WebLogic Server, see the topic "Deploying to Integrated WebLogic Server" in the JDeveloper Online Help. Other options, such as running in debug mode and logging into the WebLogic Server Administration Console, are also covered in that section.

2.2.4.3 Configuring the JVM for the Integrated WebLogic Server

Although it is not required, you have the option of changing the default Java Virtual Machine (JVM) settings the Integrated WLS in the `setDomainEnv.sh`. This file is located in `JDEV_SYSTEM_DIRECTORY/DefaultDomain/bin`.

The default memory values are:

```
-Xms512m -Xmx512m
```

When creating or referring to the `JDEV_SYSTEM_DIRECTORY`, keep in mind that, on a Windows platform, a WebCenter domain name cannot contain spaces, and the domain cannot be created in a folder that has a space in its path. Therefore, ensure that `DOMAIN_HOME` and `JDEV_SYSTEM_DIRECTORY` paths do not contain spaces.

2.2.5 Creating Application Resource Connections

Connections allow applications to access external data and services. For example, to use the Content Presenter task flow to display content from an Oracle WebCenter Content Server repository, you need to configure a connection to the repository. If you intend to consume portlets from a portlet producer, you need to configure the producer connection.

> **Tip:**
>
> A good practice is to create and test your connections once and check them into your source control system. Then, other developers on your team can check out the connections and use them. This technique also allows your team to keep in sync whenever a connection changes.

This section discusses connections and describes the different ways to access the wizards for creating new connections.

2.2.5.1 Where Are Connections Located?

Depending on how you invoke a wizard to create a connection, connections are placed 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 features, 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 chapters on tools and services for WebCenter Portal.

- Under IDE Connections in the Resource Palette

  Connections created here can be reused across 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.

2.2.5.2 How Do I Access the Connection Wizards?

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.

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.

2.3 Setup Tasks Specific to WebCenter Portlet Producer Applications

WebCenter Portal provides a variety of ready-to-use portlets that you can add to your portal pages. This section provides a brief description of the preconfigured producers and the portlets they provide. It contains the following subsections:

- Deploying the Preconfigured Portlet Producers

- Accessing OmniPortlet

- WSRP Sample Portlet Producers and Portlets

- PDK-Java Sample Portlet Producer and Portlets

For more information about WebCenter Portlet Producer applications, see Introduction to Portlets.

2.3.1 Deploying the Preconfigured Portlet Producers

The preconfigured portlet producers are not deployed by default, you must deploy them manually, as explained in this section.

> **Note:**
>
> These producer applications consume memory and require CPU time during deployment. This contributes to the WebLogic Server startup time and base memory footprint. For these reasons, it makes sense to only deploy the applications you need. For more information about tuning the deployment of internal applications, see On-demand Deployment of Internal Applications in *Deploying Applications to Oracle WebLogic Server*.

1. If it is not running, start the Integrated WebLogic Server. The server must be running for this feature to be enabled.

2. From the Run menu, select **WebCenter Portal Deployments**.

3. In the WebCenter Portal Deployments dialog (Figure 2-1), select the producer to deploy (or undeploy) and click **OK**. Each of the preconfigured producers is described in more detail in the following sections.

Figure 2-1   WebCenter Portal Deployments Dialog
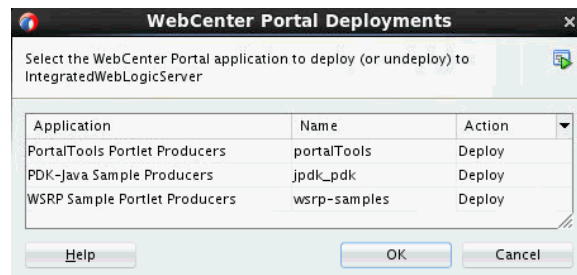
2.3.2 Accessing OmniPortlet

The Integrated WLS contains PortalTools, which provides access to the *design time at runtime* OmniPortlet 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.

To access the OmniPortlet portlet producer:

1. Start the Integrated WebLogic Server and open the WebCenter Portal Deployments dialog as described in Deploying the Preconfigured Portlet Producers.

2. Select **Portal Tools Portlet Producers** and click **OK** to deploy them to the Integrated WebLogic Server.

   The Deployment - Log tab in JDeveloper displays a link that opens the PortalTools Welcome page, from where you can open two Portal Tools test pages—one for the OmniPortlet producer and one for the Sample Portlet Producers.

- 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, and web pages. OmniPortlet users can also choose a prebuilt layout for the data. Prebuilt layouts include tabular, news, bullet, form, chart, or HTML. For more information about OmniPortlet, see Working with OmniPortlet in *Building Portals with Oracle WebCenter Portal*.

- The **Sample Portlet Producer** includes portlets built with OmniPortlet that are for demonstration purposes only. The portlet samples include a scrolling RSS portlet and a simple RSS portlet. Note that the sample producer is for demonstration and should not be used to create real-use portlet instances.

3. On the PortalTools Welcome page, copy the OmniPortlet Producer link, or the Sample Portlet Producer link, and use it to register OmniPortlet with your portal. For information about registering a portlet producer, see Registering an Oracle PDK-Java Portlet Producer in *Administering Oracle WebCenter Portal*.

2.3.3 WSRP Sample Portlet Producers and Portlets

The Integrated WLS includes sample WSRP portlet producers and portlets you can use with your application.

> **Note:**
>
> You can find the source code for the sample portlets is in the following EAR file:
>
> *JDEV_HOME*/oracle_common/modules/oracle.wccore/wsrp-samples.ear
>
> where *JDEV_HOME* is the location where JDeveloper is installed on your machine.

To access the WSRP sample portlet producers:

1. Start the Integrated WebLogic Server and open the WebCenter Portal Deployments dialog as described in Deploying the Preconfigured Portlet Producers.

2. Select **WSRP Sample Portlet Producers** and click **OK** to deploy them to the Integrated WebLogic Server.

   The Deployment - Log tab in JDeveloper displays a link that opens the WSRP Sample Producer Test Page.

3. Copy the Web Services Description Language (WSDL) URL for a WSRP producer —either WSRP v1 WSDL or WSRP v2 WSDL— and use it to register the producer. For more information about registering portlet producers, see Managing Portlet Producers in *Administering Oracle WebCenter Portal*.

2.3.4 PDK-Java Sample Portlet Producer and Portlets

The Integrated WLS includes sample PDK-Java portlet producers and portlets you can use to familiarize yourself with the types of functionality that are available through PDK-Java portlets.

To access PDK-Java sample portlet producers:

1. Start the Integrated WebLogic Server and open the WebCenter Portal Deployments dialog as described in Deploying the Preconfigured Portlet Producers.

2. Select **PDK-Java Sample Portlet Producers** and click **OK** to deploy the PDK-Java Sample Producers Test Page to the Integrated WebLogic Server.

   The Deployment - Log tab in JDeveloper displays a link that opens the provider test page.

3. Open a test page, copy the link and use it to register the producer.

   For information about registering a portlet producer, see Managing Portlet Producers in *Administering Oracle WebCenter Portal*.

Part II

Working with WebCenter Portal Assets

This part contains the following chapters:

- Introduction to WebCenter Portal Assets

- Working with the Portal Asset Application Template

- Developing Layouts

- Developing Page Styles

- Developing Page Templates

- Developing Skins

- Developing Visualization Templates

- Developing Content Presenter Display Templates

3

Introduction to WebCenter Portal Assets

This chapter describes the different assets available for you to use in WebCenter Portal, and an overview of the developer's role in developing and extending those assets.

This chapter includes the following topics:

- About Working with WebCenter Portal Assets
- Developing Assets for WebCenter Portal

3.1 About Working with WebCenter Portal Assets

WebCenter Portal provides various built-in assets that users can leverage in their portals. Copies of these built-in assets can be modified within WebCenter Portal to meet specific local requirements. For more information about modifying built-in assets in WebCenter Portal, see Creating Assets in *Building Portals with Oracle WebCenter Portal*.

Some assets, such as page styles and layouts, cannot be created in WebCenter Portal. These resources must be created in JDeveloper and deployed to WebCenter Portal. In some cases, WebCenter Portal may also not be able to add all the required functionality.

WebCenter Portal assets rely on three application templates that can be used to create or modify asset applications in JDeveloper:

- WebCenter Portal Asset template – used to create or modify assets for page templates, skins, layouts, page styles, Content Presenter templates, and Visualization templates. The resulting asset applications include a default WebCenter Portal Asset Archive deployment profile that you can use to deploy the asset to WebCenter Portal across a Portal Server connection. For more information about creating an asset application for these types of assets, see Working with the Portal Asset Application Template.

- WebCenter Portlet Producer Application template – used to develop portlets based on the JSR 286 standards. For more information, see Introduction to Portlets.

- WebCenter Portal Server Extension template – used to develop extensions to WebCenter Portal Server. For more information about developing extensions for Portal Server, see Developing Shared Libraries.

Finally, you can use WebCenter Portal's Pagelet Producer to create pagelets. Note that Pagelet Producer is a standalone application that can be accessed directly through its URL, or from the **Shared Assets** tab in WebCenter Portal. For information about accessing and using Pagelet Producer, see Introduction to Pagelets.

3.2 Developing Assets for WebCenter Portal

This section includes the following topic:

- Setting Up Your JDeveloper Environment for Asset Development

3.2.1 Setting Up Your JDeveloper Environment for Asset Development

Before you start, complete the following steps:

1. Download and install Oracle JDeveloper 12*c* (12.2.1.1).

    For details, see Installing Oracle JDeveloper.

2. Install the WebCenter Portal Extension for JDeveloper (12.2.1.1).

    For details, see Installing the WebCenter Portal Extensions for JDeveloper.

3. Restart JDeveloper if it is open.

4

Working with the Portal Asset Application Template

This chapter provides an overview of how to create, modify, and publish assets for use in WebCenter Portal using the WebCenter Portal Asset application template.

This chapter includes the following topics:

- About the WebCenter Portal Asset Application Template
- Creating a WebCenter Portal Asset Application
- Creating and Editing WebCenter Portal Assets
- Publishing WebCenter Portal Assets
- Testing WebCenter Portal Assets

4.1 About the WebCenter Portal Asset Application Template

You can use the WebCenter Portal Asset application template to create asset applications for the following asset types:

- **Layouts** define the basic properties of a page, such as the number and relationship of columns, and are part of a page's page style. For more information, see Developing Layouts.

- **Page Styles** define the layout of a newly created page, and may also dictate the type of content the page supports. For more information, see Developing Page Styles.

- **Page Templates** define the interface that surrounds page content and help to apply a consistent look and feel across groups of pages. For more information, see Developing Page Templates.

- **Skins** define the appearance and look and feel, including colors and fonts, of a specific portal or the entire application. For more information, see Developing Skins.

- **Visualization Templates** determine the layout of content in a task flow that is created at runtime. Visualization templates can also include objects that can be bound to any data visualization that is added to the task flow. For more information, see Developing Visualization Templates.

- **Content Presenter Display Templates** define templates for displaying content. For more information, see Developing Content Presenter Display Templates.

The resulting asset applications include a default WebCenter Portal Asset Archive deployment profile that you can use to publish the asset to WebCenter Portal across a Portal Server Connection.

4.2 Creating a WebCenter Portal Asset Application

To create or edit WebCenter Portal assets in JDeveloper that can be published to WebCenter Portal, you start by creating a WebCenter Portal asset application using the WebCenter Portal Asset Application template. The application defines the *asset type* (for example, a page template) that you want to create or modify. Within that application, you can then define or edit properties for the asset, and finally, publish it to WebCenter Portal as a shared or portal-specific asset.

This section includes the following topics:

- How to Create a WebCenter Portal Asset Application

- What You May Need to Know About Asset Application Artifacts

4.2.1 How to Create a WebCenter Portal Asset Application

To create a WebCenter Portal asset application:

1. From the Navigation pane, click **New Application** (or click the **New Gallery** icon and select **Application**).

    The New Gallery displays (Figure 4-1).

    Figure 4-1   New Gallery
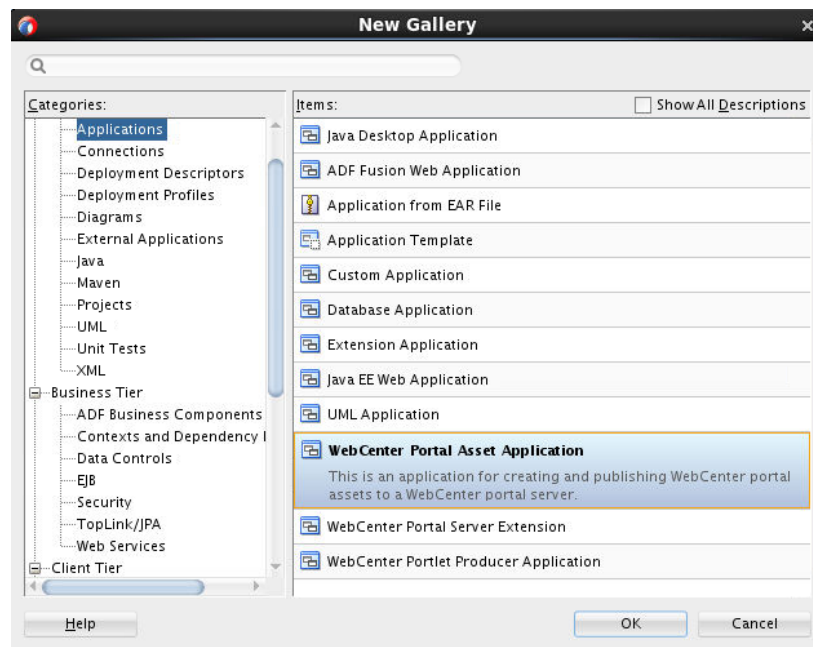
2. Select **WebCenter Portal Asset Application** and click **OK**.

    The Application Name page of the Create WebCenter Portal Application wizard displays (Figure 4-2).

Figure 4-2   Create WebCenter Portal Asset Application - Step 1
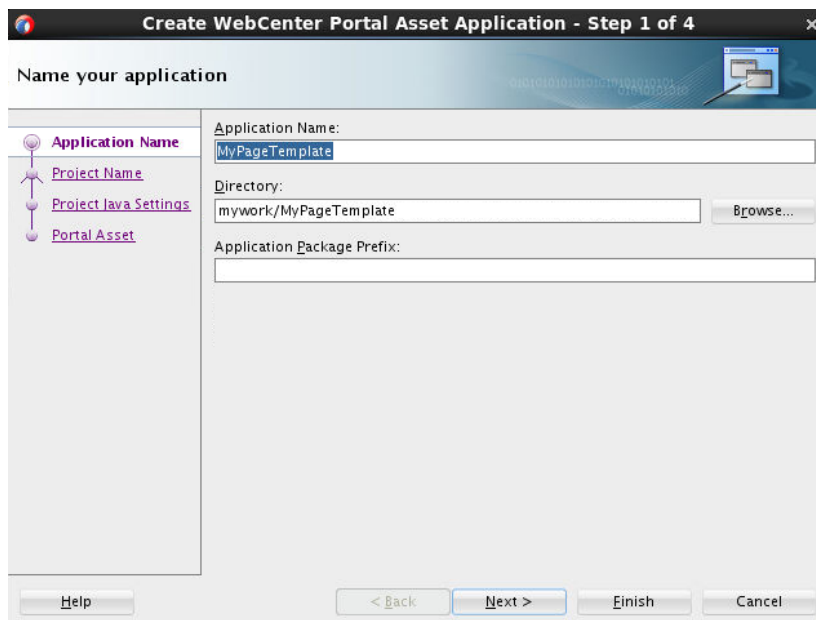
3. Enter an **Application Name**, optionally choose a **Directory** other than the default to store your application, and then click **Next**.

The Project name page of the Create WebCenter Portal Application wizard displays (Figure 4-3).

Figure 4-3   Create WebCenter Portal Asset Application - Step 2
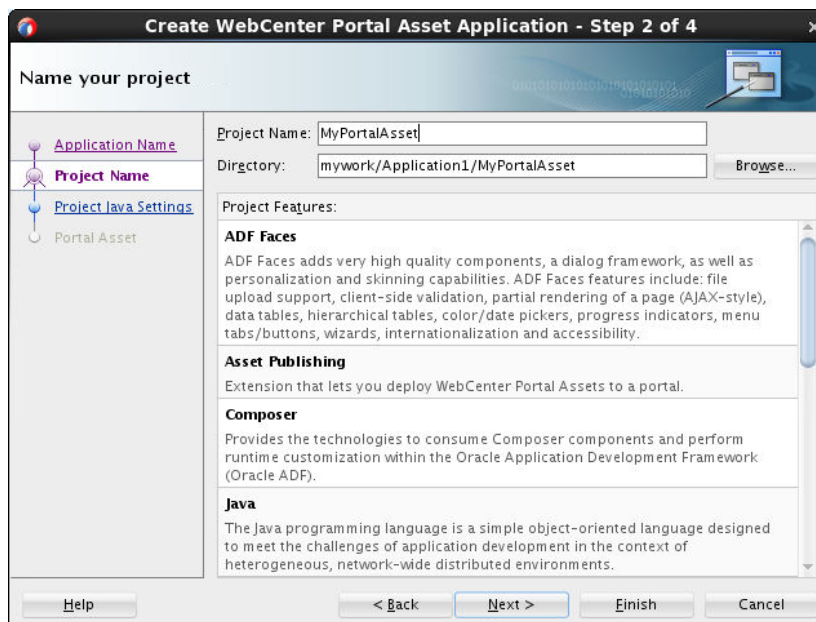
4. Enter a **Project Name** for the project and click **Next**.

You can optionally change the **Directory** to store your project in, but note that the **Project Name** is automatically appended to the path. The Project Java Settings page of the Create WebCenter Portal Application wizard displays (Figure 4-4).

Figure 4-4   Create WebCenter Portal Asset Application - Step 3
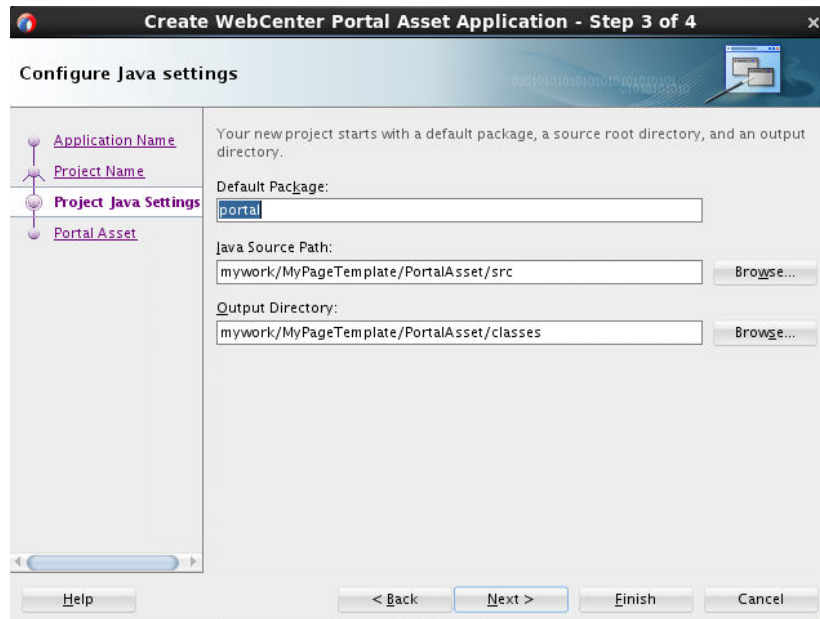
5. Optionally update the Project Java Settings fields for the asset project and click **Next**.

The Portal Asset page of the Create WebCenter Portal Application wizard displays (Figure 4-5).

Figure 4-5   Create WebCenter Portal Asset Application - Step 4
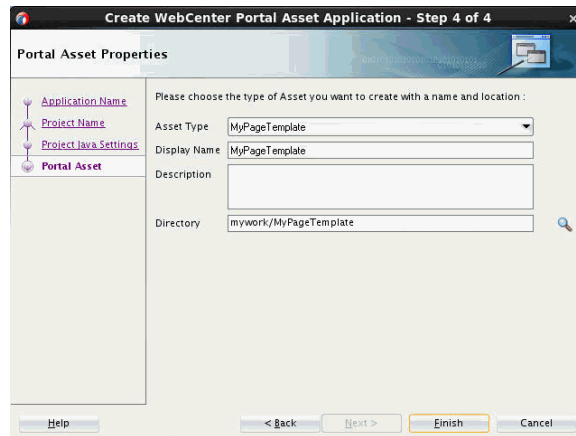
6. Select the **Asset Type** (for example, `Page Template`), optionally change the **Display Name** (note that your changes will be applied to the directory path and file name), optionally add a **Description** for the asset project, optionally change the **Directory** where the asset will be created, and click **Finish**. After the asset application has been created it appears in the Navigation pane (Figure 4-6).

Figure 4-6   Newly Created Portal Asset Application
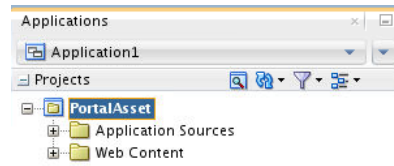
4.2.2 What You May Need to Know About Asset Application Artifacts

When you create a WebCenter Portal asset application, you create a single named workspace and a single named project (`PortalAsset` by default) in that workspace. A WebCenter Portal asset application project follows JDeveloper's standard for a Web application. It has the same structure and contains all of the features as an ADF Fusion Web application's ViewController project plus the following WebCenter Portal specific features:

- Asset Publishing - lets you publish an asset application to WebCenter Portal.

- Page Editor- lets you consume page components and perform runtime customizations within the Oracle Application Development Framework (Oracle ADF).

- Customization Components - also lets you consume Composer components and perform runtime customizations within the Oracle Application Development Framework (Oracle ADF).

The asset project is seeded with Fusion ViewController's default JSP Tag and JDeveloper libraries plus some WebCenter Portal specific JSP Tag and JDeveloper libraries. It contains a Default Package named `portal`, which is also used, for example, as the folder name for `DataBindings.cpx`.

After an asset application has been created, you can change the values for asset-specific properties, such as `displayName` and the `Description`, by editing the `assetDef.xml` file under `META-INF/assets/<assetName>`.

4.3 Creating and Editing WebCenter Portal Assets

After creating an asset application, as described in Creating a WebCenter Portal Asset Application, you are now ready to define or modify properties for the asset. Table 4-1 shows where to find further information about creating and modifying the different types of assets. After creating the asset, continue by publishing it to WebCenter Portal (see Publishing WebCenter Portal Assets) where you can test and debug it (see Testing WebCenter Portal Assets).

Table 4-1   Creating WebCenter Portal Assets

Asset Type	More Information
Layouts	Developing Layouts
Page Styles	Developing Page Styles
Page Templates	Developing Page Templates

Table 4-1 (Cont.) Creating WebCenter Portal Assets

Asset Type	More Information
Skins	Developing Skins
Visualization Templates	Developing Visualization Templates.
Content Presenter Display Templates	Developing Content Presenter Display Templates

4.4 Publishing WebCenter Portal Assets

This section describes the steps for publishing WebCenter Portal assets to WebCenter Portal as a shared or portal-specific asset.

This section contains the following topics:

- Creating a WebCenter Portal Server Connection

- Publishing a WebCenter Portal Asset

4.4.1 Creating a WebCenter Portal Server Connection

Before you can publish an asset to WebCenter Portal as a shared asset or to a specific portal, you need to set up a connection to the target Portal Server. Use the WebCenter Portal Server connection dialog to configure a connection to a Portal Server. You can set the connection up either as an application resource, or as an IDE connection that can be reused in other asset applications.

To create a WebCenter Portal Server connection:

1. Open the Portal Server Connection dialog by doing one of the following:

- From the Resources Catalog, click **New** and select **IDE Connections > WebCenter Portal Server...**

- From the Application Resources pane, right-click **Connections** and select **New Connection > WebCenter Portal Server...**

The WebCenter Portal Server Connection dialog displays (Figure 4-7):

Figure 4-7   WebCenter Portal Server Connection Dialog

2. For the **Create connection in** options, choose **Application Resources** to configure a Portal Server connection specific to the current application, or choose **IDE Connections** to save the connection as an IDE resource that can be used for other applications.

   Tip:

   If you will be creating and publishing multiple assets, Oracle recommends that you choose to save the connection in IDE Connections. To add an IDE connection to an asset application, select it from the IDE Connections node in the Resource Catalog, and either right-click the connection and select **Add to Application**, or drag it to the Connections node under the Application Resources node in the Applications panel.

3. Enter a name for the Portal Server connection you are setting up. Note that only alphanumeric characters can be used.

4. Enter the **Host** name and **Port** assignment for the Portal Server you are connecting to. For the host, you can enter either the fully qualified domain name, `localhost`, or the server name.

5. Enter a valid username and password for the Portal Server you are connecting to.

6. After completing your entries, click **Test Connection** to make sure the connection works. Note that if the connection test fails due to the server being offline, the connection will still be set up, and can be used once the server is again available.

   Tip:

   Although the **Service Path** field is not editable, you can use the field to copy and paste into a browser to determine the REST endpoint for the server connection.

4.4.2 Publishing a WebCenter Portal Asset

After creating a WebCenter Portal asset, the next step is to publish it and test it in WebCenter Portal.

To publish a WebCenter Portal asset:

1. Right-click the Portal Asset project node and select **Deploy** and the default deployment profile to deploy (Figure 4-8).

Figure 4-8   Deploying a WebCenter Portal Asset
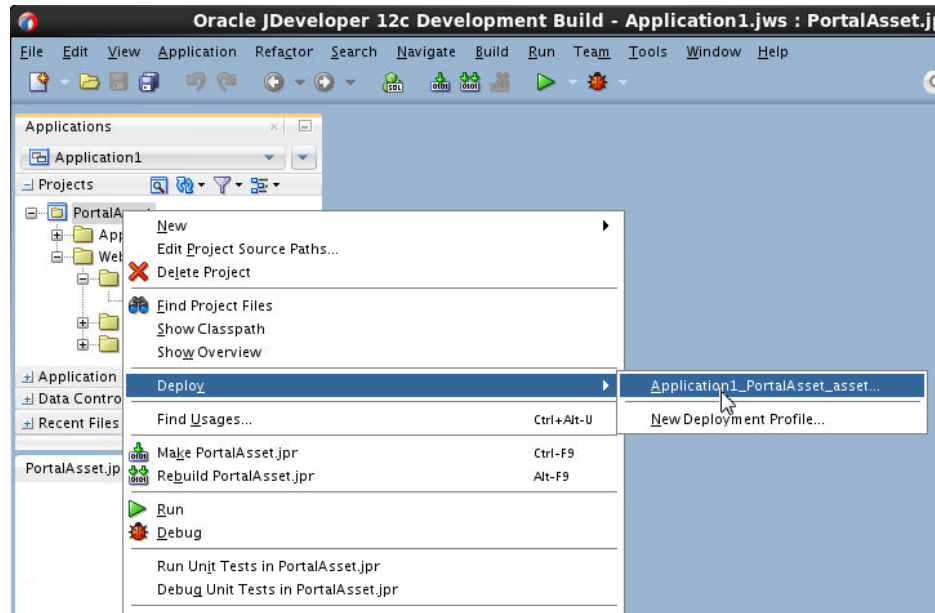
The Deploy Asset wizard displays (Figure 4-9).

Figure 4-9   Deploy Asset Wizard - Deployment Action Page
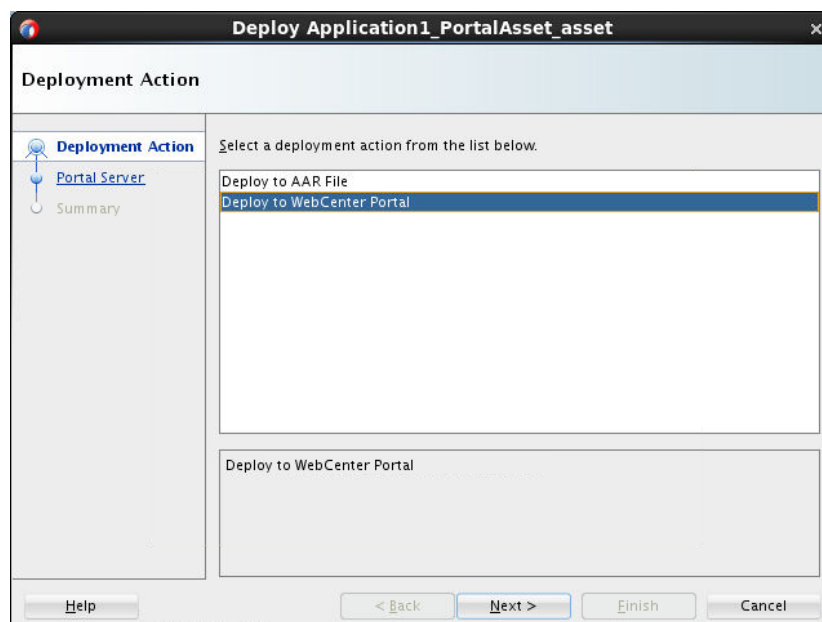
2. On the Deploy Action page, select **Deploy to WebCenter Portal** and click **Next**.

The Portal Server page displays (Figure 4-10).

Figure 4-10   Deploy Asset Wizard - Portal Server Page
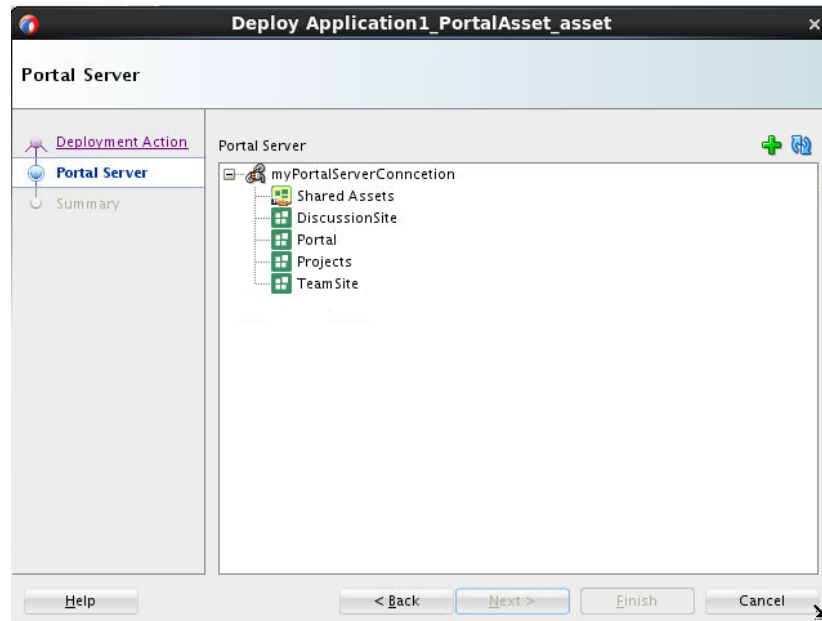
3. If you've configured a Portal Server connection, select **Shared Assets** or the specific portal for the Portal Server connection to which to publish the asset. If you have not previously configured a Portal Server connection, click the **Add** icon and complete the Portal Server connection fields as shown in Creating a WebCenter Portal Server Connection.

4. Click **Finish** to deploy the asset, or click **Summary** to see a summary of your deployment selections before deploying.

5. Open the Deployment - Log pane and check the status of the deployment. You can also click on the target URL on the Portal Server to view the recently deployed asset.

4.5 Testing WebCenter Portal Assets

After creating or modifying an asset application and publishing the asset to WebCenter Portal, you can continue by testing the runtime results. Note that you must test the asset within WebCenter Portal's runtime. You can iterate by modifying the asset in the JDeveloper, republishing to Portal Server, and then validating in WebCenter Portal. There are two ways to republish: you can either right-click on the **Portal Asset project node > Deploy > use the recently used deployment item**, or use the default **Alt-Shift-P** shortcut key.

> **Note:**
>
> To change or view shortcut key settings, go to **Tools > Preferences > Shortcut Keys >** and search for "Republish Portal Asset" or other shortcut key.

To help with the debug process, you can ask an Administrator (or someone with Administrator rights) for the test or staging environment to configure error messages and the calling stack to display in the runtime. For more information, see Viewing and Configuring Error Messages in WebCenter Portal in *Administering Oracle WebCenter Portal*.

5

Developing Layouts

This chapter describes how to create or modify page layouts and publish them to WebCenter Portal as a shared or portal-specific asset.

This chapter includes the following topics:

- About Developing Layouts
- Creating a Layout
- Editing a Layout
- Publishing a Layout

5.1 About Developing Layouts

When you create a new page, you select a page style for that page. Every page style includes a default layout, which is inherited by the new page. WebCenter Portal provides several built-in layouts that you can select on the **Shared Assets** or **Assets** pages.

If you want to change the layout for a page or page style, you can select a new existing layout for any page, or you can create a custom layout, and publish it as a shared or portal-specific asset. You can do this by either making a copy of a built-in layout and editing it in WebCenter Portal, or you can create a WebCenter Portal asset application, copy and paste the built-in layout and modify it in JDeveloper.

This chapter describes how to create or modify page layouts in a WebCenter Portal asset application, and publish them to WebCenter Portal as a shared or portal-specific asset. For information about modifying layouts in WebCenter Portal, see Working with Layouts in *Building Portals with Oracle WebCenter Portal*.

5.2 Creating a Layout

This section describes how to create a WebCenter Portal asset application for a page layout, and the artifacts that are created along with it.

This section includes the following topics:

- How to Create a Layout
- What You May Need to Know About Layout Artifacts

5.2.1 How to Create a Layout

This section describes how to create a WebCenter Portal Asset application for a new layout. Once the asset application with its associated artifacts has been created, you can continue by modifying the layout as described in Editing a Layout.

> **Note:**
>
> Oracle recommends that you store asset-related artifacts, such as images and icons, on your content server and that you create a folder structure on your content server specifically for asset artifacts so that content is easy to identify and move, if required.

To create a custom layout:

1. Create an asset application for the asset specifying `Layout` as the **Asset Type**.

   For more information about creating WebCenter Portal asset applications, see Creating a WebCenter Portal Asset Application. For information about the artifacts that are created when you create an asset application for a layout, see What You May Need to Know About Layout Artifacts.

2. In the Application Navigator, right-click the newly created layout's CSS and JSPX files, and choose **Open**. If you are using an existing WebCenter Portal layout as a starting point, copy and paste the source into the corresponding file in JDeveloper (for more information about using an existing asset as a starting point, see Copying an Asset in *Building Portals with Oracle WebCenter Portal*).

3. Continue by adding or modifying the layout as described in Editing a Layout.

5.2.2 What You May Need to Know About Layout Artifacts

Creating a layout asset application produces a default layout, with the following artifacts:

- A CSS file (for example, `Layout1.css`)

- A JSF file (for example, `Layout1.jspx`)

Both of these files appear in the Navigation bar as shown in Figure 5-1.

Figure 5-1   Layout Asset Application Artifacts
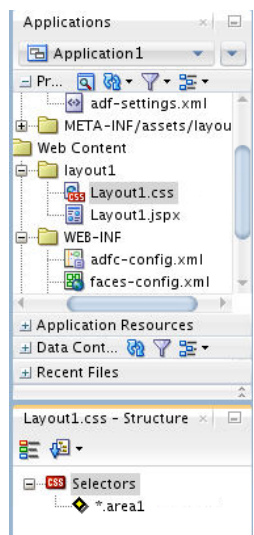
The following examples show the JSPX file and corresponding CSS file for the built-in three column (Three Columns) layout from the **Shared Assets** page in WebCenter Portal.

Example Three-Column JSPX File

```
<?xml version='1.0' encoding='UTF-8'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1" xmlns:f="http://
java.sun.com/jsf/core" xmlns:h="http://java.sun.com/jsf/html" xmlns:af="http://
xmlns.oracle.com/adf/faces/rich">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <af:componentDef var="attrs" componentVar="comp">
        <af:xmlContent>
            <component xmlns="http://xmlns.oracle.com/adf/faces/rich/component">
                <display-name>ThreeColumnFlow</display-name>
                <facet>
                    <facet-name>area1</facet-name>
                </facet>
                <facet>
                    <facet-name>area2</facet-name>
                </facet>
                <facet>
                    <facet-name>area3</facet-name>
                </facet>
            </component>
        </af:xmlContent>
        <af:group id="threeColumnFlow">
            <af:resource type="css" source="${pageDocBean.layoutCssPath}"/>
            <div xmlns="http://www.w3.org/1999/xhtml" class="area1">
                <af:facetRef facetName="area1"/>
            </div>
            <div xmlns="http://www.w3.org/1999/xhtml" class="area2">
                <af:facetRef facetName="area2"/>
            </div>
            <div xmlns="http://www.w3.org/1999/xhtml" class="area3">
                <af:facetRef facetName="area3"/>
            </div>
        </af:group>
    </af:componentDef>
</jsp:root>
```

Example Three-Column Layout CSS File

```
/* one column */
.area1 {
}

.area2 {
}

.area3 {
}


/* two column with area3 under area2 */
@media only screen and (min-width : 481px) {
  .area1 {
    position:absolute;
    left: 0px;
    width:50%;
  }
  .area2 {
    position:relative;
    left:50%;
    width:50%;
  }
```

```
    .area3 {
      position: relative;
      left: 50%;
      width:50%;
    }
}

/* three column */
@media only screen and (min-width: 769px) {
  .area1 {
    position: static;
    left: auto;
    float:left;
    width:33%;
  }

  .area2 {
    position: static;
    left: auto;
    float:left;
    width:34%;
  }

  .area3 {
    position: static;
    left: auto;
    float:left;
    width:33%;
  }
}
```

5.3 Editing a Layout

After creating the asset application for the layout, as described in Creating a Layout, continue by modifying the CSS and JSPX files.

> **Note:**
>
> Oracle recommends that you store asset-related artifacts, such as images and icons, on your content server and that you create a folder structure on your content server specifically for asset artifacts so that content is easy to identify and move, if required.

To edit a layout:

1. In the Application Navigator, right-click the layout's CSS and JSPX files and choose **Open**.

2. If you are using an existing WebCenter Portal layout (that is, other than the one created by default when you create the asset application) as a starting point, copy and paste the source for the CSS and JSPX files into the corresponding file in JDeveloper (see Copying an Asset in *Building Portals with Oracle WebCenter Portal*).

3. In the source code for the layout, make the appropriate changes. .

4. Save the CSS and JSPX files.

5. Publish and test the layout as shown in Publishing a Layout.

5.4 Publishing a Layout

After creating a WebCenter Portal asset application and making the changes you want to the layout, the next step is to publish it and test it in WebCenter Portal. For instructions on how to publish a layout to WebCenter Portal as a shared or portal-specific asset, see Publishing WebCenter Portal Assets.

6

Developing Page Styles

This chapter describes how to use Oracle JDeveloper to create, edit, and publish page styles for use in WebCenter Portal.

This chapter includes the following topics:

- About Developing Page Styles

- Best Practices for Creating Page Styles

- Creating a Page Style

- Editing a Page Style

- Publishing a Page Style

6.1 About Developing Page Styles

A page style is a JSPX page used for pages created at runtime. A page style describes the layout of a newly created page and may also dictate the type of content that the page supports.

When a user creates a page using a page style, the layout and initial content are copied from the page style to the newly created page. Unlike page templates, page styles are not reference-based. That is, if you change a page style, the change is not inherited in pages that use the style.

Typically, a page style contains components that enhance the usefulness and appearance of a given page. These include an in-place HTML text editor, images, layout boxes, hyperlinks, and so on. Content contributors can further populate the page with content. Figure 6-1 shows a sample portal page that is based on a page style.

Figure 6-1   Sample Page Using a Page Style
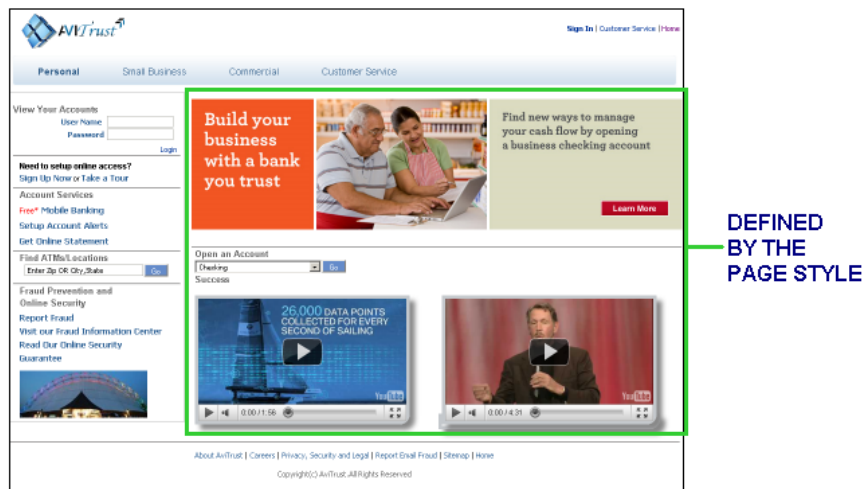
You can create page styles in either JDeveloper, or from the WebCenter Portal **Shared Assets** page (or **Assets** page for individual portals), to use at runtime to create pages. In the runtime, a user creates pages based on the available page styles. When they choose **Add Page**, the Select a Style dialog displays a set of predefined styles, as shown in Figure 6-2. The user chooses a style and creates a page based on that style. As the layout is already in place in the new page, the user only needs to add content to different areas of the page.

Figure 6-2   Select a Style Dialog
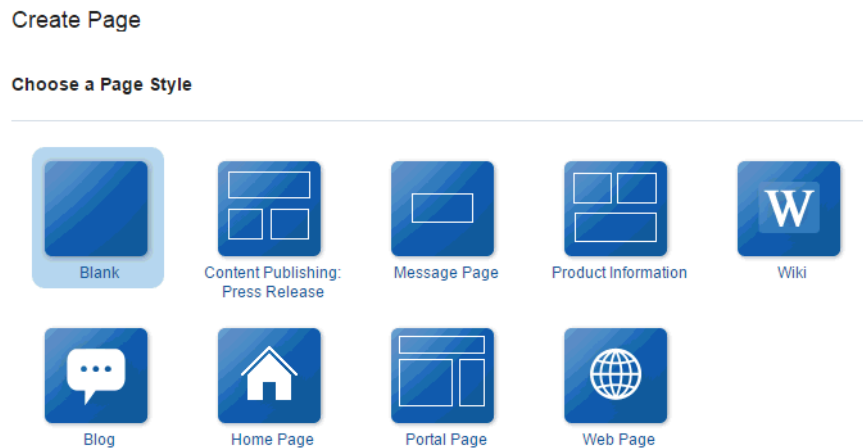
6.2 Best Practices for Creating Page Styles

The following prerequisites and guidelines may be helpful when developing page styles to be exposed as WebCenter Portal assets:

- If a page style is based on a page template, then while designing the page style JSF (`.jspx`) file, add the `content` facet, or any other facet defined in the page template, to contain page content.

- To make the page customizable, add a `Page Customizable` component within the content facet. As the `Page Customizable` contains a child

panelCustomizable component by default, users can add content to the page at runtime.

- To enable dynamic selection of the page template to use for the page style, in the page style JSF (.jspx) file, specify that the page template is defined in the page style's page definition, as follows:

```
<af:pageTemplate value="#{bindings.pageTemplateBinding.templateModel}" id="T">
```

You can then specify an EL value in the page style page definition to retrieve the name of the page template from the default page template setting, as follows:

```
<page viewId="#{preferenceBean.defaultPageTemplateViewId}"
id="pageTemplateBinding" Refresh="ifNeeded"/>
```

At runtime, users with the appropriate permissions can switch to a different page template by selecting a new default page template in the runtime administration console. For more information, see Choosing a Default Page Template in *Administering Oracle WebCenter Portal*.

- You can create a managed bean that controls which page template to use, as shown in the following example:

Example: Managed Bean to Control Page Template

```
public class siteTemplatesManager {
final private String templateA ="/templateA.jspx";
 final private String templateB ="/templateB.jspx";
 private String currentSiteTemplateViewId;

public siteTemplatesManager() {
super();
 currentSiteTemplateViewId =templateA;
 }
public String gettemplateViewId() {
return currentSiteTemplateViewId;
 }
public void settemplateAViewId(ActionEvent ae) {
currentSiteTemplateViewId =templateA;
 }
public void settemplateBViewId(ActionEvent ae) {
currentSiteTemplateViewId =templateB;
 }
}
```

6.3 Creating a Page Style

This section describes how to create a WebCenter Portal asset application for a new page style. Once the asset application with its associated artifacts has been created, you can continue by modifying the page style's JSPX file as described in Editing a Page Style.

This section includes the following topics:

- How to Create a Page Style

- What You May Need to Know About Page Style Artifacts

6.3.1 How to Create a Page Style

This section describes how to create a WebCenter Portal asset application for a new page style. Once the asset application with its associated artifacts has been created, you can continue by modifying the page style's JSPX file as described in Editing a Page Style.

To create a custom page style:

1. Create a WebCenter Portal asset application for the asset, selecting `Page Style` as the **Asset Type**.

    See Creating a WebCenter Portal Asset Application for more information about creating WebCenter Portal asset applications. For information about the artifacts that are created when you create an asset application for a page style, see What You May Need to Know About Page Style Artifacts.

2. In the Application Navigator, right-click the newly created page style's JSPX file, and choose **Open** (Figure 6-3).

    Figure 6-3   Page Style JSPX File
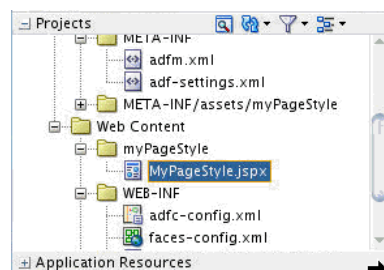
3. Continue by adding to or modifying the page style as shown in Editing a Page Style.

6.3.2 What You May Need to Know About Page Style Artifacts

Creating a page style asset application produces a default page style, with the following artifacts:

*   A JSP file containing the page elements for the page style (for example, `PageStyle1.jspx`)

*   The corresponding page definition file for the page style (for example, `PageStyle1PageDef.xml`)

Both of these files appear in the Navigation bar as shown in Figure 6-4.

Figure 6-4   Page Style Asset Application Artifacts
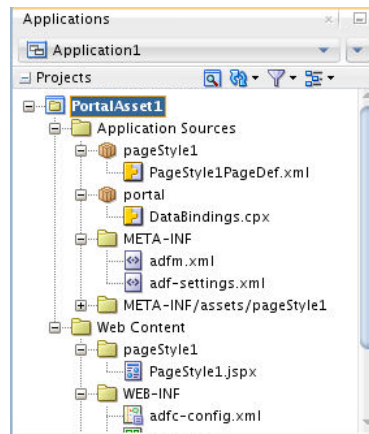
The following examples show the page style code for the JSPX file and associated page definition file for the built-in Three Column page style, which creates a basic page designed to flow and provide three columns.

JSPX File for Three Column Page Style

```
<?xml version='1.0' encoding='UTF-8'?>
<jsp:root version="2.1" xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
xmlns:cust="http://xmlns.oracle.com/adf/faces/customizable" xmlns:f="http://
java.sun.com/jsf/core" xmlns:jsp="http://java.sun.com/JSP/Page" xmlns:trh="http://
myfaces.apache.org/trinidad/html">
    <jsp:directive.page deferredSyntaxAllowedAsLiteral="true"/>
    <jsp:directive.page contentType="text/html;charset=utf-8"/>
    <f:view>
        <af:document id="docrt" title="#{pageDocBean.title}">
            <f:facet name="metaContainer">
                <trh:meta content="#{bindings.SEO_KEYWORDS}" name="keywords"/>
            </f:facet>
            <af:form id="f1" usesUpload="true">
                <af:pageTemplate id="T"
value="#{bindings.shellTemplateBinding.templateModel}">
                    <f:facet name="content">
                        <af:panelGroupLayout id="pgl1"
inlineStyle="replace_with_inline_style" layout="scroll"
styleClass="replace_with_scheme_name">
                            <af:declarativeComponent id="dclay"
viewId="#{pageDocBean.layoutViewId}">
                                <f:facet name="area1">
                                    <cust:panelCustomizable id="pcarea1" layout="auto"/>
                                </f:facet>
                                <f:facet name="area2">
                                    <cust:panelCustomizable id="pcarea2" layout="auto"/>
                                </f:facet>
                                <f:facet name="area3">
                                    <cust:panelCustomizable id="pcarea3" layout="auto"/>
                                </f:facet>
                            </af:declarativeComponent>
                        </af:panelGroupLayout>
                    </f:facet>
                </af:pageTemplate>
            </af:form>
        </af:document>
    </f:view>
</jsp:root>
```

Example: Page Definition File for Three Column Page Style

```
<?xml version='1.0' encoding='UTF-8'?>
<pageDefinition
Package="oracle.webcenter.siteresources.scopedMD.s8bba98ff_4cbb_40b8_beee_296c916a23e
d.pageStyle.gsr7ae8ef60_d5b9_4411_becb_11239bf4bb63" id="TemplateThreeColumnPageDef"
version="11.1.1.41.30" xmlns="http://xmlns.oracle.com/adfm/uimodel">
    <parameters>
        <parameter id="page_layout" value="gsr22bbc98b_834b_425d_8d48_c136d0956ec8"/>
        <parameter id="page_title" value="Three Column"/>
    </parameters>
    <executables>
        <page Refresh="ifNeeded" id="shellTemplateBinding"
viewId="#{WCAppContext.application.siteTemplatesManager.currentSiteTemplateViewId}"/>
    </executables>
    <bindings/>
    <permission
permissionClass="oracle.webcenter.page.model.security.CustomPagePermission"
target="ps_targetusage" xmlns="http://xmlns.oracle.com/adf/security">
        <privilege-map operation="administer" privilege="manage"/>
        <privilege-map operation="create" privilege="create"/>
        <privilege-map operation="delete" privilege="delete"/>
        <privilege-map operation="edit" privilege="update"/>
        <privilege-map operation="personalize" privilege="personalize"/>
        <privilege-map operation="view" privilege="view"/>
    </permission>
</pageDefinition>
```

6.4 Editing a Page Style

After creating the WebCenter Portal asset application for the page style, you can continue by adding content to the newly created page style file.

Note:

Oracle recommends that you copy an existing or built-in page style in WebCenter Portal and paste it into the page style JSPX file in JDeveloper. You can then use this page style as the starting point for your new page style and publish it back to WebCenter Portal once you've modified it. For more information about how to copy an asset in WebCenter Portal, see Copying an Asset in *Building Portals with Oracle WebCenter Portal*.

To edit a page style:

1. In the Application Navigator, right-click the page style's JSPX file, and choose **Open** (Figure 6-5).

Figure 6-5   Page Style JSPX File
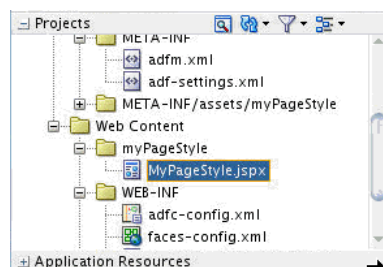
2. In the visual editor, make any changes to the layout of content of the page style in the same way as you would edit any page.

   For information about best practices for defining the content of the page style, see Best Practices for Creating Page Styles. For information about the files created for a page style asset application, see What You May Need to Know About Page Style Artifacts.

3. Save your changes.

4. Publish and test the page style as shown in Publishing a Page Style.

6.5 Publishing a Page Style

After creating a WebCenter Portal asset application and editing the JSPX file for the page style, the next step is to publish it and test it in WebCenter Portal. For instructions on how to publish an asset to WebCenter Portal as a shared asset, or to a specific portal as a portal asset, see Publishing WebCenter Portal Assets.

7

Developing Page Templates

This chapter describes how to use Oracle JDeveloper to create, edit, and publish page templates for use in WebCenter Portal.

Page templates provide the structure for common areas in portal pages. Using Oracle JDeveloper, you can design and develop page templates for portal pages displayed in WebCenter Portal.

This chapter includes the following topics:

- Introduction to Developing Page Templates
- Best Practices for Developing Page Templates
- Creating a Page Template
- Editing a Page Template
- Publishing a Page Template
- Page Template Tutorials and Examples

7.1 Introduction to Developing Page Templates

This section includes the following topics:

- Understanding Page Templates
- Understanding Page Template Structure
- Understanding Page Template Layout
- Understanding Page Template Layout Components

7.1.1 Understanding Page Templates

Page templates define how individual pages and groups of pages display on a user's screen, and help to ensure that the pages in a portal are consistent in structure and layout.

If you change a page template, then all pages that reference that template automatically inherit the changes.

> **Note:**
>
> For general information about page templates, see Using Page Templates in *Developing Web User Interfaces with Oracle ADF Faces*.

For tips and best practices for creating page templates for use in Oracle WebCenter Portal, see Best Practices for Developing Page Templates. Oracle recommends that you use Oracle JDeveloper to develop page templates for use in WebCenter Portal. You can also develop page templates in WebCenter Portal, but the editing capabilities are limited.

When fully developed, you can publish page templates directly to WebCenter Portal as a shared asset or to a specific portal for immediate use or for testing. For more information, see Publishing a Page Template.

7.1.2 Understanding Page Template Structure

Typical elements of a page template include:

- Header, content area (different in each page), and footer. The header and footer commonly include brand-specific elements. For example, a header usually includes a logo and possibly a slogan, and a footer usually includes contact and copyright information.

- Navigation. A page template can expose the navigation for a portal in many ways. For example, on a mobile device, portal navigation may be shown as a popup or slide in/out. On a desktop browser, portal navigation is typically shown along the top of the page, or down the side of the page.

- Branding elements. For example, a page template my include a company logo, slogan, or copyright message.

- Links and actions. For example, a log in/log out link, drop-down menus, or global links (such as links to send a mail message to the web administrator or to display a privacy statement).

- Conditional elements. For example, some elements on the page might differ depending on whether the user is public or authenticated or depending on the user's role and privileges.

Figure 7-1 shows a sample application based on a page template that illustrates these elements.

Figure 7-1   Sample Portal that Uses a Page Template
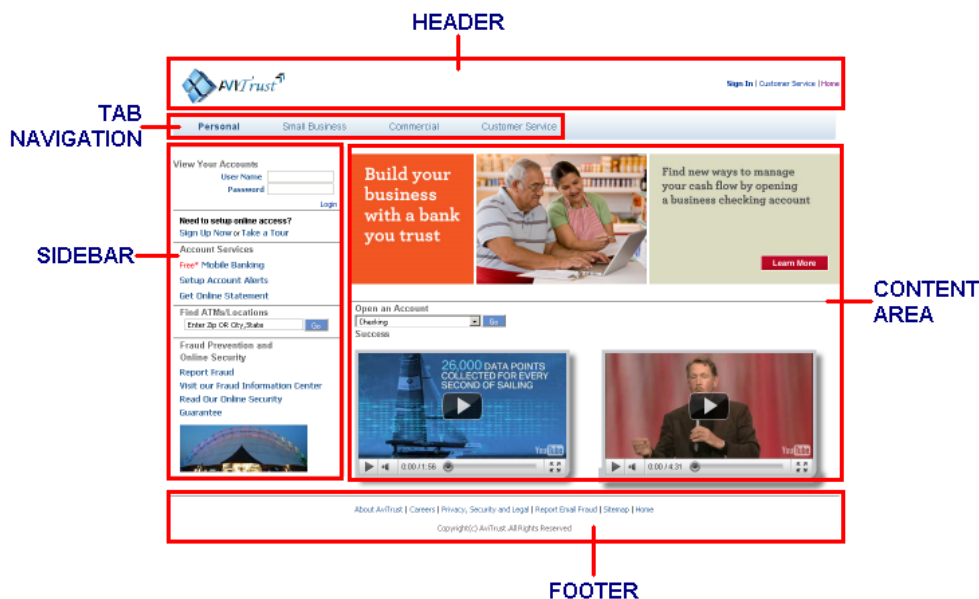
7.1.3 Understanding Page Template Layout

One of the most important aspects of page template design is the layout of components, both elements of the template and page content.

There are two basic strategies:

- A *flowing* layout is the most typical layout. Components are arranged side by side or one below the other, displayed using their natural size. When the content of the page extends beyond the size of the browser window, the browser displays scroll bars on the page.

- A *stretching* layout may be a suitable choice when your page content fills a large area, or you want the page content to grow and shrink depending on the size of the browser window. Components are stretched to occupy the available space on the page. For example, a stretching layout may be suitable when a page contains a table or graph that you want to fill up the whole content area, no matter what size it is. Another example is a page that contains an editing area, where you want the editor to be exactly as tall and wide as the space given to the content area. This layout has a region for the page content, and adds scroll bars to the region on the page when the content cannot be contained within the size of the browser window. In other words, individual components display scroll bars (which might mean that you have multiple scroll bars on one page).

   Stretching enables you to maximize the usage of the viewable area. You can use tabs, accordions, menus, and popups to expand your viewable area. When scroll bars are added to the page, the navigation area, page header, and page footer remain in view while the content area scrolls.

Because most web sites use a flowing layout, you will probably also want to use a flowing layout as it will likely feel more familiar to your users. However, stretching layouts are good for dashboards and applications that are rich in nature or when you want to mimic a desktop experience. You can also combine flowing and stretching layout on the same page.

Vertical Behavior

Depending on whether your layout is flowing or stretching, the vertical behavior of the page differs as follows:

- **Flowing layout:**

  - The header and/or footer might not always be visible

  - The height of the page is calculated based on the page contents

  - The content is never stretched vertically

  - The browser might display a scroll bar

- **Stretching layout:**

  - The header and footer are always visible

  - The height of the page is determined by the browser window

  - The content is stretched vertically

  - The content area might display a scroll bar

Horizontal Behavior

Depending on whether your layout is flowing or stretching, the horizontal behavior of the page differs as follows:

- **Flowing layout:**

  - If your page includes a side bar (for example, left-side navigation), the side bar might not always be visible

  - The width of the page is calculated based on the components

  - Some components might be stretched to fill up existing space

  - The browser might display a scroll bar

- **Stretching layout:**

  - If your page includes a side bar (for example, left-side navigation), the side bar is always visible

  - The width of the page is determined by the browser window

  - The content is stretched horizontally

  - The content might display a scroll bar

7.1.4 Understanding Page Template Layout Components

The underlying structure of a page template is provided by Oracle Application Development Framework (ADF) layout components.

After you decide on the appropriate layout to use for your page template (see Understanding Page Template Layout), you will use ADF layout components to create your page template. This is a complex task, and requires knowledge of the ADF components that will achieve the structure and layout you require, and best practices to employ (see Best Practices for Developing Page Templates).

> **Note:**
>
> - For details on what different layout components look like, see the ADF Faces Rich Client demo online tool:
>
>   `http://jdevadf.oracle.com/adf-richclient-demo/faces/visualDesigns/index.jspx`
>
>   In the demo tool, you can select **Page** or **Page Template** from the **View Source** menu to see what layout components and attributes are used to achieve the page structure.
>
> - Page Template Tutorials and Examples

Figure 7-2 and Figure 7-3 illustrate common ADF components used in a page template layout, showing the page template code followed by the generated layout:

- `af:panelGridLayout`—a versatile layout component that uses rows (`gridRow`) and cells (`gridCell`) to define a structured layout. This component is the preferred general layout component as it offers a small client side footprint while being very flexible in layout capabilities. It allows you to more naturally define areas of the page to match your desired layout in the form of rows and columns. With `panelGridLayout`, you can easily specify fixed or variable column widths (% or pixels), which cannot be done as easily with other layout components. See Figure 7-2.

- `af:panelGroupLayout`—a flowing series of components arranged horizontally, vertically, or in scrollable structures. Typically, `panelGroupLayout` is used with flowing layouts, and with flowing content inside a stretching layout. It provides vertical and horizontal scroll bars if the content does not fit into the available space. See Figure 7-2 and Figure 7-3.

- `af:panelSplitter`—a stretched box divided into two user-modifiable sections. See Figure 7-3.

Figure 7-2   Page Template Code and Generated Stretching Layout: Example 1

```
<af:panelGridLayout id="pgl1">
  <af:gridRow height="50px" id="gr2">
    <af:gridCell halign="stretch" valign="stretch" columnSpan="3" id="gc4">
      <af:panelGroupLayout id="pg1" />
    </af:gridCell>
  </af:gridRow>
  <af:gridRow height="100%" id="gr1">
    <af:gridCell width="20%" halign="stretch" valign="stretch" id="gc2">
      <af:panelGroupLayout id="pg2" />
    </af:gridCell>
    <af:gridCell width="70%" halign="stretch" valign="stretch" id="gc5">
      <af:panelGroupLayout id="pg3" />
    </af:gridCell>
    <af:gridCell width="10%" halign="stretch" valign="stretch" id="gc3">
      <af:panelGroupLayout id="pg4" />
    </af:gridCell>
  </af:gridRow>
  <af:gridRow height="50px" id="gr3">
    <af:gridCell halign="stretch" valign="stretch" columnSpan="3" id="gc1">
      <af:panelGroupLayout id="pg5" />
    </af:gridCell>
  </af:gridRow>
</af:panelGridLayout>
```

Figure 7-3   Page Template Code and Generated Stretching Layout: Example 2

```
<af:panelSplitter orientation="horizontal" splitterPosition="100" id="ps1">
  <f:facet name="first">
    <af:panelGroupLayout id="pg1" />
  </f:facet>
  <f:facet name="second">
    <af:panelGroupLayout id="pg1" />
  </f:facet>
</af:panelSplitter>
```

7.2 Best Practices for Developing Page Templates

Because page templates are present in every page of a portal, the design of your page templates should be carefully planned to optimize their performance and conform to best practices.

This section provides tips for developing page templates for WebCenter Portal (as a shared asset), or a specific portal.

Note:

Oracle recommends that you use JDeveloper to develop page templates for portals. You can also develop page templates in Oracle WebCenter Portal, but the editing capabilities are limited.

When fully developed, you can publish page templates to WebCenter Portal as a shared asset or to a specific portal for immediate use or for testing. Alternatively, you can export the page template to a file and upload the page template to WebCenter Portal later.

For more information, see Creating a Page Template, Editing a Page Template, and Publishing a Page Template.

Table 7-1 provides a quick reference summary of considerations and guidance for achieving the best results out of your page templates.

Table 7-1   Best Practices Summary for Page Templates

Table 7-1   (Cont.) Best Practices Summary for Page Templates

Considerations	Best Practices
Performance	While all of this section is aimed at improving the performance of your page templates, there are a few general tips to keep in mind as overall best practices:
	• Minimize the use of embedded task flows. For examples, refer to the latest out-of-the-box page templates included with WebCenter Portal, described in About Built-in Page Templates in WebCenter Portal in *Building Portals with Oracle WebCenter Portal*.
	• Minimize the use of ADF layout components, using `panelGridLayout` to implement your page template structure. The fewer ADF layout components, the easier it is to apply skins.
	• Avoid using images for decorative purposes (such as rounded corners). Consider using CSS instead of images.
	• Defer loading of ADF components, such as menus and dialogs, where possible. This can be controlled through ADF such as `contentDelivery`, `childCreation`, and so on.
	• Avoid elements that require extra time in rendering page templates and try to substitute for elements that require less processing time.
Layout	One of the most important aspects of page template design is how to lay out components, both elements of the template and page content. Page templates can have a flowing layout or a stretching layout. For more details about these two strategies, see Understanding Page Template Layout.
	As a page template developer, you can control whether the content facet is in a stretching or flowing region of the page. Page content must be created taking the layout strategy into consideration.
	Once you decide on the best strategy for your page templates, see the following sections for tips on creating the layout you choose:
	• Best Practices for Creating Stretching Layouts
	• Best Practices for Creating Flowing Layouts

Table 7-1   (Cont.) Best Practices Summary for Page Templates

Considerations	Best Practices
Navigation	A page template can expose the navigation for a portal in many ways. For example, in a desktop browser, portal navigation is typically shown along the top of the page, or down the side of the page:
	• A *top navigation* page template exposes the portal navigation in header area. Top navigation makes effective use of the horizontal space on the page, and is recommended when there are seven or fewer top level pages in the portal navigation. This page template design generally has a header, page and footer sections, and is an ideal starting point for sites that require a flowing layout.
	• A *side navigation* page template exposes the portal navigation in a sidebar on the left side of the page. The vertical nature of side navigation allows for a more lengthy list of navigation items, and is recommended when there are more than seven top level pages in the portal navigation. Choose a side navigation template for more complex navigation models.
	On different devices, navigation can be exposed differently. For example, in page templates optimized for mobile devices, navigation can be provided as either a popup or slide in/out.
Skin	Each page template works together with a skin to determine the overall look and feel of the pages in your portal. While the page template controls the structure of components on the page, the skin controls the visual appearance of those components, such as the colors, fonts, and other aspects such as the position, height, and width of components on the page.
	Each page template can define a preferred skin to identify the skin that works best with that page template. When the page template is selected as the default page template for a portal or as the system default, the default skin automatically updates to the page template's preferred skin.
	For more information, see Developing Skins.
Runtime behavior	If you develop a page template in JDeveloper that you plan to allow authorized users to edit in Composer at runtime, follow the tips provided in Best Practices for Developing Page Templates That Can Be Edited at Runtime.
Components	For details on what different ADF layout components look like, see the ADF Faces Rich Client demo online tool (`http://jdevadf.oracle.com/adf-richclient-demo/faces/visualDesigns/index.jspx`). Select **Page** or **Page Template** from the **View Source** menu for a component to see what tags and attributes are used as well as what the component structure looks like for the page.
	Because a page template layout can be changed, create pages and design custom components that display properly in flowing and stretching context.
	For tips on customizing the appearance of components, see Best Practices for Customizing the Appearance of Components.

Table 7-1   (Cont.) Best Practices Summary for Page Templates

Considerations	Best Practices
Scrolling	Add scrollbars only around flowing island content. For tips on implementing scrollbars, see Best Practices for Defining Scrolling in a Page Template.
Margins, borders, and padding	Due to browser CSS Box Model Rules, defining margins, borders and padding on components might be complex. For tips on resolving the complexities of defining margins, borders and padding on components, see Best Practices for Defining Margins, Borders, and Padding.
Attributes	Consider attributes that can be set in your page templates or pages that use the page templates. For example, `showFooter` specifies whether or not to show the page footer. A page template without attributes is syntactically correct. However, page template attributes are useful when you want to use one page template for several pages where the template should render slightly differently. Information about using attributes in your page templates is provided throughout this chapter, including Creating a Page Template.
Links	To add links to page templates, copy components provided in the out-of-the-box page templates to include links navigation, menus, breadcrumbs, buttons, and images. `go` components such as `goButton` and `goLink` have little to no client side footprint since they do not carry client side functionality. They are also preferred over command components (`commandButton`, `commandLink`) for general navigation in portals because they enable URLs that are optimized for search engines.
Internationalization	To create page templates with internationalization techniques in mind, see Guidelines for Building Multilanguage Portals for information about recommended practices such as storing static text in resource bundle files.
Naming page templates (display name)	The display name is exposed to users when creating a new page. For this reason, you should make the page template name something that helps users quickly identify which type of pages the template should be used for.

This section includes the following topics:

- Best Practices for Creating Stretching Layouts

- Best Practices for Creating Flowing Layouts

- Best Practices for Developing Page Templates That Can Be Edited at Runtime

- Best Practices for Customizing the Appearance of Components

- Best Practices for Defining Scrolling in a Page Template

- Best Practices for Defining Margins, Borders, and Padding

7.2.1 Best Practices for Creating Stretching Layouts

If your page template is best suited to a stretching layout, follow these tips as you develop the layout.

For more information about different layouts, see Understanding Page Template Layout.

- Build the outer structure with containers that can be stretched and can stretch their children. Inside your `document` component, use containers such as `panelGridLayout` (Figure 7-2) with rows (`gridRow`) and cells (`gridCell`), and `panelSplitter` (Figure 7-3).

> **Note:**
>
> Each layout or panel component's tag documentation identifies whether it is stretchable and how to achieve it in its "Geometry Management" documentation. Some components have attributes to determine whether children will be stretched or not. For example: `document` has a `maximized` attribute, `showDetailItem` has a `stretchChildren` attribute.

- Create flowing islands. Inside of the stretchable outer structure, create islands of flowing (non-stretched) components. To make this transition from stretching to flowing, use `panelGroupLayout` with `layout="scroll"` or `layout="vertical"` since it supports being stretched but will not stretch its children.

- On stretching components, set `dimensionsFrom="auto"` so that the stretching component (for example, `panelStretchLayout`) will only try to stretch its child if it itself is being stretched. If it is not being stretched, then it will flow (not stretch) its children.

- Do not stretch something vertically (by using a height with a percent value) when inside a flowing container.

- Do not use the `position` CSS property in an `inlineStyle`. Doing so will impede the ability to override this property with a style specified in the skin.

> **Note:**
>
> The following components are just some of the components that cannot be reliably stretched:
>
> - Most input components
>
> - `panelBorderLayout`
>
> - `panelFormLayout`
>
> - `panelGroupLayout` (with `layout="default"`)
>
> - `panelGroupLayout` (with `layout="horizontal"`)
>
> - `panelHeader` (with `type="flow"`)
>
> - `panelLabelAndMessage`
>
> - `panelList`
>
> - `panelGrid`

7.2.2 Best Practices for Creating Flowing Layouts

If your page template is best suited to a flowing layout, follow these tips as you develop the layout.

For more information about different layouts, see Understanding Page Template Layout.

- Use `panelGridLayout` with rows (`gridRow`) and cells (`gridCell`) to define a structured layout that will flow.

- To avoid multiple scroll bars, do not nest scrolling `panelGroupLayout` components, instead use `layout="vertical"`.

- Most stretchable ADF components also work in flowing context with `dimensionsFrom="auto"`.

- To stretch a component horizontally, use `styleClass="AFStretchWidth"` (instead of `inlineStyle="width:100.0%"`).

Working with customizable components:

- In `panelCustomizable`, use `layout="auto"` to detect whether to stretch its children.

- To support flowing and stretching layouts, use `showDetailFrame` with `stretchChildren="auto"`.

7.2.3 Best Practices for Developing Page Templates That Can Be Edited at Runtime

To create a page template at design time (in Oracle JDeveloper) that is suited to being editable at runtime (in Oracle WebCenter Portal), follow these tips.

- Add components from the Composer group in the Component Palette.

- Do not add `pageCustomizable` to a page template.

- Add at least one `panelCustomizable` component, which provides a container with horizontal or vertical layout that holds other components.

- Inside a `panelCustomizable` component, add ADF components (such as `outputText`, `richTextEditor`, or `goLink`) surrounded by a `showDetailFrame`, which provides a chrome for a component that enables you to add actions like show, hide, and move. You can edit the frame or the embedded component properties.

> **Caution:**
>
> Use this technique sparingly, as `showDetailFrame` components impact processing time. If the end user is not expected to move components around, do not wrap them in a `showDetailFrame`.

Many of the components available to add to a page at design time are also available at runtime in the resource catalog. Table 7-2 maps design time components to runtime components.

Table 7-2   Component Mapping: Design time to Runtime

Design Time (JDeveloper)	Runtime (WebCenter Portal)
`panelCustomizable`	`Box`
`outputText`	`HTML Markup`
`goLink`	`Hyperlink`
`goImageLink`	`Image`
`showDetailFrame`	`Movable Box`
`richTextEditor`	`Text`
`inlineFrame`	`Web Page`

7.2.4 Best Practices for Customizing the Appearance of Components

To customize the appearance of components, follow these tips.

- For custom styling, use Cascading Style Sheets (CSS), which is easy to maintain and can be changed without changing the page template source. For example, make the background color of a page blue using the CSS code `background-color: blue`.

- Use a custom skin for consistently modified appearances if the existing skin doesn't provide all that you need.

- For instance-specific alternative styling, use the `styleClass` attribute. Keep the corresponding style definitions in an easy-to-maintain location such as in a custom skin (recommended) or in a style provided by the `af:resource` tag.

- As a last resort, use component attributes such as `inlineStyle`, `contentStyle`, and `labelStyle`. These are harder to maintain as they are scattered throughout

components rather than collected in a single style sheet, contribute more to the page's raw HTML size, and may not even be needed if one or more of the above mechanisms are used. Styles are directly processed by the web browser, which gives you a great deal of power but at the cost of being error-prone.

- Browsers do not support all styles on all elements and certain combinations of styles produce non-obvious results. Here is some guidance on style configurations to avoid:

  - An `inlineStyle` with a `height` value with `%` units

  - An `inlineStyle` with a `width` value between `90%` and `100%` (use `styleClass="AFStretchWidth"` or `styleClass="AFAuxiliaryStretchWidth"` instead)

  - An `inlineStyle` with `height`, `top`, and `bottom` values

  - An `inlineStyle` with `width`, `left`, and `right` values

  - An `inlineStyle` with a `position` value

  - In a child being stretched by a parent component, an `inlineStyle` with `width` or `height` values

7.2.5 Best Practices for Defining Scrolling in a Page Template

To define scrolling in a page template, follow these tips.

- Try to avoid the need for end users to scroll horizontally by designing the page content to occupy the available screen size.

- Add scrollbars only around flowing island content. The recommended transition component for switching from a stretching outer frame into a flowing island is the `panelGroupLayout` with `layout="scroll"`. If the contents of this `panelGroupLayout` cannot fit in the space allocated, the browser will determine whether scrollbars are needed and will add them automatically.

- Do not nest scrolling `panelGroupLayout` components because this will make the user see multiple scrollbars. Also, this should only be used at transitions from stretching to flowing areas and since you should not have stretching areas inside of flowing areas, you would generally never end up with nested scrollbars. It is best to minimize the number of areas that users must scroll in order to see what they are looking to find. Take time to consider what scrolling users will need. In cases where undesired scrollbars exist, you may want to change the `layout` attribute of the `panelGroupLayout` to `vertical`.

7.2.6 Best Practices for Defining Margins, Borders, and Padding

Due to browser CSS Box Model Rules, defining margins, borders and padding on components can be complex. In many cases, to apply these kinds of styles, you need to use multiple components together.

- In a scrolling area, you can add an extra `panelGroupLayout` with `layout="vertical"` with the padding defined on it, inside of the outer `panelGroupLayout` with `layout="scroll"`.

- In a stretching area, you may need to wrap a `panelGroupLayout` component inside a `panelGridLayout` with spacers inside `gridCell` components. See Figure 7-2.

Note:

Refer to the Navigation-Master-Detail, Tiled Flowing, and Tiled Stretching layout pattern examples for various mechanisms to apply padding:

http://jdevadf.oracle.com/adf-richclient-demo/faces/feature/layoutBasicTest.jspx

---

7.3 Creating a Page Template

This section contains the following subsections:

- About Creating Page Templates

- How to Create a Page Template

- What You May Need to Know About Page Template Artifacts

7.3.1 About Creating Page Templates

Before you can design your page template, you must first create a WebCenter Portal Asset application for the page template.

Before you create a page template, be sure to read through Best Practices for Developing Page Templates.

Note:

Oracle recommends that you use JDeveloper to develop page templates for Oracle WebCenter Portal. You can also develop page templates in WebCenter Portal, but the editing capabilities are limited.

When fully developed, you can publish a page template directly to WebCenter Portal as a shared asset or to a specific portal for immediate use or for testing. Alternatively, you can export the page template to a file and upload the page template to WebCenter Portal later

For more information, see Editing a Page Template and Publishing a Page Template.

For examples of features that you can include in your own page templates, refer to the built-in page templates provided with WebCenter Portal, specifically the latest responsive page templates (Mosaic and Unicorn), and other page templates that minimize the use of task flows and include the performance-optimizing `panelGridLayout` component, which uses rows (`gridRow`) and cells (`gridCell`) to define a structured layout. See About Built-in Page Templates in WebCenter Portal in *Building Portals with Oracle WebCenter Portal*.

You can start by copying and pasting one of one of the built-in templates into JDeveloper, or you can build an entirely new page template from scratch. Even if you do not intend to base your page templates on the built-in ones at all, it is still worth studying them for ideas. For example, the built-in page templates include a login form,

and a good example of navigation visualization, which you can modify according to your requirements.

7.3.2 How to Create a Page Template

This section describes how to create a WebCenter Portal asset application for a new page template.

To create a custom page template:

1. Create a WebCenter Portal asset application for the asset, selecting **Page Template** as the **Asset Type**.

   Figure 7-4   Page Template Asset Type
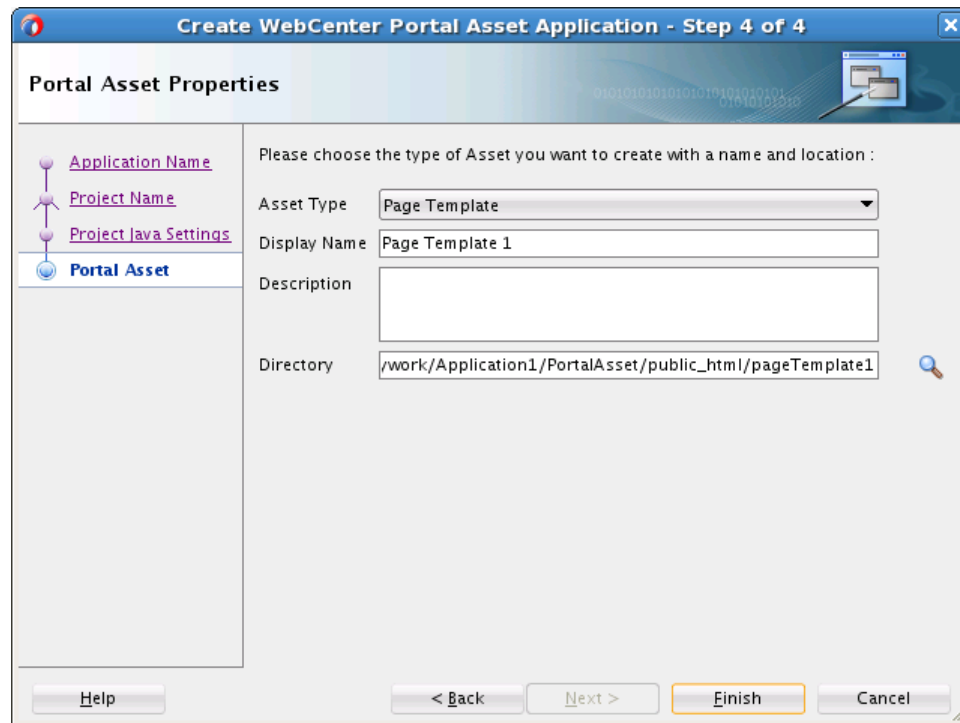
   See Creating a WebCenter Portal Asset Application for more information about creating WebCenter Portal asset applications.

   A JSPX file is created for the page template.

Figure 7-5   Page Template JSPX File
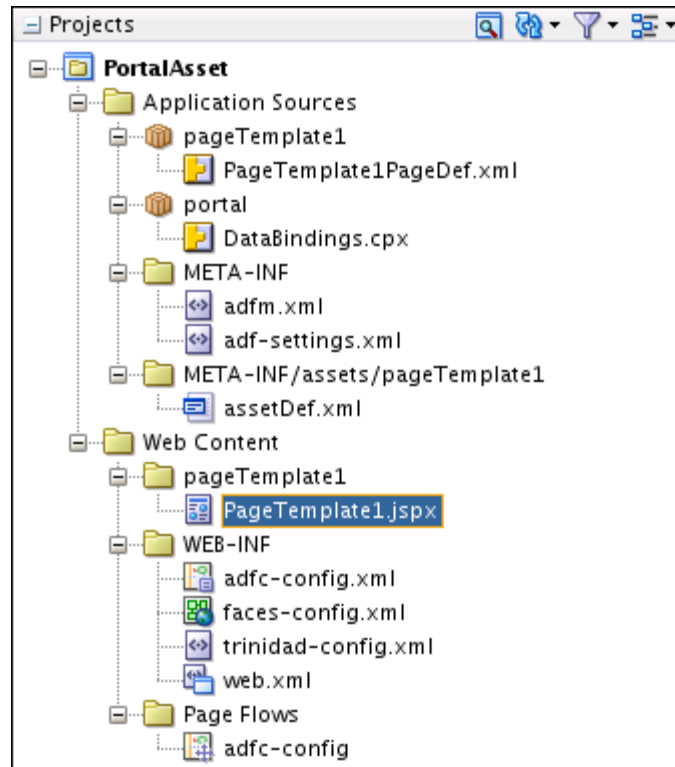
For more information about the artifacts created when you create an asset application for a new page template, see What You May Need to Know About Page Template Artifacts.

2. Continue by editing the contents of the JSPX file as described in Editing a Page Template.

7.3.3 What You May Need to Know About Page Template Artifacts

Creating a page template asset application produces a default template, with the following artifacts:

- A JSPX file containing the page elements for the page template (for example, `PageTemplate1.jspx`)

- The corresponding page definition file for the page template (for example, `PageTemplate1PageDef.xml`)

Both of these files appear in the Application Navigator as shown in Figure 7-6.

Figure 7-6   Page Template Asset Application Artifacts
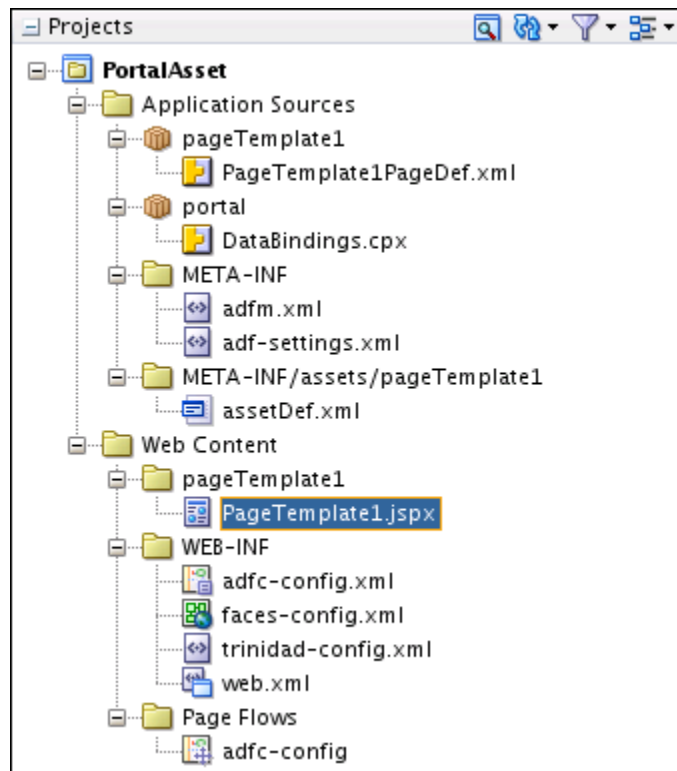
7.4 Editing a Page Template

After creating a WebCenter Portal asset application and initial page template, continue by adding or modifying elements.

To edit a page template:

1. In the Application Navigator, right-click the JSPX file of the page template that you want to edit, and choose **Open**.

2. In the editor, make required changes to the layout and content of the page template by dragging components from the Component Palette.

   For more information about how to build your page template, see Introduction to Developing Page Templates and Best Practices for Developing Page Templates.

   The built-in page templates provide useful examples of page templates, and see also Page Template Tutorials and Examples.

   For general information about JSF page templates, see Using Page Templates in *Developing Web User Interfaces with Oracle ADF Faces*.

3. Save your changes.

7.5 Adding a Floating Toolbar to a Page Template

When users with the `Contribute Page Content` permission view a page, they will see a floating toolbar with a Contribute option when the page template includes the toolbar.

To add the floating toolbar to a custom page template:

1. Include the following xml namespace declaration (if not already present):

   ```
   xmlns:wcdc="http://xmlns.oracle.com/webcenter/spaces/taglib"
   ```

2. Within the xml namespace declaration, include the following reference to the floating toolbar:

   ```
   <wcdc:portalToolbar id="ptbdc"/>
   ```

   Figure 7-7 shows how the floating toolbar with the **Contribute** option will appear when the page template includes the toolbar.

   Figure 7-7   The Contribute Button in the Floating Toolbar
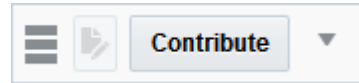
   For more information, see About Content Contribution and Publishing in *Building Portals with Oracle WebCenter Portal*.

7.6 Publishing a Page Template

After creating a page template and editing its JSPX file, the next step is to publish and test the template in WebCenter Portal.

For instructions on how to publish a page template as a shared asset, or to a specific portal as a portal asset, see Publishing WebCenter Portal Assets.

7.7 Page Template Tutorials and Examples

The supplementary tutorials and examples listed in this section provide additional information about page templates.

- *Dissecting a Page Template*. Analyzes the details of a simple page template, including recommended practices.

  ```
  http://www.oracle.com/technetwork/middleware/webcenter/
  portal/learnmore/dissectingapagetemplate-1926909.pdf
  ```

- *Oracle WebCenter Portal Online Training: Creating and Using Page Templates in Oracle WebCenter Portal Applications*. Provides a recorded presentation and slides to create and use page templates for a portal in an earlier release.

  ```
  http://www.oracle.com/technetwork/middleware/webcenter/
  portal/learnmore/pagetemplates-1438595.pdf
  ```

- *Optimizing WebCenter Portal Mobile Delivery*. Identifies and analyzes some common WebCenter Portal performance bottlenecks related to page weight and describes a generic approach that can streamline a portal while improving the performance and response times. Of particular interest in the context of developing page templates is the section "Page Design and Component Choices." A sample application is available for download.

  ```
  http://www.ateam-oracle.com/webcenter-mobile-delivery/
  ```

- *Working with Links in WebCenter Application*. Describes considerations for using links. Pertinent to page templates are links in menus, breadcrumbs, buttons, and images.

```
http://www.ateam-oracle.com/working-with-links-in-webcenter-
application/
```

8

Developing Skins

This chapter describes how to use Oracle JDeveloper to create and edit skins for use in WebCenter Portal.

This chapter includes the following topics:

- Introduction to Developing Skins

- Best Practices for Developing Skins

- Creating a Skin

- Editing a Skin

- Publishing a Skin

- Conditionally Changing Skins for Users

8.1 Introduction to Developing Skins

Skins help you define the colors, fonts, images, and some dimensional details like the height and width of your application components to represent your company's preferred look and feel.

This section includes the following topics:

- About Skins

- About Runtime Management of Skins

8.1.1 About Skins

Skins are based on the Cascading Style Sheet (CSS) specification. A skin is a CSS file containing various skin selectors that define the styles of your application components. You can adjust the look and feel of any component by changing its style-related properties. Use of a skin in an application helps you to avoid specifying styles for each component individually or inserting a style sheet on each page. Every component automatically uses the styles defined in the skin. Skins help you to change an application's appearance without changing the portal pages themselves.

8.1.2 About Runtime Management of Skins

WebCenter Portal supports the runtime administration of skins to help users continue developing a portal even after it has been deployed. With runtime administration, authorized users can manage an application's skins, and create new ones, in a browser-based environment, with no requirement to install or understand JDeveloper. For more information, see Working with Skins in *Building Portals with Oracle WebCenter Portal*.

8.2 Best Practices for Developing Skins

You can approach creating or modifying a skin to be used in WebCenter Portal as two somewhat separate tasks. The first task is to apply basic styling elements to the larger elements on the page, such as the background and the center of the page, to provide the look and feel for your corporate brand. This is probably the first thing you will want to do. Secondarily, you may also want to apply styling to some of the smaller page elements to fine tune the look and feel.

In many cases a hybrid approach may also work well. Taking the larger elements (page background, main portion of the body, and so on) and applying traditional CSS styling with those, but then getting specific using ADF skinning to generate the overall appearance for WebCenter Portal.

When using CSS-based styling and other techniques in addition to ADF styling, it is often helpful to store the styling assets outside of WebCenter Portal. To do this you can use Content Server to manage all of the unstructured assets for WebCenter Portal. They can include things like CSS and images that you want manage within your environment and provides revision control and workflow. This best practice lets design teams access and work with WebCenter Portal without involving the development team for each and every change.

8.3 Creating a Skin

By default, WebCenter Portal uses the `portal` skin, which is defined in the `portal-skin.css` file. After creating a WebCenter Portal asset application, you can easily copy and paste this skin into JDeveloper to alter it to meet your specific requirements, or create a new skin from scratch. For more information about how to copy an asset in WebCenter Portal, see Copying an Asset in *Building Portals with Oracle WebCenter Portal*.

This section includes the following topics:

- How to Create a Skin

- What You May Need to Know About Skin Artifacts

8.3.1 How to Create a Skin

This section describes how to create a WebCenter Portal asset application for a new skin. Once the asset application with its associated artifacts has been created, you can continue by modifying the skin's CSS file as described in Editing a Skin.

> **Note:**
>
> Oracle recommends that you store asset-related artifacts, such as images and icons, on Content Server, and that you create a folder structure on your content server specifically for asset artifacts so that content is easy to identify and move, if required. This also allows other users to alter those types of content without involving a developer.

To create a WebCenter Portal asset application for a skin:

1. Create an asset application for the asset specifying `Skin` as the **Asset Type**.

See Creating a WebCenter Portal Asset Application for more information about creating WebCenter Portal asset applications. For information about the artifacts that are created when you create an asset application for a skin, see What You May Need to Know About Skin Artifacts.

2. In the Application Navigator, right-click the newly created skin's CSS file, and choose **Open** (Figure 8-1).

Figure 8-1   Skin CSS File
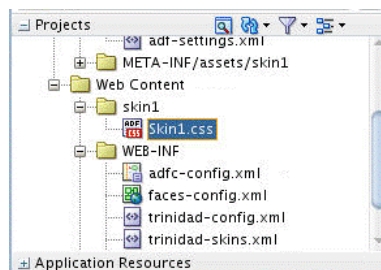
3. Continue by editing the skin as described in Editing a Skin.

8.3.2 What You May Need to Know About Skin Artifacts

Creating a skin asset application produces a default skin, with the following artifacts:

- the CSS skin selector file (for example, `Skin1.css`)

- `trinidad-skins.xml`, the skin definition file

Both of these files appear in the Navigation bar as shown in Figure 8-2.

Figure 8-2   Skin Asset Application Artifacts
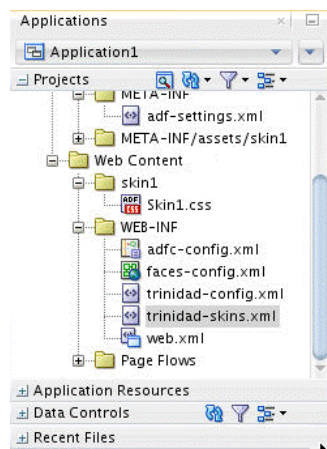
8.4 Editing a Skin

You can edit a skin after its initial creation by editing the CSS file to edit or add style selectors.

> **Note:**
>
> Oracle recommends that you copy an existing or built-in skin in WebCenter Portal and paste it into the page style CSS file in JDeveloper. You can then use this skin as the starting point for your new skin and publish it back to WebCenter Portal once you've modified it. For more information about how to copy an asset in WebCenter Portal, see Copying an Asset in *Building Portals with Oracle WebCenter Portal*.

To edit a skin:

1. In the Application Navigator, right-click the skin's CSS file and choose **Open** (Figure 8-3).

   Figure 8-3   Skin CSS File
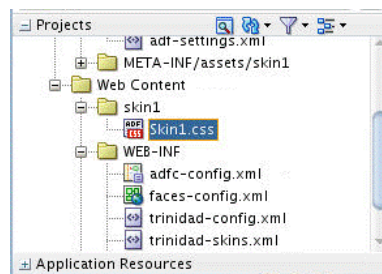
2. If you are using an existing WebCenter Portal skin (that is, other than the one created by default when you create the asset application) as a starting point (for example, the `portal` skin), copy and paste it into the CSS file (see Copying an Asset in *Building Portals with Oracle WebCenter Portal*).

3. In the source code for the skin, define the required ADF Faces skin selectors for the components in your application. For example, to define the font family for your application content, you can use the `.AFDefaultFont:alias` skin selector as shown here:

   ```
   .AFDefaultFontFamily:alias {
   font-family: Tahoma, Verdana, Helvetica, sans-serif;
   }
   ```

   For information about:

   - ADF Faces skin selectors in general, refer to Customizing the Appearance Using Styles and Skins in *Developing Web User Interfaces with Oracle ADF Faces*. Also refer to the Oracle JDeveloper online help for information about the selectors that you can use in a skin. These are documented in the "Skin Selectors for Fusion's ADF Faces Components" and "Skin Selectors for Fusion's Data Visualization Tools Components" topics in the online help.

   - Defining ADF Faces component style selectors, see Enabling End Users to Change an Application's ADF Skin and Changing the Style Properties of a Component in *Developing Web User Interfaces with Oracle ADF Faces*.

4. Save the CSS file.

5. Publish and test the skin as shown in Publishing a Skin.

8.5 Publishing a Skin

After creating a WebCenter Portal asset application and editing the CSS file for the skin, the next step is to publish it and test it in WebCenter Portal. For instructions on how to publish an asset to WebCenter Portal as a shared asset, or to a specific portal as a portal asset, see Publishing WebCenter Portal Assets.

8.6 Conditionally Changing Skins for Users

You can assign different skins per user, page, application, and so on, without impacting the actual application logic. To conditionally set a skin, you use the `<skin-family>` entry in the `trinidad-config.xml` file.

You can use EL expressions that can be evaluated dynamically to determine the skin to display. For example, if you want to use the German skin when the user's browser is set to the German locale, and to use the English skin otherwise, use the following `<skin-family>` entry in the `trinidad-config.xml` file:

```
<skin-family>#{facesContext.viewRoot.locale.language=='de' ?
'german' : 'english'}</skin-family>
```

For more information, see Enabling End Users to Change an Application's ADF Skin in *Developing Web User Interfaces with Oracle ADF Faces*.

> **Note:**
>
> The default value of `<skin-family>` is the name of the skin asset you created with the asset application (for example, `Skin1`) minus any spaces or special characters. If you change the default value, your application users cannot set skins at runtime in the Resource Manager.

You can also use the `SkinSetting` API to set the skin for a user conditionally. For more information, refer to *Oracle WebCenter Portal Java API Reference*.

9

Developing Visualization Templates

This chapter describes how to use Oracle JDeveloper to create and edit templates for visualizing data in WebCenter Portal.

This chapter includes the following topics:

- About Visualization Templates

- Creating a Visualization Template

- Editing a Visualization Template

- Publishing a Visualization Template

- What Happens at Runtime

9.1 About Visualization Templates

Introduced in WebCenter Portal 12*c* (12.2.1) is the ability to retrieve data from a REST or SQL data source using a *business object*, simplifying the complexities of application integration. The retrieved data can be rendered on a portal page in a *data visualization* using a *visualization template*. For example, the data can be presented in one of the built-in visualization templates, or you can build a custom visualization template in JDeveloper if none of the built-in templates meet the needs of your organization.

> **See Also:** For information about the built-in visualization templates, see About Visualization Templates in *Building Portals with Oracle WebCenter Portal*

A visualization template is a JSFF file (JSP page fragment) that defines how the data retrieved by a business object is presented on a portal page. Unlike built-in visualization templates, which are generic presentation styles that can be used in multiple scenarios by binding to different business objects, a custom visualization template has little to no reusability as each custom template is designed to be used with a specific business object. Hence, a custom visualization template is also known as a *bound visualization template*.

When an application specialist or portal manager adds a data visualization component to a portal page in WebCenter Portal (as described in Adding a Data Visualization to a Page in *Building Portals with Oracle WebCenter Portal*), they configure the data visualization using the Define Data Visualization wizard to select the data source and visualization template, which is bound to the business object. The Options page of the wizard dynamically displays the selected template's attributes, either those of a built-in template or the placeholder EL specified in a custom template. For a custom visualization template, placeholder EL is bound to each business object attribute, which replaces the EL at runtime. For more information, see Configuring a Data Visualization in *Building Portals with Oracle WebCenter Portal*

9.2 Creating a Visualization Template

This section describes how to create a new custom visualization template in the following topics:

- How to Create a Visualization Template

- What You May Need to Know About Visualization Template Artifacts

9.2.1 How to Create a Visualization Template

This section describes how to create a WebCenter Portal asset application for a new visualization template. Once the asset application with its associated artifacts has been created, you can continue by modifying the visualization template as described in Editing a Visualization Template.

Note:

Oracle recommends that you store asset-related artifacts, such as images and icons, on your content server and that you create a folder structure on your content server specifically for asset artifacts so that content is easy to identify and move, if required.

To create a custom visualization template:

1. Create an asset application for the asset, specifying **Visualization Template** as the **Asset Type** (Figure 9-1).

   Figure 9-1   Visualization Template Asset Type

   See Creating a WebCenter Portal Asset Application for more information about creating WebCenter Portal asset applications. For information about the artifacts that are created when you create an asset application for a visualization template, see What You May Need to Know About Visualization Template Artifacts.

2. In the Application Navigator, right-click the newly created visualization template's page fragment file (Figure 9-2), choose **Open**, then switch to **Source** view.

Figure 9-2   Visualization Template Page Fragment File
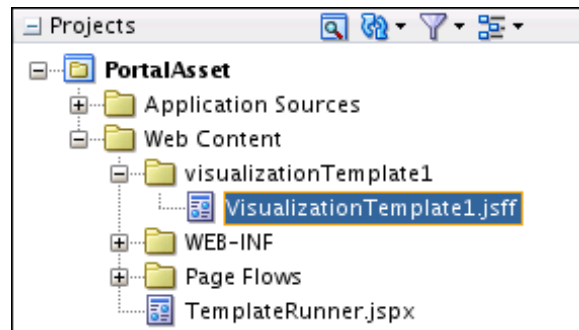
3. Continue by editing the contents of the page fragment as described in Editing a Visualization Template.

9.2.2 What You May Need to Know About Visualization Template Artifacts

Creating a visualization template asset application produces a default template, with the following artifact:

- A view fragment file (for example, `VisualizationTemplate1.jsff`)

This file appears in the Application Navigator as shown in Figure 9-3.

Figure 9-3   Visualization Template Asset Application Artifacts
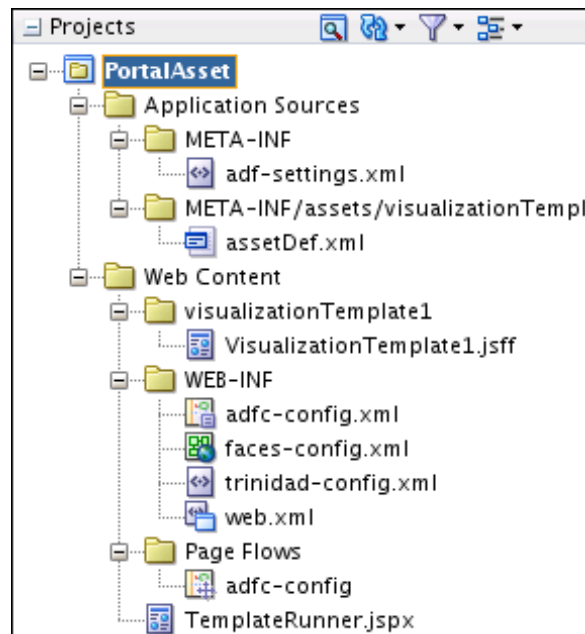
To expose a visualization template for use when creating data visualizations at runtime, you must consider the following requirements:

- Create *only one* view fragment for a task flow that will be used as a visualization template. The view fragment JSFF file must be created in the same location as the task flow definition XML file.

- If the visualization template references code (for example, using an EL value), then that code must be present in the deployed application.

9.3 Editing a Visualization Template

After creating the initial framework of a visualization template (see How to Create a Visualization Template), you must edit it to define the template.

> **Note:** If changes are made to a custom visualization template that is already selected in a data visualization, you will need to configure a new data visualization and reselect the revised visualization template in order to reflect the changes in the data presented on the page.

To edit a visualization template:

1. In the Application Navigator, right-click the visualization template JSFF file, and choose **Open**.

2. In the editor, make required changes to the code of the visualization template.

> **Notes:**
>
> - Use `af:subform` around the `root-element` to allow any user input required in the visualization template to be submitted, along with any input required by other components on the page. Without `af:subform`, user input in a custom visualization template may prevent the submission of other input, such as in another form on the page.
>
> - To add placeholder EL to bind to business object attributes at runtime, see Placeholder EL Binding in Visualization Templates.

3. Save your changes.

9.3.1 Placeholder EL Binding in Visualization Templates

Custom visualization templates contain placeholder Expression Language (EL) that is bound to business object attributes at runtime. Using placeholder EL allows you to add a binding hint to the custom visualization template so that users creating the data visualization can select the placeholder EL in the Define Data Visualization wizard to bind to a business object attribute, parameter, or method at runtime.

The following ADF components support binding placeholder EL to a business object:

- `af:form`

- `af:image`

- `af:inputDate`

- `af:inputText`

- `af:iterator`

- `af:link`

- `af:listView`

- `af:outputFormatted`

- `af:outputText`

- `af:selectBooleanCheckbox`

- `af:selectOneChoice`

- `af:selectOneRadio`

- `af:selectBooleanRadio`

- `af:table`

Within these components, the syntax for binding custom visualization template placeholder EL to a business object is as follows:

Binding Template Placeholder EL to a Business Object Attribute

Placeholder EL syntax:

`#{owcp.appsint.data('`*`displayName`*`','`*`parentInformation`*`','`*`defaultVa lue`*`','`*`attribute binding hint`*`','')}` where:

- `owcp.appsint.data` is a keyword to indicate that this placeholder is to be bound to a business object *attribute*. Only placeholder EL that contains`owcp.appsint.data` will be bound to a business object attribute at runtime.

- *`displayName`* is the placeholder display name of the attribute that will be shown to the user in the Define Data Visualization wizard Options page.

- *`parentInformation`* is the value of the `id` attribute of the parent (iterator), if the attribute's component depends on a parent (iterator). For example, `af:outputText` inside `af:column` has *`parentInformation`* as the `id` of the containing `af:table` component:

```
<af:table var="row" id="t1" value="#{owcp.appsint.data('DOCUMENT
Details','','None','','')}" >
   <af:column headerText="Name" id="c1" >
      <af:outputText value="#{owcp.appsint.data('Name','t1','Name','','')}"
id="ot1"/>
   </af:column>
 </af:table>
```

- *`defaultValue`* is the default value for the component to render it in JDeveloper for testing purposes.

- *`attribute binding hint`* is the xpath (*`operationName`*`.`*`accessor1`*`.`*`accessor2`*`.`*`attribute`*) for the attribute, so that the exposed placeholder in the EL is bound to the business object attribute by default.

  For a SQL data source, the binding hint is `GET.ResultSet.`*`column_name`*. For example, for a SQL data source that retrieves `NAME` and `ID` values (`select NAME, ID from `*`some_table`*), the binding hint for `NAME` is `GET.ResultSet.NAME` and for `ID` is `GET.ResultSet.ID`.

  Example JSFF to list employees for a SQL data source:

– Using table:

```
<af:table var="row" id="t1" value="#{owcp.appsint.data('Employee
List','','None','GET.ResultSet','')}" >
   <af::column headerText="Name" id="c1" >
      <af:outputText
value="#{owcp.appsint.data('Name','t1','Name','GET.ResultSet.NAME','')}"
id="ot1"/>
   </af:column>
</af:table>
```

– Using iterator:

```
<af:iterator id="i1" value="#{owcp.appsint.data('Employee
List','','None','GET.ResultSet','')}" var="row">
  <af:panelGroupLayout id="pgl2" layout="vertical">
    <af:outputText
value="#{owcp.appsint.data('Name','i1','John','GET.ResultSet.NAME','')}"
id="of1"/>
    <af:inputText label="ID : "
value="#{owcp.appsint.data('ID','i1','1','GET.ResultSet.ID','')}" id="it1"/>
  </af:panelGroupLayout>
</af:iterator>
```

For a REST data source, given the following example of returned data, where emp is repeating data:

```
<emps>
   <emp>
      <name>EMP1</name>
      <id>1</id>
   </emp>
   <emp>
      <name>EMP2</name>
      <id>2</id>
   </emp>
</emps>
```

The iterator hint is GET.emps.emp and the value hint is GET.emps.emp.name.

Example JSFF with binding hints to list employees for the REST data shown above:

```
<af:iterator id="i1" value="#{owcp.appsint.data('Employee
List','','None','GET.emps.emp','')}" var="row">
   <af:panelGroupLayout id="pgl2" layout="vertical">
      <af:outputText
value="#{owcp.appsint.data('Name','i1','John','GET.emps.emp.name','')}" id="of1"/>
      <af:inputText label="ID : "
value="#{owcp.appsint.data('ID','i1','1','GET.emps.emp.id','')}" id="it1"/>
   </af:panelGroupLayout>
</af:iterator>
```

> **Note:** If the data type returned by the **Resource Path** URL (specified in the Define Data Visualization wizard) for the REST data source is in JSON format, then methodReturn must be added after *operationName* in the binding hint. For example: GET.methodReturn.name

Binding Template Placeholder EL to a Business Object Parameter

Placeholder EL syntax:

```
#{owcp.appsint.parameter('displayName','','defaultValue','parame
ter binding hint','')} where:
```

- `owcp.appsint.parameter` is a keyword to indicate that this placeholder is to be bound to a business object *parameter*. Only placeholder EL that contains `owcp.appsint.parameter` will be bound to a business object parameter at runtime.

- *displayName* is the placeholder display name of the attribute that will be shown to the user in the Define Data Visualization wizard Options page.

- *defaultValue* is the default value for the component to render it in JDeveloper for testing purposes.

- *parameter binding hint* is the xpath (*operationName.parameter1*) for the parameter, so that the exposed attribute in the EL is bound to the business object parameter by default.

  For a SQL data source, the binding hint is `GET.bind_param_name`. For example, for a SQL data source that retrieves data using a bind parameter named `type` (`select * from some_table where column= :type`), the binding hint is `GET.type`.

  For a REST data source, the binding hint is `GET.query_param` or `GET.path_param`, assuming a `GET` operation on REST.

Binding Template Placeholder EL to a Business Object Method

Placeholder EL syntax:

```
#{owcp.appsint.method('displayName','','defaultValue','method
binding hint','')} where:
```

- `owcp.appsint.method` is a keyword to indicate that this placeholder is to be bound to a business object *method*. Only placeholder EL that contains `owcp.appsint.method` will be bound to a business object method at runtime.

- *displayName* is the placeholder display name of the attribute that will be shown to the user in the Define Data Visualization wizard Options page.

- `defaultValue` is the default value for the component to render it in JDeveloper for testing purposes.

- *method binding hint* is the xpath for the method, so that the exposed attribute in the EL is bound to the business object method by default.

Binding Template Placeholder EL to a Business Object Attribute and Parameter

Placeholder EL syntax:

```
#{owcp.appsint.inout('displayName','','defaultValue','attribute
binding hint','parameter binding hint')} where:
```

- `owcp.appsint.inout` is a keyword to indicate that this placeholder is to be bound to a business object *attribute and parameter*.

- *displayName* is the placeholder display name of the attribute that will be shown to the user in the Define Data Visualization wizard Options page.

- *defaultValue* is the default value for the component to render it in JDeveloper for testing purposes.

- *attribute binding hint* is the xpath
  (*operationName.accessor1.accessor2.attribute*) for the attribute, so
  that the exposed attribute in the EL is bound to the business object attribute by
  default.

- *parameter binding hint* is the xpath (*operationName.parameter1*) for
  the parameter, so that the exposed attribute in the EL is bound to the business
  object parameter by default.

Example: Binding Template Placeholder EL to a Business Object Attribute

This template placeholder EL will display as `Name of the Employee` in the Define
Data Visualization wizard Options page and it will be bound to the business object
attribute. Parent information is empty for the table component, as the table component
does not depend on any parent.

```
<af:outputText value="#{owcp.appsint.data('Name of the
Employee','row','John','GET.emp.name','')}" />
<af:table value="#{owcp.appsint.data('Name of the
Employee','' ,None,'GET.emps.emp','')}"   id="t1">
   <af:column sortable="false" headerText="#{owcp.appsint.data('Fourth Column
Header','t1','Column Header','GET.emps.emp.name','') }" id="c4">
      <af:outputText value="#{owcp.appsint.data('Fourth
Column','t1','Text','GET.emps.emp.name','')}" id="ot4"/>
   </af:column>
</af:table>
```

Example: Search and Update Employee Data

This example illustrates placeholder EL that binds to a business object parameter,
method, and attribute/parameter.

Assuming the REST data source supports `GET` and `PUT` operations:

- For `GET` - Retrieve employee based on `id`:

  Query parameter: `id`
  Response Payload: `<employee><id>1</id><name>Arnold</name></employee>`

- For `PUT` - Update employee:

  Request Payload: `<employee><id>1</id><name>Arnold</name></employee>`

> **Note:** Request payload contains `id` to which the update applies.

The visualization template for this example may look as follows:

```
<?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"
                                                    xmlns:f="http://
java.sun.com/jsf/core">
   <af:panelGridLayout id="pgl1">
      <af:gridRow id="gr1">
         <af:gridCell halign="stretch" valign="stretch" id="gc3">
            <af:panelHeader text="Update portal details" id="ph3"/>
            <af:panelFormLayout id="pfl4">
               <af:inputText label="id" editable="always"
value="#{owcp.appsint.parameter('id','','','GET.id','')}" id="search1"/>
```

```
                     <af:spacer height="20" id="s2"/>
                     <af:button text="Search emp" id="b1"
actionListener="#{owcp.appsint.method('Search emp','','None','GET','')}"/>
                     <af:inputText label="name"
value="#{owcp.appsint.inoutdata('name','','','GET.employee.name','PUT.employee.name')
}"
                                  id="it8" partialTriggers="::b1"/>
                     <af:inputText label="id"
value="#{owcp.appsint.inoutdata('id','','','GET.id','PUT.employee.id')}"
                                  id="it10" partialTriggers="::b1"/>
                     <af:spacer height="20" id="s6"/>
                     <af:button text="Update employee" id="b3"
actionListener="#{owcp.appsint.method('update employee','','None','PUT','')}"/>
                 </af:panelFormLayout>
             </af:gridCell>
         </af:gridRow>
     </af:panelGridLayout>
</jsp:root>
```

9.4 Publishing a Visualization Template

After creating a WebCenter Portal asset application and editing the JSFF file for the visualization template, the next step is to publish it and test it in WebCenter Portal. For instructions on how to publish a visualization template to WebCenter Portal as a shared asset, or to a specific portal as a portal asset, see Publishing WebCenter Portal Assets.

9.5 What Happens at Runtime

When an application specialist or portal manager configures a data visualization at runtime and selects a custom visualization template, WebCenter Portal dynamically renders the wizard's **Options** page based on the placeholder EL that needs to be bound to business object attributes.

Figure 9-4 shows an example **Options** page for a custom visualization template selected in the Define Data Visualization wizard.

Figure 9-4   Define Data Visualization Wizard: Options Page
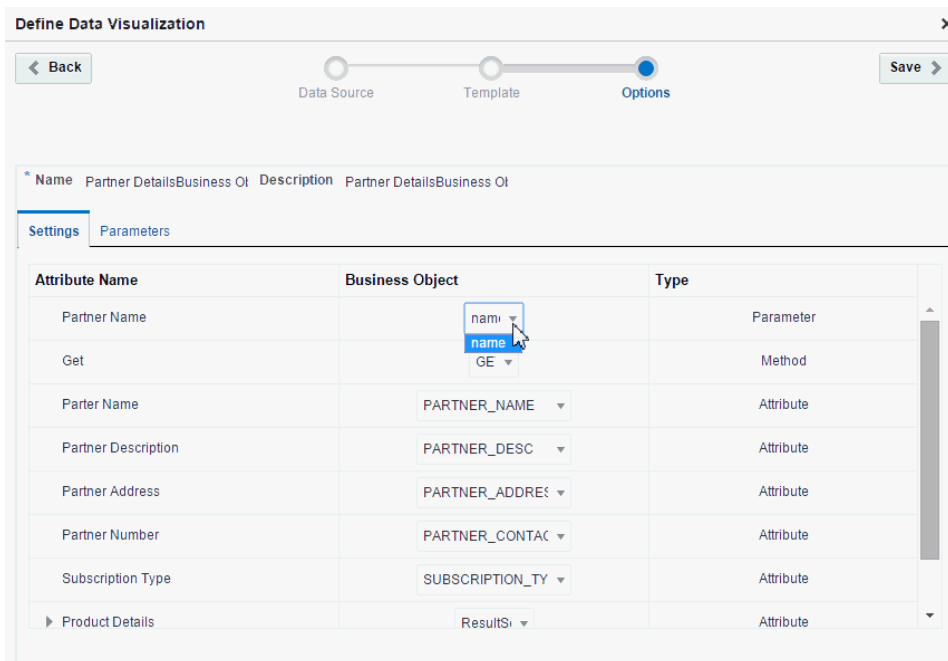
When the user clicks **Save** in the Define Data Visualization wizard, WebCenter Portal assigns the selected business object attributes to the visualization template. The page definition is also updated with the associated information. For example, before binding, the visualization template code for the Partner Name parameter value shown in Figure 9-4 may look like this:

```
...
<af:panelLabelAndMessage label="Partner Name" id="plam3">
   <af:inputText simple="true" value="#{owcp.appsint.parameter('Partner
Name','','None','GET.NAME','')}" id="it7"/>
</af:panelLabelAndMessage>
...
```

After binding, the visualization fragment will look like this:

```
...
<af:panelLabelAndMessage label="Partner Name" id="plam3">
   <af:inputText simple="true" value="#{bindings.name.inputValue}" id="it7"/>
</af:panelLabelAndMessage>
...
```

10

Developing Content Presenter Display Templates

This chapter describes how to use Oracle JDeveloper to create, edit, and publish new Content Presenter display templates for use in WebCenter Portal.

> **Note:**
>
> Content Presenter can only be used with Oracle WebCenter Content Server-based content. No other content repository connection types are supported.

This chapter includes the following topics:

- Introduction to Developing Content Presenter Display Templates
- Creating a Content Presenter Display Template
- Editing a Content Presenter Display Template
- Publishing a Content Presenter Display Template
- Optimizing Performance for Content Presenter Display Templates
- Using Content Presenter (Tips, Tutorials, and Examples)

10.1 Introduction to Developing Content Presenter Display Templates

A *Content Presenter display template* is a JSFF file (JSF page fragment) that defines how Content Presenter renders content items (including images and text) on a portal page.

WebCenter Portal provides several out-of-the-box display templates to get you started, but you can also create your own templates to solve specific display requirements.

Some typical situations where Content Presenter display templates provide value are:

- Providing different layouts for different parts of a page. For example, you may have articles from different sources on the same page, each requiring its own layout (for more information, see How to Define a Multiple-Item Display Template).

- Presenting content based on the capabilities of the viewing device. You can provide display templates that respond to whether the viewing device is a standard monitor, tablet, or smart phone (for more information, see How to Use Responsive Templates).

Making content available through Content Presenter display templates lets you solve complex display issues through a standardized template rather than doing each individual layout by hand. By making display templates available to others you enable

them to solve layout requirements based on predefined department or company standards without developer involvement.

> **Tip:**
>
> You can find sample display templates in `$ORACLE_HOME/jdeveloper/webcenter/samples/contentpresenter`. These sample files were installed as part of WebCenter Portal's extension for Oracle JDeveloper.

Display templates can use the full set of Rich ADF components, so you can quickly and easily create robust and attractive templates to display your content. Note however, that you are not required to use these components in your template.

A Content Presenter display template can handle either single-content items, multiple-content items, or combinations of the two. For example, a multiple content item template might render tabs for each item and then call a single item template to render the details of a selected item.

Each content item is associated with a specific *content type* defined in the Oracle WebCenter Content Server repository. A content type defines the properties of the content item. Content types can map to WebCenter Content Server profile definitions and Site Studio region definitions.

> **Tip:**
>
> Oracle recommends that you use Content Presenter display templates to integrate Site Studio and WebCenter Portal instead of Site Studio region templates. The recommended flow is:
>
> - Develop region definitions in Site Studio.
>
> - Develop Content Presenter display templates referencing region definitions using JDeveloper.
>
> - Publish the templates to WebCenter Portal.
>
> - Use Content Presenter to render the content and to enable users to contribute content.

Content types are created on the content repository (that is, WebCenter Content Server). As a Content Presenter display template developer, you need to know the names of the properties defined for the associated content type so that you can define how to display the selected content item(s) on the page.

> **Tip:**
>
> One way to determine the properties for the existing content types defined in WebCenter Content Server is to use the Content Presenter Configuration dialog in a portal. For a detailed description of this technique, see How to Discover Content Type Property Names.

At runtime, an authorized end user can choose a display template in the Content Presenter Configuration dialog. For more information about best practices and determining when and how to use Content Presenter templates, see the blog entry at:

```
http://www.ateam-oracle.com/portal-and-content-content-
integration-best-practices.
```

10.2 Creating a Content Presenter Display Template

This section describes how to create a new custom visualization template and includes the following topics.

- How to Create a Content Presenter Display Template
- What You May Need to Know About Content Presenter Display Templates

10.2.1 How to Create a Content Presenter Display Template

This section describes how to create a WebCenter Portal asset application for a new Content Presenter display template.

To create a WebCenter Portal asset application for a Content Presenter display template:

1. Create an asset application, specifying the **Asset Type** as either **Content Presenter Template - Single Item** or **Content Presenter Template - Multiple Items** (Figure 10-1).

   Figure 10-1   Content Presenter Display Template Asset Type
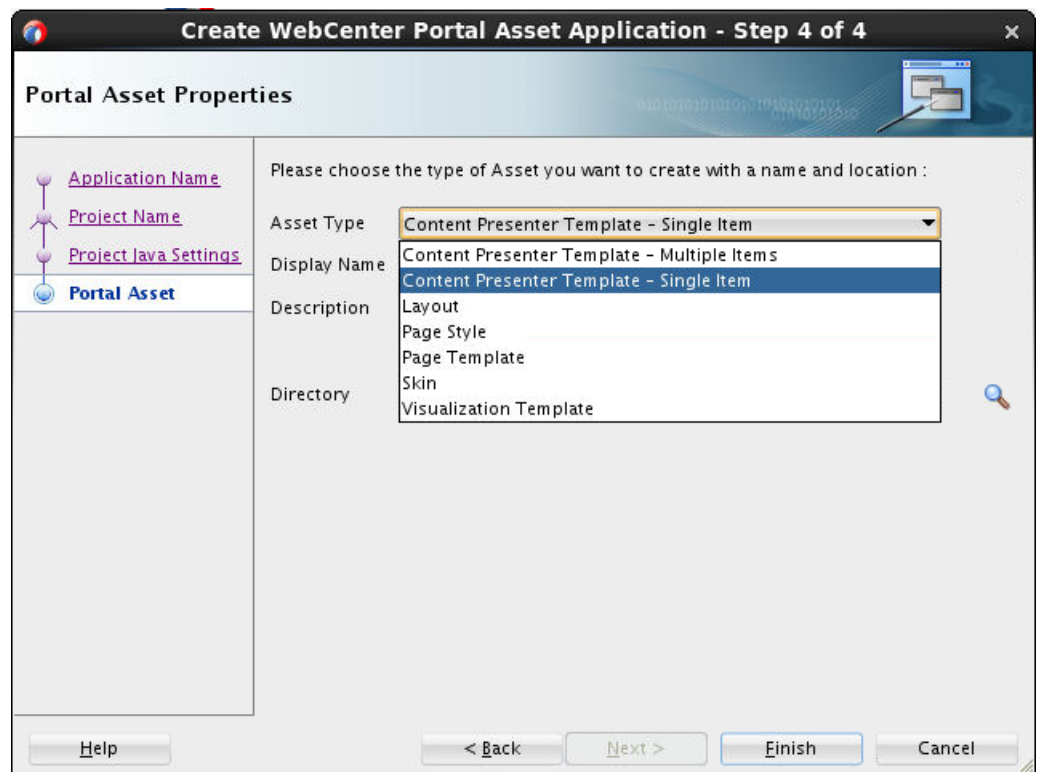
   See Creating a WebCenter Portal Asset Application for more information about creating WebCenter Portal asset applications. For information about the artifacts that are created when you create an asset application for a Content Presenter display template see What You May Need to Know About Content Presenter Display Templates.

2. In the Application Navigator, right-click the newly created Content Presenter display template's page fragment file (Figure 10-2), choose **Open**, then switch to **Source** view.

Figure 10-2   Content Presenter Display Template Page Fragment File
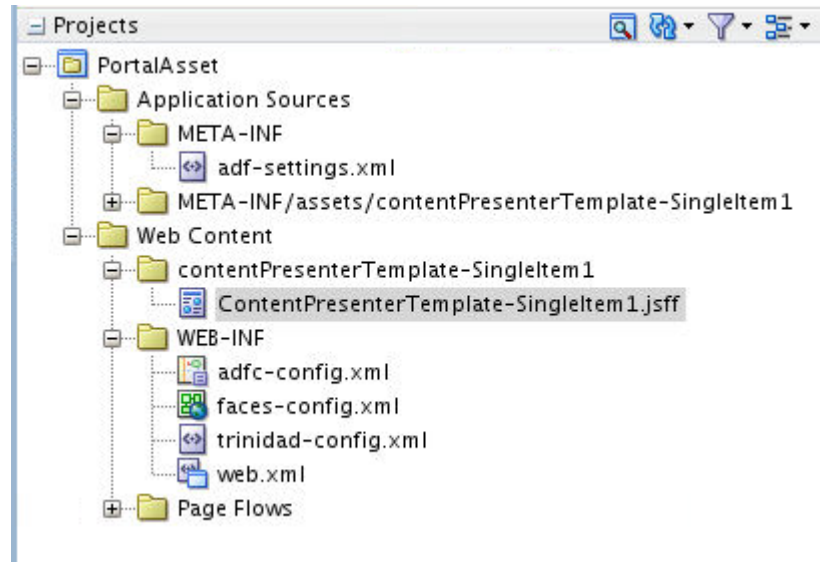
3. Continue by editing the contents of the page fragment as described in Editing a Content Presenter Display Template.

10.2.2 What You May Need to Know About Content Presenter Display Templates

Depending on your needs, the approach you take to defining Content Presenter display templates will vary. Typically, you define display templates for specific single items of content, then define a multiple content item display template that includes calls to the single item display templates.

Creating a visualization template asset application produces a default template, with the following artifact:

- A view fragment file (for example, `ContentPresenterTemplate-SingleItem1.jsff`)

This file appears in the Application Navigator as shown in Figure 10-3.

Figure 10-3   Content Presenter Display Template Asset Application Artifacts
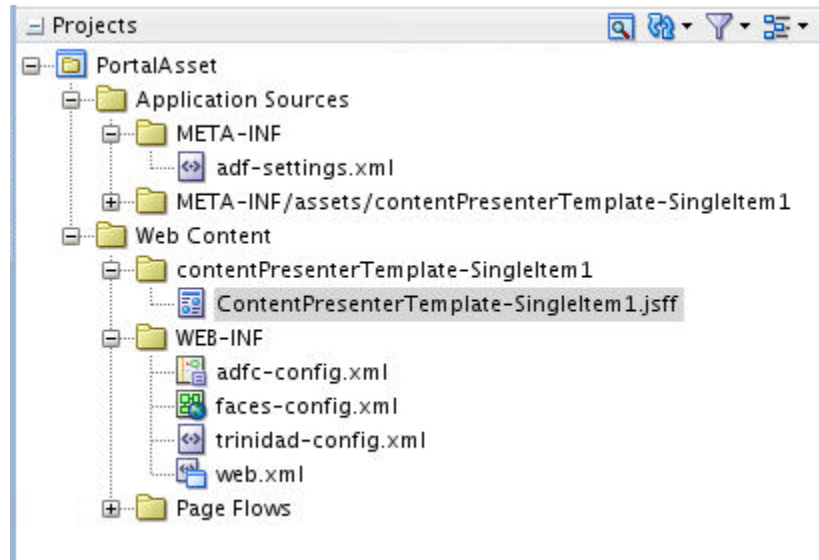
Template definitions can include calls to other display templates in any of the following ways:

- A single content item display template can call another single content item display template.

- A multiple content item display template can call a single content item display template (as shown in the examples below).

- A multiple content item display template can call another multiple content item display template.

The basic tasks for creating Content Presenter display templates include:

- Deciding whether to create a single or multiple item display template. See How to Define a Single-Item Display Template and How to Define a Multiple-Item Display Template.

- Deciding what kind of information you want to display about the selected content items. Consider also, for example, using WebCenter Portal's EL expressions in your template to retrieve and display this information. See How to Use EL Expressions to Retrieve Content Item Information and Expression Language Expressions for more information about using EL expressions.

- Designing the look and feel of the template. In addition to standard JSFF constructs, display templates can use ADF components (see *Developing Web User Interfaces with Oracle ADF Faces*).

- Publishing the template, either as a shared or portal-specific asset. See Publishing a Content Presenter Display Template.

- Configuring Content Presenter to use Coherence as the caching mechanism for production (optional, but recommended) and HA environments (required) as described in Modifying Cache Settings for Content Presenter in *Administering Oracle WebCenter Portal*.

10.3 Editing a Content Presenter Display Template

After creating the initial framework of a Content Presenter display template, you must edit it to define the template.

This section discusses these topics:

- How to Define a Single-Item Display Template
- Content Display Template Tags for Single Content Items
- How to Define a Multiple-Item Display Template
- Content Display Template Tags for Multiple Content Items
- How to Use Responsive Templates
- How to Extend Responsive Templates
- How to Use Image Renditions in Content Presenter Display Templates
- How to Use EL Expressions to Retrieve Content Item Information
- How to Discover Content Type Property Names
- How to Reference External Files in Display Templates
- How to Reference Site Studio Region Elements in a Custom View

10.3.1 How to Define a Single-Item Display Template

Examples of single content items for which you may want to create Content Presenter display templates are:

- Individual items to display with a specific look and feel on a page.
- Different views of a specific type of item (such as short and detailed views of an article).
- Different versions of a similar item (such as Press Releases that may be formatted differently for different groups and using a different set of properties).

To define a single-item display template:

1. In the Application Navigator, right-click the display template JSFF file, and choose **Open**.

2. If necessary, select **Window**, then **Components** to open the Component Palette.

3. From the dropdown list at the top of the Component Palette (Figure 10-4):

   - Select **WebCenter Content Display Templates** for a list of display template tags.

   - Select **WebCenter Content Management Faces** for the renderProperty tag.

Figure 10-4   Content Display Template Tags in Component Palette
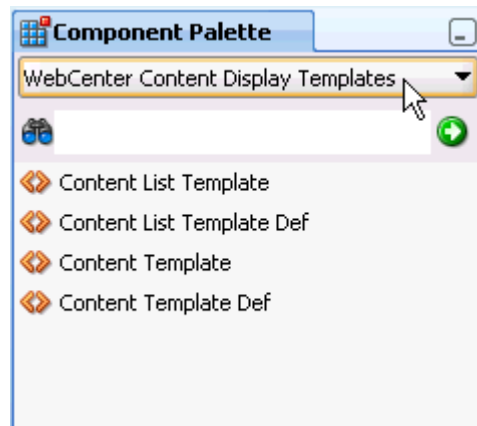
4. In Source view, drag and drop the required display template tags from the Component Palette onto the page to define your template. Refer to Content Display Template Tags for Single Content Items for information about each tag and required parameter values.

The following two examples show sample single content item display template definitions. These examples illustrate a use case where a certain kind of document (a press release) is produced by two different departments inside a company, and each department has defined its own content type and properties. These sample Content Presenter display templates allow these two different content types to be displayed in a consistent manner.

The template shown in the first example handles a press release that uses a property named xHeading to describe the heading of the press release document, and xDestinationUrl to describe the location of the document. To learn how you can retrieve these property names for a given content type, see How to Discover Content Type Property Names.

```xml
<?xml version = '1.0'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
          version="2.1"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
          xmlns:dt="http://xmlns.oracle.com/webcenter/content/templates">
   <dt:contentTemplateDef var="node">
      <af:goImageLink text="#{node.propertyMap['xHeading'].value}"
                      id="gil1"
                      icon="#{node.icon.smallIcon}"
                      destination="#{node.propertyMap['xDestinationUrl'].value}"
                      targetFrame="_blank">
      </af:goImageLink>
   </dt:contentTemplateDef>
</jsp:root>
```

The template shown in the next example handles a press release that uses a property named dDocTitle to describe the heading of the press release document, and xLinkUrl to describe the location of the document.

```xml
<?xml version = '1.0'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
          version="2.1"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
          xmlns:dt="http://xmlns.oracle.com/webcenter/content/templates">
   <dt:contentTemplateDef var="node">
```

```
<af:goImageLink text="#{node.propertyMap['dDocTitle'].value}"
                id="gil1"
                icon="#{node.icon.smallIcon}"
                destination="#{node.propertyMap['xLinkUrl'].value}"
                targetFrame="_blank">
        </af:goImageLink>
      </dt:contentTemplateDef>
    </jsp:root>
```

10.3.2 Content Display Template Tags for Single Content Items

The definition of a single content item display template uses the JSP tasks listed below.

Table 10-1 Content Display Template Tags for Single Content Items

JSP Tag	Description	Example
contentTemplateDef	Required. Defines a single content item template. **Attributes**: var - Specifies the content node that will be rendered by this display template. In the template definition code, you can use the EL expressions described in How to Retrieve Basic Information About a Content Item to retrieve required information about the node.	`<dt:contentTemplateDef var="node">` `<af:outputText value="#{node.name}" />` `</dt:contentTemplateDef>`

Table 10-1   (Cont.) Content Display Template Tags for Single Content Items

JSP Tag	Description	Example

renderProperty	Optional. Retrieves and renders the string value(s) of the specified node property inline.
	
	Attributes:
	
	id - Identifies this component. This value must be unique within the closest parent component that is a naming container.
	
	name - Specifies the name of the property to get. If this is a system property (cm_createdBy, cm_createdDate, cm_modifiedDate, and cm_path) then the value on the node will be used. If not specified, then the primaryProperty will be used if defined.
	
	node - Specifies the node to use. This value can either be a bound attribute to a managed bean, or a request attribute variable.
	
	blockSize - Specifies the size of the blocks in bytes, to read the bytes of a binary property. Default is 2048 bytes.
	
	startIndex - Specifies the index (from 0) in the document's bytes at which to start printing. Defaults to 0.
	
	endIndex - Specifies the index (from 0) in the document's bytes at which to stop printing. Defaults to value of blockSize.
	
	rendered - Specifies whether or not this component should be rendered (during Render Response Phase), or processed on any subsequent form submit. The default value is true.
		`<!--`
		` Handling of text-based primary`
		` properties (HTML, text, etc.).`
		`-->`
		`<dt:contentTemplateDef`
		`var="node">`
		` <cmf:renderProperty id="rp1"`
		`name="#{node.primaryProperty.nam`
		`e}"`
		`node="#{node}"/>`
		`</dt:contentTemplateDef>`

Table 10-1   (Cont.) Content Display Template Tags for Single Content Items

JSP Tag	Description	Example

`contentTemplate`	Optional. Nested under `contentTemplateDef`.
	
	Calls another single item template.
	
	Attributes:
	
	`node` - Required. Specifies the content repository node that should be displayed.
	
	`nodesHint` - Optional. When the display template is called in an iterating component, allows the pre-inclusion of all possible templates. This attribute is not needed when `contentTemplate` is used inside a `contentTemplateDef` tag.
	
	`view` - Optional. Specifies the target view name.
	
	`id` - Optional. Specifies the JSF component ID.
	
	`rendered` - Optional. Specifies whether the component should be rendered.
		`<dt:contentTemplateDef var="node">`
		`  <af:outputText value="#{node.name}" >`
		`   <dt:contentTemplate node="#{node}" view="templates.pressrelease.item" />`
		`  </af:outputText>`
		`</dt:contentTemplateDef>`

10.3.3 How to Define a Multiple-Item Display Template

Examples of multiple content items for which you may want to create Content Presenter display templates are:

- A group of similar items that you want to display on a page, such as a list of books or an employee directory of pictures.

- Query results, such as all documents modified in the last week.

- A list of all documents in an Oracle WebCenter Content Server folder.

A Content Presenter display template can also combine both single and multiple items. For example, a multiple content item template might render tabs for each item and then call a single item template to render the details of a selected item. For an example of how to do this, see the blog entry at:

http://www.ateam-oracle.com/enable-content-editing-of-iterative-components

To define a multiple-item display template:

1. In the Application Navigator, right-click the display template JSFF file, and choose **Open**.

2. If necessary, select **Window**, then **Components** to open the Component Palette.

3. From the dropdown list at the top of the Component Palette (Figure 10-5):

   - Select **WebCenter Content Display Templates** for a list of display template tags.

- Select **WebCenter Content Management Faces** for the `renderProperty` tag.

Figure 10-5   Content Display Template Tags in Component Palette
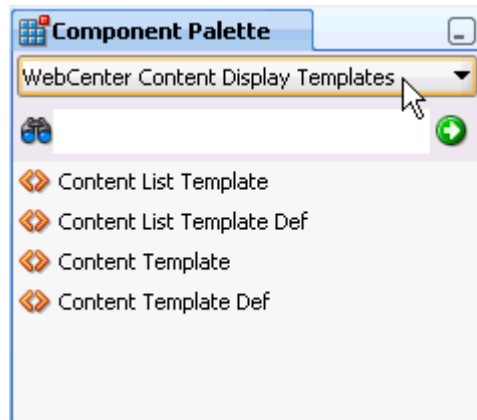
4. In Source view, drag and drop the required display template tags from the Component Palette onto the page. Refer to Content Display Template Tags for Multiple Content Items for information about each tag and required parameter values.

The following example shows a sample multiple content item display template definition.

This template definition iterates over the data items selected for display, and processes them according to the referenced view (view="mycorp.content.templates.pressrelease.listitem").

```
<?xml version = '1.0'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
         xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
         xmlns:dt="http://xmlns.oracle.com/webcenter/content/templates">
   <dt:contentListTemplateDef var="nodes">
      <af:panelGroupLayout layout="scroll" id="nodeListPanel" valign="middle">
        <af:iterator rows="0" var="node" varStatus="iterator" value="#{nodes}">
           <dt:contentTemplate node="#{node}"
               view="mycorp.content.templates.pressrelease.listitem"
               nodesHint="#{nodes}"/>
        </af:iterator>
      </af:panelGroupLayout>
   </dt:contentListTemplateDef>
</jsp:root>
```

10.3.4 Content Display Template Tags for Multiple Content Items

The definition of a multiple content item display template uses the JSP tags listed below.

Table 10-2   Content Display Template Tags for Multiple Content Items

Table 10-2 (Cont.) Content Display Template Tags for Multiple Content Items

JSP Tag	Description	Example
contentListTemplateDef	Required. Defines the multiple content item template. **Attributes**: var - Specifies the content node that will be rendered by this display template. In the template definition code, you can use the EL expressions described in How to Retrieve Basic Information About a Content Item to retrieve required information about the node.	```<dt:contentListTemplateDef var="nodes">``` ```<af:iterator value="#{nodes}" var="node">``` ```<af:outputText value="#{node.name}" />``` ```</af:iterator>``` ```</dt:contentListTemplateDef>```
contentListTemplate	Optional. Nested under contentListTemplateDef. Calls another multiple item template. **Attributes**: nodes - Required. Provides the list of VCR nodes that should be displayed category - Required. Specifies the target template category. view - Required. Specifies the target view name. id - Optional. Specifies the JSF component ID. rendered - Optional. Specifies whether the component should be rendered.	```<dt:contentListTemplateDef var="nodes">``` ```<!--``` ``` Reuse the default bulleted list view, but``` ``` indent it with a <blockquote>``` ```-->``` ```<f:verbatim>``` ```<blockquote>``` ```</f:verbatim>``` ```<dt:contentListTemplate nodes="#{nodes}"``` ```category="oracle.webcenter.content.``` ```templates.default.category"``` ```view="oracle.webcenter.content.``` ```templates.default.list.bulleted"/>``` ```<f:verbatim>``` ```</blockquote>``` ```</f:verbatim>``` ```</dt:contentListTemplateDef>```

Table 10-2   (Cont.) Content Display Template Tags for Multiple Content Items

JSP Tag	Description	Example
`contentTemplate`	Optional. Nested under `contentListTemplateDef`. Calls a single item template. **Attributes**: `node` - Required. Specifies the content repository node that should be displayed. `nodesHint` - Optional. When the display template is called in an iterating component, allows the pre-inclusion of all possible templates. This attribute is usually required when `contentTemplate` is used inside a `contentListTemplateDef` tag. `view` - Optional. Specifies the target view name. `id` - Optional. Specifies the JSF component ID. `rendered` - Optional. Specifies whether the component should be rendered.	```<dt:contentListTemplateDef var="nodes"> <af:iterator rows="0" var="node" varStatus="iterator" value="#{nodes}"> <dt:contentTemplate node="#{node}" view="templates.pressrelease.listitem" nodesHint="#{nodes}"/> </af:iterator> </dt:contentListTemplateDef>```

10.3.5 How to Use Responsive Templates

This section includes the following topics:

- About Responsive Templates
- Prerequisites for Responsive Templates
- Displaying Multiple Articles
- Using CSS3 Media Queries

10.3.5.1 About Responsive Templates

Using CSS3 media queries, you can render content to adapt to conditions such as screen resolution.

The out-of-the-box Content Presenter display templates `Articles View` and `Full Articles View` are examples of how CSS3 media queries can be used in a Content Presenter display template for a responsive layout. For information about how these responsive templates can be extended, see How to Extend Responsive Templates. For information about optimizing viewport settings for mobile devices such as smart phones and tablets, see Optimizing Portals for Mobile Devices in *Building Portals with Oracle WebCenter Portal*.

10.3.5.2 Prerequisites for Responsive Templates

The out-of-the-box Content Presenter templates `Articles View` and `Full Articles View` rely on the Site Studio `RD_ARTICLE` region definition.

Consequently, Site Studio must be enabled in Oracle WebCenter Content Server to enable the `Articles View` and `Full Article View` Content Presenter display templates to be used.

Follow the steps below to enable Site Studio and seed WebCenter Content Server with the `RD_ARTICLE` region definition:

1. Enable Site Studio (see Understanding Site Studio Integration in *Building Portals with Oracle WebCenter Portal*).

2. Start (or restart) WebCenter Portal after Site Studio has been enabled (this will seed the `RD_ARTICLE` region definition).

10.3.5.3 Displaying Multiple Articles

The template definition in the following example iterates over the data items selected for display, and uses three style classes:

- **`article`** - applied to each content item

- **`article-3`** - applied to each content item that will be in the first column if the page is split into rows of three columns

- **`article-2`** - applied to each content item that will be in the first column if the page is split into rows of two columns

The style classes are defined in the `articles.css` style sheet.

```
<?xml version = '1.0'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
        xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
        xmlns:dt="http://xmlns.oracle.com/webcenter/content/templates"
        xmlns:f="http://java.sun.com/jsf/core"
        xmlns:h="http://java.sun.com/jsf/html"
        xmlns:rah="http://xmlns.oracle.com/webcenter/resourcehandler"
        <dt:contentListTemplateDef var="nodes">
    <af:resource type="css" source="/oracle/webcenter/content/templates/seeded/
articles.css"/>
    <af:iterator rows="0" var="node" varStatus="iterator" value="#{nodes}"
                 id="it0">
        <h:panelGroup styleClass="#{(iterator.index % 3 == 0) ? 'article-3 ' :
''}#{(iterator.index % 2 == 0) ? 'article-2 ' : ''}article"
                      id="pg1">
            <af:commandLink immediate="true" partialSubmit="true" id="cl2">
                <rah:resourceActionBehavior id="rah1"

serviceId="oracle.webcenter.content.presenter"
                                        resourceId="#{node.id}"

resourceTitle="#{node.propertyMap['RD_ARTICLE:TITLE'].asTextHtml}"
                                        useResourcePopup="never"/>
                <f:attribute name="taskFlowInstId"
                             value="a5fafea8-90e6-4972-997d-314401b6c98b"/>
                <f:attribute name="datasourceType" value="dsTypeSingleNode"/>
                <f:attribute name="datasource"
```

```
value="#{node.id.repositoryName}#dDocName:#{node.propertyMap['dDocName'].value}"/>
                <f:attribute name="templateView"

value="oracle.webcenter.content.templates.sitestudio.fullarticle"/>
                <f:attribute name="regionTemplate" value="#{false}"/>
                <af:outputText
value="#{node.propertyMap['RD_ARTICLE:IMAGE'].asTextHtml}"
                               escape="false" id="ot4"/>
                <tr:panelHeader
text="#{node.propertyMap['RD_ARTICLE:TITLE'].asTextHtml}"
                               id="ph1">
                   <af:outputText
value="#{node.propertyMap['RD_ARTICLE:SUMMARY'].asTextHtml}"
                               escape="false" id="ot3"/>
                </af:panelHeader>
            </af:commandLink>
         </h:panelGroup>
      </af:iterator>
    </dt:contentListTemplateDef>
</jsp:root>
```

10.3.5.4 Using CSS3 Media Queries

The style sheet in the following example uses media queries to render the content in different ways depending on the width of the browser.

```
/* Default styles */
.article {
  display: block;
  float: left;
  width: 31.623931623931625%;
  margin-left: 2.564102564102564%;
  margin-bottom: 1em;
}
.article-3 {
  margin-left: 0;
  clear: both;
}
.article img {
  width: 100%;
  margin: 0 0 .25em;
  float: none;
  padding-top: 0;
  display: block;
  border: 0;
}
.article h1 {
  font-size: 1.5em;
}

@media only screen and (max-width : 480px) {
  /* up to width of iPhone (excluding iPhone 5 in landscape) */
  .article {
    margin-top: .5em;
    float: left;
    display: inline;
    width: 100%;
    margin-left: 0;
  }
  .article-3 {
    width: 100%;
    clear: none;
```

```
    }
    .article img {
      margin-right: 4%;
      width: 48%;
      float: left;
    }
    .article {
      font-size: 1em;
    }
    .article h1 {
      font-size: 1.25em;
    }
}

@media only screen and (min-width : 481px) and (max-width : 780px) {
  /* up to the width of iPad in portrait and iPhone 5 in landscape */
  .article {
    display: block;
    float: left;
    width: 48.71794871794871%;
    margin-left: 2.564102564102564%;
    clear: none;
  }
  .article-3 {
    margin-left: 2.564102564102564%;
    clear: none;
  }
  .article-2 {
    margin-left: 0;
    clear: both;
  }
  .article h1 {
    font-size: 1.25em;
  }
}

@media only screen and (min-width : 769px) and (max-width : 1024px) {
  /* up to the width of iPad in landscape */
}

@media only screen and (min-width : 1025px) {
  /* desktop */
}
```

10.3.6 How to Extend Responsive Templates

These sections describe the steps required to modify the out-of-the-box templates if they don't meet your local requirements.

For information about optimizing viewport settings for mobile devices such as smart phones and tablets, see Optimizing Portals for Mobile Devices in *Building Portals with Oracle WebCenter Portal*.

This section includes the following topics:

- How to Extend the Articles View Template

- How to Extend the Full Article View Template

- How to Adapt the Out-of-the-Box Templates

10.3.6.1 How to Extend the Articles View Template

To extend the `Articles View` template follow the steps below:

1. Create an empty template for displaying multiple items (see How to Define a Multiple-Item Display Template) and copy the contents of `articles.jsff` (located in the `JDEVELOPER/webcenter/samples/contentpresenter/` directory) into the empty template.

2. Embed the CSS used by `artices.jsff` into the new template by changing:

```
<af:resource type="css" source="/oracle/webcenter/content/templates/seeded/
articles.css"/>
```

To:

```
<af:resource type="css">
/* Default styles */
.article {
  display: block;
  float: left;
  width: 31.623931623931625%;
  margin-left: 2.564102564102564%;
  margin-bottom: 1em;
}
.article-3 {
  margin-left: 0;
  clear: both;
}
.article img {
  width: 100%;
  margin: 0 0 .25em;
  float: none;
  padding-top: 0;
  display: block;
  border: 0;
}
.article h1 {
  font-size: 1.5em;
}

@media only screen and (max-width : 480px) {
  /* up to width of iPhone */
  .article {
    margin-top: .5em;
    float: left;
    display: inline;
    width: 100%;
    margin-left: 0;
  }
  .article-3 {
    width: 100%;
    clear: none;
  }
  .article img {
    margin-right: 4%;
    width: 48%;
    float: left;
  }
  .article {
    font-size: 1em;
```

```
    }
    .article h1 {
      font-size: 1.25em;
    }
  }

  @media only screen and (min-width : 481px) and (max-width : 780px) {
    /* up to the width of iPad in portrait */
    .article {
      display: block;
      float: left;
      width: 48.71794871794871%;
      margin-left: 2.564102564102564%;
      clear: none;
    }
    .article-3 {
      margin-left: 2.564102564102564%;
      clear: none;
    }
    .article-2 {
      margin-left: 0;
      clear: both;
    }
    .article h1 {
      font-size: 1.25em;
    }
  }

  @media only screen and (min-width : 769px) and (max-width : 1024px) {
    /* up to the width of iPad in landscape */
  }

  @media only screen and (min-width : 1025px) {
    /* desktop */
  }
  </af:resource>
```

3. Edit the new template to adapt it to your requirements (see How to Adapt the Out-of-the-Box Templates).

4. Publish the new template as a portal asset (see Publishing a Content Presenter Display Template).

10.3.6.2 How to Extend the Full Article View Template

Follow the steps below to extend the `Full Article View` template:

1. Create an empty template for displaying a single content item (see How to Define a Single-Item Display Template) and copy the contents of `full-article.jsff` (located in the `JDEVELOPER/webcenter/samples/contentpresenter/` directory) into the empty template.

2. Embed the CSS used by `full-artice.jsff` into the new template by changing:

```
<af:resource type="css" source="/oracle/webcenter/content/templates/seeded/
articles.css"/>
```

To:

```
<af:resource type="css">
/* Default styles */
.full-article h1 {
```

```
  font-size: 1.5em;
}
.full-article-image img {
  margin-left: 4%;
  width: 48%;
  float: right;
}

@media only screen and (max-width : 480px) {
  /* up to width of iPhone */
  .full-article-image img {
    margin-left: 0;
    width: 100%;
    float: none;
  }
}

@media only screen and (min-width : 481px) and (max-width : 780px) {
  /* up to the width of iPad in portrait */
}

@media only screen and (min-width : 769px) and (max-width : 1024px) {
  /* up to the width of iPad in landscape */
}

@media only screen and (min-width : 1025px) {
  /* desktop */
}
</af:resource>
```

3. Edit the new template to adapt it your requirements (see How to Adapt the Out-of-the-Box Templates).

4. Publish the new template as a portal asset (see Publishing a Content Presenter Display Template).

10.3.6.3 How to Adapt the Out-of-the-Box Templates

This section describes changes you may want to make to the Articles View and Full Article View templates, and outlines how you could go about making those changes.

For example:

- You may want to update the templates to support a responsive layout on older browsers like Internet Explorer 8.

- You may want to update the templates to work with a different region definition.

These extensions are described in the following sections:

- How to Support Responsive Layouts for Older Browsers

- How to Use Different Region Definitions

- How to Update the Layout in a Template

10.3.6.3.1 How to Support Responsive Layouts for Older Browsers

The current responsive templates rely on CSS3 media queries, which are not supported by older browsers, so you may want to use a third-party JavaScript library that can take the media queries and use JavaScript to enable a responsive design.

To support responsive layouts for older browsers:

1. Create and deploy a custom Shared Library for WebCenter Portal containing the third-party JavaScript library.

   See Developing Shared Libraries.

2. Add a reference to a third-party JavaScript library to the template (where *path/to/3rd/party/library.js* is the path to the JavaScript file within the Custom Shared Library):

   ```
   <af:resource type="javascript" source="path/to/3rd/party/library.js"/>
   ```

10.3.6.3.2 How to Use Different Region Definitions

This section shows you how you can modify the region definitions in your responsive template.

For example, you might have a region definition called `RD_NEWS` with four elements: `TITLE`, `LEAD`, `IMAGE`, and `BODY` (see How to Reference Site Studio Region Elements in a Custom View). You could then update the template to reference this new region definition by changing references of `RD_ARTICLES` to `RD_NEWS` and changing the references of `SUMMARY` to `LEAD`.

News View

This template displays a list of news items (of type `RD_NEWS`) based on the `Articles View` template:

```
<?xml version = '1.0'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
          xmlns:dt="http://xmlns.oracle.com/webcenter/content/templates"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:h="http://java.sun.com/jsf/html"
          xmlns:rah="http://xmlns.oracle.com/webcenter/resourcehandler"
          xmlns:tr="http://myfaces.apache.org/trinidad">
   <dt:contentListTemplateDef var="nodes">
      <af:resource type="css" source="/oracle/webcenter/content/templates/seeded/
articles.css"/>
      <af:iterator rows="0" var="node" varStatus="iterator" value="#{nodes}"
                   id="it0">
         <h:panelGroup styleClass="#{(iterator.index % 3 == 0) ? 'article-3 ' :
''}#{(iterator.index % 2 == 0) ? 'article-2 ' : ''}article"
                       id="pg1">
            <af:commandLink immediate="true" partialSubmit="true" id="cl2">
               <rah:resourceActionBehavior id="rah1"

serviceId="oracle.webcenter.content.presenter"
                                           resourceId="#{node.id}"

resourceTitle="#{node.propertyMap['RD_NEWS:TITLE'].asTextHtml}"
                                           useResourcePopup="never"/>
               <f:attribute name="taskFlowInstId"
                            value="a5fafea8-90e6-4972-997d-314401b6c98b"/>
               <f:attribute name="datasourceType" value="dsTypeSingleNode"/>
```

```
                <f:attribute name="datasource"
value="#{node.id.repositoryName}#dDocName:#{node.propertyMap['dDocName'].value}"/>
                <f:attribute name="templateView"
                             value="oracle.webcenter.content.templates.newsitem"/>
                <f:attribute name="regionTemplate" value="#{false}"/>
                <af:outputText value="#{node.propertyMap['RD_NEWS:IMAGE'].asTextHtml}"
                             escape="false" id="ot4"/>
                <tr:panelHeader text="#{node.propertyMap['RD_NEWS:TITLE'].asTextHtml}"
                               id="ph1">
                    <af:outputText
value="#{node.propertyMap['RD_NEWS:LEAD'].asTextHtml}"
                                   escape="false" id="ot3"/>
                </tr:panelHeader>
            </af:commandLink>
         </h:panelGroup>
      </af:iterator>
   </dt:contentListTemplateDef>
</jsp:root>
```

News Item View

This template displays a full news item (`View ID: oracle.webcenter.content.templates.newsitem`) based on the `Full Articles View` template:

```
<?xml version = '1.0'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
          xmlns:dt="http://xmlns.oracle.com/webcenter/content/templates"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:h="http://java.sun.com/jsf/html"
          xmlns:tr="http://myfaces.apache.org/trinidad">
    <dt:contentTemplateDef var="node">
       <af:resource type="css" source="/oracle/webcenter/content/templates/seeded/
articles.css"/>
       <af:panelGroupLayout id="psl1" layout="vertical">
           <tr:panelHeader text="#{node.propertyMap['RD_NEWS:TITLE'].asTextHtml}"
                           styleClass="full-article" id="ph1">
               <tr:panelGroupLayout id="pgl1" styleClass="full-article-image">
                   <af:outputText
value="#{node.propertyMap['RD_NEWS:IMAGE'].asTextHtml}"
                                  escape="false" id="ot4"/>
               </tr:panelGroupLayout>
               <af:outputText value="#{node.propertyMap['RD_NEWS:BODY'].asTextHtml}"
                              escape="false" id="ot3"/>
           </tr:panelHeader>
       </af:panelGroupLayout>
    </dt:contentTemplateDef>
</jsp:root>
```

10.3.6.3.3 How to Update the Layout in a Template

You might want to change the layout of the templates to better suit your content.

For example, in the single column layout the `Articles View` template will flow the text of an article summary under the image if the text is sufficiently long:

Figure 10-6   Article with Wide Layout

You could change this so that the text doesn't flow under the image:

Figure 10-7   Article with Narrow Layout
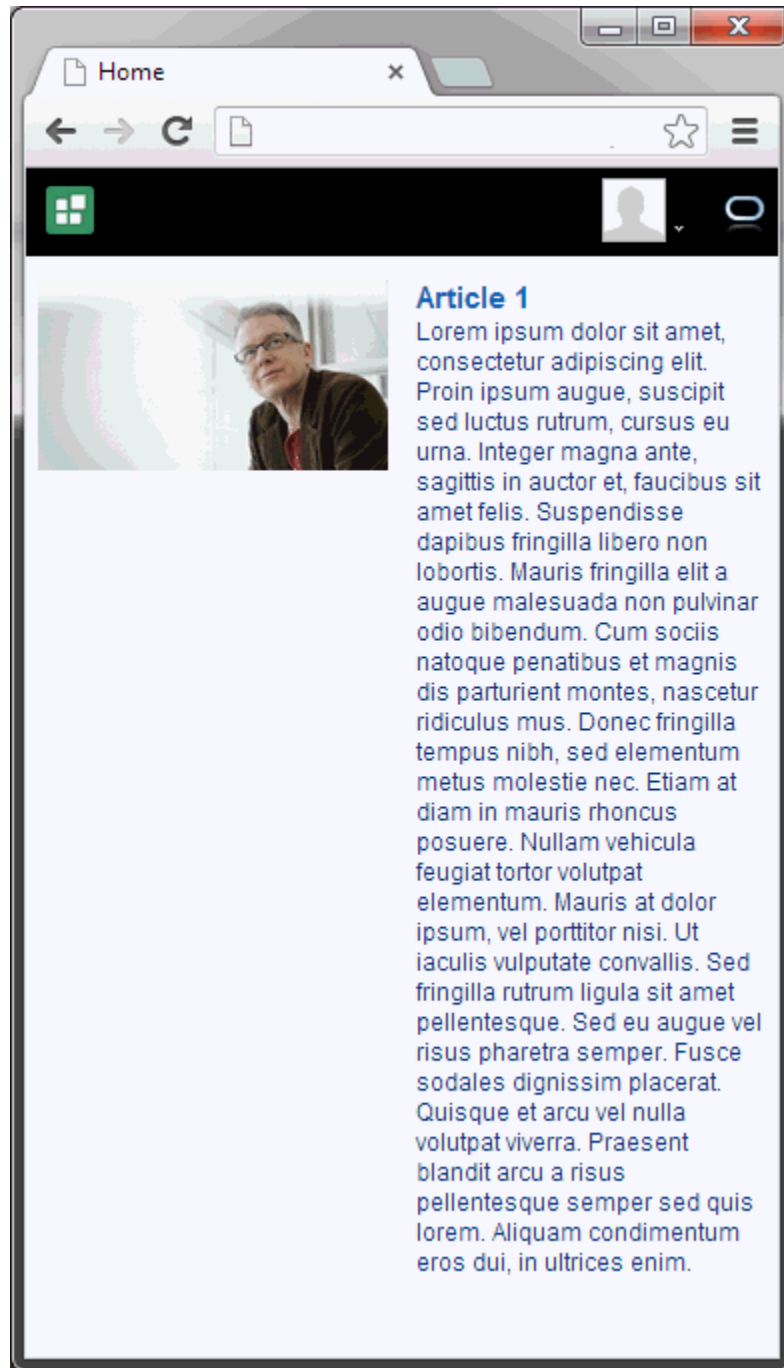
To do this, you could create a new template based on the `Articles View` template, add a style class to the block of text (`article-text`), and set the display CSS property for this style class to `table-cell` for the narrow layout:

```
<?xml version = '1.0'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
        xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
        xmlns:dt="http://xmlns.oracle.com/webcenter/content/templates"
        xmlns:f="http://java.sun.com/jsf/core"
        xmlns:h="http://java.sun.com/jsf/html"
        xmlns:rah="http://xmlns.oracle.com/webcenter/resourcehandler"
```

```
            <dt:contentListTemplateDef var="nodes">
              <af:resource type="css">
              /* Default styles */
              .article {
                display: block;
                float: left;
                width: 31.623931623931625%;
                margin-left: 2.564102564102564%;
                margin-bottom: 1em;
              }
              .article-3 {
                margin-left: 0;
                clear: both;
              }
              .article img {
                width: 100%;
                margin: 0 0 .25em;
                float: none;
                padding-top: 0;
                display: block;
                border: 0;
              }
              .article h1 {
                font-size: 1.5em;
              }

              @media only screen and (max-width : 480px) {
                /* up to width of iPhone */
                .article {
                  margin-top: .5em;
                  float: left;
                  display: inline;
                  width: 100%;
                  margin-left: 0;
                }
                .article-3 {
                  width: 100%;
                  clear: none;
                }
                .article img {
                  margin-right: 4%;
                  width: 48%;
                  float: left;
                }
                .article {
                  font-size: 1em;
                }
                .article h1 {
                  font-size: 1.25em;
                }
                /* Set the display CSS property to table-cell for the article-text style
      class in the narrow layout */
                .article-text {
                  display: table-cell;
                }
              }

              @media only screen and (min-width : 481px) and (max-width : 780px) {
                /* up to the width of iPad in portrait */
                .article {
                  display: block;
```

```
      float: left;
      width: 48.71794871794871%;
      margin-left: 2.564102564102564%;
      clear: none;
    }
    .article-3 {
      margin-left: 2.564102564102564%;
      clear: none;
    }
    .article-2 {
      margin-left: 0;
      clear: both;
    }
    .article h1 {
      font-size: 1.25em;
    }
  }

  @media only screen and (min-width : 769px) and (max-width : 1024px) {
    /* up to the width of iPad in landscape */
  }

  @media only screen and (min-width : 1025px) {
    /* desktop */
  }
  </af:resource>
  <af:iterator rows="0" var="node" varStatus="iterator" value="#{nodes}"
              id="it0">
      <h:panelGroup styleClass="#{(iterator.index % 3 == 0) ? 'article-3 ' :
''}#{(iterator.index % 2 == 0) ? 'article-2 ' : ''}article"
                    id="pg1">
          <af:commandLink immediate="true" partialSubmit="true" id="cl2">
              <rah:resourceActionBehavior id="rah1"

serviceId="oracle.webcenter.content.presenter"
                                          resourceId="#{node.id}"

resourceTitle="#{node.propertyMap['RD_ARTICLE:TITLE'].asTextHtml}"
                                          useResourcePopup="never"/>
              <f:attribute name="taskFlowInstId"
                           value="a5fafea8-90e6-4972-997d-314401b6c98b"/>
              <f:attribute name="datasourceType" value="dsTypeSingleNode"/>
              <f:attribute name="datasource"

value="#{node.id.repositoryName}#dDocName:#{node.propertyMap['dDocName'].value}"/>
              <f:attribute name="templateView"

value="oracle.webcenter.content.templates.sitestudio.fullarticle"/>
              <f:attribute name="regionTemplate" value="#{false}"/>
              <af:outputText
value="#{node.propertyMap['RD_ARTICLE:IMAGE'].asTextHtml}"
                              escape="false" id="ot4"/>
              <af:panelHeader
text="#{node.propertyMap['RD_ARTICLE:TITLE'].asTextHtml}"
                              id="ph1" styleClass="article-text">
                  <af:outputText
value="#{node.propertyMap['RD_ARTICLE:SUMMARY'].asTextHtml}"
                                  escape="false" id="ot3"/>
              </af:panelHeader>
          </af:commandLink>
      </h:panelGroup>
```

```
        </af:iterator>
      </dt:contentListTemplateDef>
   </jsp:root>
```

10.3.7 How to Use Image Renditions in Content Presenter Display Templates

This section includes the following topics:

- About Image Renditions

- Prerequisites for Image Renditions

- Retrieving Image Rendition Information for Image Documents

- Retrieving Image Renditions for Site Studio Region Definitions

10.3.7.1 About Image Renditions

In some cases you may want to use different renditions of an image in different circumstances.

For example, you might want to use a large, high resolution image when the page containing the image is displayed using a desktop browser. However, if the same page is displayed on a mobile device, a large, high resolution image might not be appropriate given the smaller screen size and possible slower download speed. For mobile devices it would be better to display a smaller, lower resolution version, or rendition, of the image. Content Presenter display templates provide the capability of specifying which rendition of an image to display under which circumstances.

To enable the use of different image renditions, you can use EL expressions in your Content Presenter display templates to determine which image renditions to use when.

> **Note:**
>
> Image renditions are not supported through CMIS.

10.3.7.2 Prerequisites for Image Renditions

For full image rendition support, the Oracle WebCenter Content Server where your images are checked in must have Digital Asset Management (DAM) enabled.

If DAM is not enabled, there is support for separate `web` and `thumbnail` renditions only. For information about enabling DAM, see Enabling Digital Asset Manager in *Administering Oracle WebCenter Portal*.

When DAM is enabled, different renditions are automatically created when an image is checked in, determined by the rendition set specified during check in. DAM provides some built-in rendition sets but the WebCenter Content Server administrator can also create new rendition sets. The individual renditions can then be referenced by name in Content Presenter display templates by using the appropriate EL expression.

For more information about DAM and rendition sets, see Working with Image and Video Conversions in *Managing Oracle WebCenter Content*.

> **Note:**
>
> WebCenter Portal supports multiple renditions for images only, not video.

In addition, the following other prerequisites must also be met:

* You must be using Oracle WebCenter Content Release 11*g*R1 (11.1.1.9.0) or later.

* WebCenter Portal and WebCenter Content must be using the same OHS front end.

* The `webContextRoot` parameter must be set in line with the common OHS front end so that WebCenter Portal can find WebCenter Content objects, for example, `/cs`.

* Single sign-on must be enabled across WebCenter Portal and WebCenter Content.

10.3.7.3 Retrieving Image Rendition Information for Image Documents

To retrieve image rendition information for image documents in your Content Presenter display template, use the following EL expression:

```
node.renditionsMap['renditionName:renditionProperty']
```

Where:

* *renditionName* is:

    – for DAM implementations, the name of the rendition from the appropriate rendition set. For example, `web`, `thumbnail`, `preview`, or `lowres`.

    – for non-DAM implementations, either `web` or `thumbnail`.

    > **Note:**
    >
    > The *renditionName* is not case sensitive, therefore `preview` refers to the same rendition as `Preview`.

* *renditionProperty* is the required rendition information.

    In most cases, you will want to retrieve the URL of the image rendition so that the rendition can be displayed on a page. To do this use the `url` property.

    You can also retrieve other information about the rendition, such as `name`, `type`, `width`, `height`, `resolution`, `size`, or `contentType`.

    > **Note:**
    >
    > For `web` and `thumbnail` renditions, only `size`, `contentType`, and `url` are applicable; `name` returns the name of the native file, not of the web or thumbnail rendition.

For example to display the `preview` rendition of an image, add the following to your display template:

```
<af:image source="#{node.renditionsMap['preview:url']}" shortDesc="Preview
rendition" id="imageContent3"/>
```

> **Note:**
>
> Wherever possible, `url` points to a static rendition URL for the image rendition on the WebCenter Content Server. If the static URL cannot be determined, the `url` uses the WebCenter Portal `showProperty` servlet.

The following example shows a Content Presenter display template that displays a carousel of images. If the carousel is displayed on a desktop device (`rendered="#{DeviceAgent.desktop}"`), then the `a200` rendition is used for the images (`node.renditionsMap['a200:url']`). If the carousel is displayed on a mobile device (`rendered="#{DeviceAgent.mobile}"`), then the smaller `thumbnail` rendition is used (`node.renditionsMap['thumbnail:url']`).

```
<?xml version='1.0' encoding='utf-8'?>
<!-- Copyright (c) 2014 Oracle and/or its affiliates.
All rights reserved. -->

<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
          xmlns:dt="http://xmlns.oracle.com/webcenter/content/templates"
          xmlns:f="http://java.sun.com/jsf/core">
  <dt:contentListTemplateDef var="nodes">
    <af:carousel id="c1"
                 value="#{nodes}"
                 var="node"
                 emptyText="#{templateBundle.EMPTY_NODES}"
                 inlineStyle="#{DeviceAgent.desktop ? '' : 'height:200px;'}">
      <f:facet name="nodeStamp">
        <af:carouselItem id="ci1"
                         text="#{node.isFolder ? node.name : (empty
node.propertyMap['dDocTitle'] ?
                               node.name :
node.propertyMap['dDocTitle'].value.stringValue)}"
                         shortDesc="#{not empty
node.propertyMap['xComments'].value.stringValue ?
                                     node.propertyMap['xComments'].value.stringValue :
                                     node.primaryProperty.value.binaryValue.name}">
          <af:image id="cimg1"
                    source="#{node.primaryProperty.isImage ?
node.renditionsMap['a200:url'] :
                             node.icon.largeIcon}"
                    shortDesc="#{node.primaryProperty.value.binaryValue.name}"
                    rendered="#{DeviceAgent.desktop}"/>
          <af:image id="cimg2"
                    source="#{node.primaryProperty.isImage ?
node.renditionsMap['thumbnail:url'] : node.icon.largeIcon}"
                    shortDesc="#{node.primaryProperty.value.binaryValue.name}"
                    rendered="#{DeviceAgent.mobile}"/>

        </af:carouselItem>
      </f:facet>
    </af:carousel>
  </dt:contentListTemplateDef>
</jsp:root>
```

10.3.7.4 Retrieving Image Renditions for Site Studio Region Definitions

The handling of image renditions for images in Site Studio region definitions is determined by way the region definitions are defined in Site Studio.

You can either have a region definition that includes a single region element that defines the base image or a region definition that includes separate region elements for each image rendition. Your Content Presenter display template must use the appropriate EL expression according to how the region definition is set up.

For example, your region definition could have a single region element (`myimage`) for the base image, therefore to display a particular image rendition, you must use an EL expression where you can specify which rendition to retrieve from Oracle WebCenter Content Server, for example, `highres` or `lowres`. Alternatively, if your region definition has separate region elements for the different image renditions, say one element (`desktop`) for a rendition suitable for desktop browsers and one (`mobile`) for a rendition more suited to mobile devices, you can use an EL expression that refers directly to the region element that defines the rendition that you want to use.

> **Tip:**
>
> If you anticipate adding further image renditions in the future, for example for new devices, it makes sense to create region definitions with single image region elements so that as new renditions are added, they can be included in your Content Presenter display templates without having to create new region elements.

- If the region definition includes a single region element that defines the base image, you can also specify which rendition of the image you want to use:

  ```
  node.propertyMap['regionDefinitionName:elementName/
  Rendition:renditionName'].asTextHtml
  ```

- To reference a region element within a region definition that defines a specific image rendition, use the following EL expression:

  ```
  node.propertyMap['regionDefinitionName:elementName'].asTextHtml
  ```

Where:

- *regionDefinitionName* is the name of the region definition.

- *elementName* is the name of the region element.

- *renditionName* is:

  - for DAM implementations, the name of the rendition from the appropriate rendition set. For example, `web`, `thumbnail`, `highres` or `lowres`.

  - for non-DAM implementations, either `web` or `thumbnail`.

  > **Note:**
  >
  > The *renditionName* is not case sensitive, therefore `preview` refers to the same rendition as `Preview`.

The EL expressions return an HTML `img` tag with a `src` element that points to the URL of the appropriate image rendition.

> **Note:**
>
> Wherever possible, the `src` URL points to a static rendition URL for the image rendition on WebCenter Content Server. If the static URL cannot be determined, the `url` uses the WebCenter Portal `showProperty` servlet.

Figure 10-8 shows a Site Studio region definition (`RD_REUSEIMG`) for an article. The definition includes region elements for the title, summary, body and image of the article. The single image region element defines the original image (`Image`).

Figure 10-8   Region Definition with Single Image Region Element
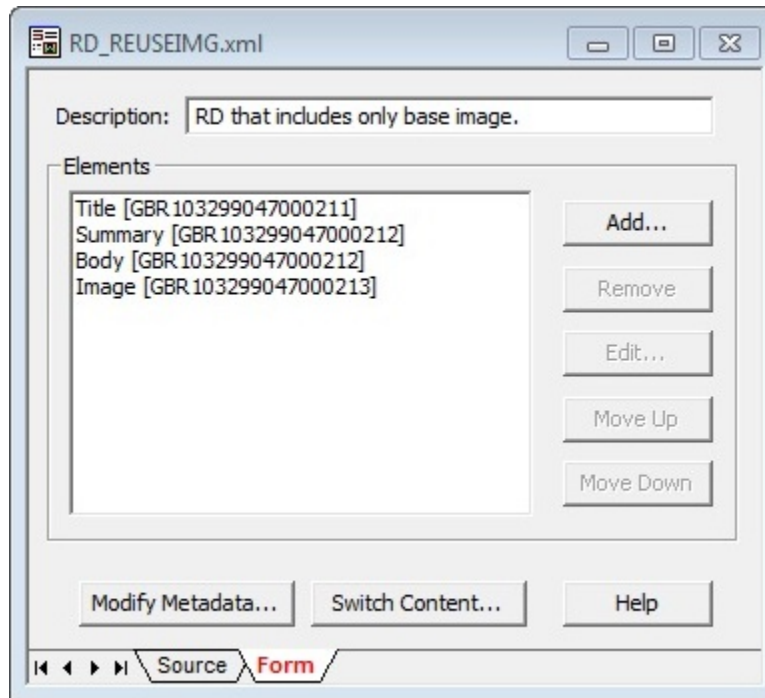
To use a lower resolution rendition (`lowres`) of the image on a mobile device, use the following code:

```
<af:outputText value="#{node.propertyMap['RD_REUSEIMG:Image/
Rendition:lowres'].asTextHtml}" escape="false" id="olmaget"
rendered="#{DeviceAgent.mobile}"/>
```

> **Note:**
>
> If the specified `renditionName` does not exist for the image, the image's primary rendition is used instead.

Figure 10-9 shows a different region definition (`RD_NAMEDREND`) for a different article that includes separate region elements for the different possible renditions of the article image: `BaseImage`, `ThumbnailImage`, `A100Image`, and `A200Image`.

Figure 10-9   Region Definition with Region Elements for Named Renditions
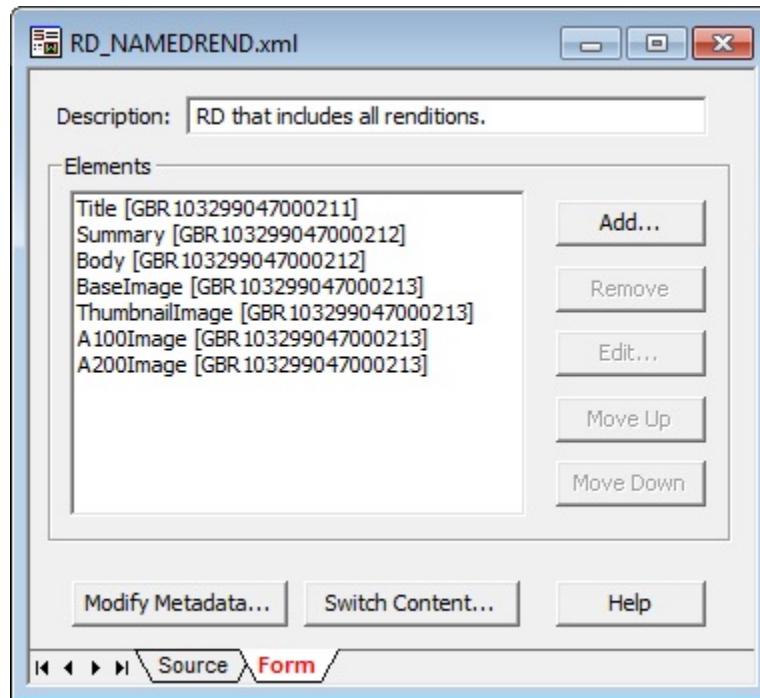
To use the `ThumbnailImage` region element to render the image on a mobile device, use the following code:

```
<af:outputText value="#{node.propertyMap['RD_NAMEDREND:ThumbnailImage'].asTextHtml}"
escape="false" id="olmaget" rendered="#{DeviceAgent.mobile}"/>
```

> **Note:**
>
> If you use an EL expression with a specified *renditionName* for an region element that explicitly references a particular image rendition (rather than the base image), the rendition defined by the region element is returned (not the rendition specified by *renditionName*).

Within a WYSIWYG or static list region element you can reference a specific rendition of an image using the
`node.propertyMap['`*regionDefinitionName*`:`*elementName*`']` EL expression.
Alternatively, you can reference base images in the WYSIWG or static list region elements and then use the
`node.propertyMap['`*regionDefinitionName*`:`*elementName*`/`
`Rendition:`*renditionName*`']` EL expression to use a specific rendition for all images within the element.

The following example shows how a particular rendition (`Preview`) is used for images within a static list region element (`paragraph`).

```
<af:iterator value="#{node.propertyMap['RD_MYREGION:paragraph/
Rendition:Preview'].values}"
             var="row"
             id="i1"
             rendered="#{DeviceAgent.desktop}">
  <af:showDetailItem text="#{row.nestedValue[0].value}" id="sdi1"
stretchChildren="first">
```

```
        <af:panelGroupLayout layout="horizontal" id="tt" inlineStyle="padding:5px;">
          <af:outputText escape="false" value="#{row.nestedValue[1].value}" id="ld1"/>
          <af:outputText escape="false" value="#{row.nestedValue[2].value}" id="ld2"/>
        </af:panelGroupLayout>
      </af:showDetailItem>
</af:iterator>
```

In the following example RD_MYREGION:body corresponds to a WYSIWYG region element. All images within that WYSIWYG element will be rendered using the A100 rendition.

```
<af:outputText value="#{node.propertyMap['RD_MYREGION:body/
Rendition:A100'].asTextHtml}"
                escape="false" id="ot5"/>
```

> **Note:**
>
> If the WYSIWYG or static list element explicitly references a particular rendition of an image, that rendition supersedes any rendition specified using the Rendition:*renditionName* syntax.

10.3.8 How to Use EL Expressions to Retrieve Content Item Information

This section describes the EL expressions that you can use in your Content Presenter display template definitions to retrieve and display specific information about content items.

Use the EL expressions described in the following tables as you define your Content Presenter display templates as explained in How to Define a Single-Item Display Template and How to Define a Multiple-Item Display Template. These expressions are used with the JSP tags described in Table 10-1 and Table 10-2.

This section contains the following topics:

- How to Retrieve Basic Information About a Content Item

- How to Work with Content Item Properties and Values

- How to Work with Content Item Icons and URLs

- How to Work with Image Renditions

- How to Work with Group Portal Information

10.3.8.1 How to Retrieve Basic Information About a Content Item

The EL expressions listed in this section enable you to display basic information about a content item in a display template.

Table 10-3   EL Expressions for Retrieving Basic Content Information

EL Expression	Description
#{node.createdBy}	Returns the user name of the node's creator.
#{node.createdDate}	Returns the node's creation date.
#{node.hasParentNode}	Returns true if the current node has a valid parent node ID, or a node that has no parent.

Table 10-3   (Cont.) EL Expressions for Retrieving Basic Content Information

EL Expression	Description
#{node.icon}	Returns the icon service defined in the current web application.
#{node.id}	Returns the node's identifier.
#{node.isFolder}	Returns true if this node is associated with a folder or container.
#{node.isInherited}	Returns true if this node is inherited by another object class definition.
#{node.modifiedBy}	Returns the user name of the node's last modifier.
#{node.modifiedDate}	Returns the node's last modification date.
#{node.name}	Returns the node's name.
#{node.parentId}	Returns the parent node's identifier.
#{node.path}	Returns the node's path.
#{node.primaryProperty}	Returns the node's primary property, if available.
#{node.propertyMap}	Creates and returns a map of wrapped property objects, keyed by property name. Properties can be accessed as #{node.propertyMap['myProp']} or #{node.propertyMap.myProp}.
#{node.url}	Returns an instance of the node property URL service for the primary property of this node (if any). By default, it resolves to #{node.url.renderUrl}. This is a shortcut for #{node.primaryProperty.url}.

10.3.8.2 How to Work with Content Item Properties and Values

Use the EL expressions described in this section to perform actions on content item node properties and property values.

Tip:

To determine the names of the properties defined for a given content type, see How to Discover Content Type Property Names.

Table 10-4   EL Expressions for Content Item Node Properties

EL Expression	Description
#{node.propertyMap['myProp'].asTextHtml}	Returns this property as text or HTML if the type is text or HTML. If isHTML or isPlainText is true, the text or HTML is returned as a string.
#{node.propertyMap['myProp'].hasValue}	Returns true if a value is associated with this property.

Table 10-4   (Cont.) EL Expressions for Content Item Node Properties

EL Expression	Description
#{node.propertyMap['myProp'].icon}	Returns the icon service defined in the current web application.
#{node.propertyMap['myProp'].indexedName}	Returns the indexed name of a multi-valued property. For example, if a multi-valued node property named color contains blue, red, orange, the indexed name of the red value is color[1]. The following EL expression references the color orange in this list: #{node.propertyMap['color[2]'].value.stringValue}
#{node.propertyMap['myProp'].isAudio}	Returns true if the property's mime type is 'audio/'.
#{node.propertyMap['myProp'].isBinary}	Returns true if the current property is of type Property.BINARY.
#{node.propertyMap['myProp'].isBoolean}	Returns true if the current property is of type Property.BOOLEAN.
#{node.propertyMap['myProp'].isCalendar}	Returns true if the current property is of type Property.CALENDAR.
#{node.propertyMap['myProp'].isDouble}	Returns true if the current property is of type Property.DOUBLE.
#{node.propertyMap['myProp'].isExcel}	Returns true if the property's mime type is 'application/vnd.ms-excel', 'application/excel', 'application/x-excel', or 'application/x-msexcel'.
#{node.propertyMap['myProp'].isGIF}	Returns true if the property's mime type is 'image/gif'.
#{node.propertyMap['myProp'].isHTML}	Returns true if the property's mime type is 'text/html'.
#{node.propertyMap['myProp'].isImage}	Returns true if the property's mime type is 'image/'.
#{node.propertyMap['myProp'].isJPEG}	Returns true if the property's mime type is 'image/jpeg'.
#{node.propertyMap['myProp'].isLink}	Returns true if the current property is of type Property.LINK.
#{node.propertyMap['myProp'].isLong}	Returns true if the current property is of type Property.LONG.
#{node.propertyMap['myProp'].isMSWord}	Returns true if the property's mime type is 'application/vnd.ms-word' or 'application/msword'.
#{node.propertyMap['myProp'].isMultiValued}	Returns true if this property is multi-valued.
#{node.propertyMap['myProp'].isNested}	Returns true if the current property is of type Property.NESTED.

Table 10-4   (Cont.) EL Expressions for Content Item Node Properties

EL Expression	Description
`#{node.propertyMap['myProp'].isPDF}`	Returns `true` if the property's mime type is `'application/pdf'`.
`#{node.propertyMap['myProp'].isPlainText}`	Returns `true` if the property's mime type is `'text/plain'`.
`#{node.propertyMap['myProp'].isPNG}`	Returns `true` if the property's mime type is `'image/png'`.
`#{node.propertyMap['myProp'].isPowerPoint}`	Returns `true` if the property's mime type is `'application/vnd.ms-powerpoint'`, `'application/mspowerpoint'`, or `'application/x-mspowerpoint'`.
`#{node.propertyMap['myProp'].isPrimaryProperty}`	Returns `true` if this property is the primary property.
`#{node.propertyMap['myProp'].isRequired}`	Returns `true` if this property is required/mandatory.
`#{node.propertyMap['myProp'].isRetricted}`	Returns `true` if this property is restricted.
`#{node.propertyMap['myProp'].isRichText}`	Returns `true` if the property's mime type is `'text/richtext'`.
`#{node.propertyMap['myProp'].isString}`	Returns `true` if the current property is of type `Property.STRING`.
`#{node.propertyMap['myProp'].isTextBased}`	Returns `true` if this property is text-based (`isHTML`, `isPlainText`, `isString`, `isCalendar`, `isBoolean`, `isDouble`, `isLong`).
`#{node.propertyMap['myProp'].isVideo}`	Returns `true` if the property's mime type is `'video/'`.
`#{node.propertyMap['myProp'].isXML}`	Returns `true` if the property's mime type is `'text/xml'`.
`#{node.propertyMap['myProp'].isZip}`	Returns `true` if the property's mime type is `'application/zip'`.
`#{node.propertyMap['myRefNode'].linkAsNode}`	Returns the `myRefNode` property as a node, where `myRefNode` is a link property type. Properties can be referenced in EL expressions. **Example:**`#{node.propertyMap['myRefNode'].linkAsNode.primaryProperty.url.renderUrl}`
`#{node.propertyMap['myProp'].name}`	Returns the property's name.
`#{node.propertyMap['myProp'].nestedProperty}`	Retrieves nested properties for this single-valued property, and returns a list of properties.
`#{node.propertyMap['myProp'].type}`	Returns the data type of this property value. For example: `String`, `Integer`, `Long`, and so on.
`#{node.propertyMap['myProp'].url}`	Returns a URL service for this property.
`#{node.propertyMap['myProp'].value}`	Returns the value service for this property.

Table 10-4 (Cont.) EL Expressions for Content Item Node Properties

EL Expression	Description
#{node.propertyMap['myProp'].nestedProperties}	Returns elements from a static list. For example: ``` <af:iterator var="listItem" value="#{node.propertyMap['ARTICLE_RGD:Paragraphs'].nestedProperties}" varStatus="vs"> <af:outputText id="ot1" value='#{listItem[0].value}'/> <af:outputText id="ot3" value="#{listItem[1].value}"/> </af:iterator> ```
#node.propertyMap['regionDefName:elementName'].asTextHtml}	Returns Site Studio data as HTML text. For example: ``` #node.propertyMap['RD_NEWS:LEAD'].asTextHtml} ``` The element name (LEAD) is prefixed by the Region Definition name (RD_NEWS). See also How to Reference Site Studio Region Elements in a Custom View.

Table 10-5 EL Expressions for Content Item Node Property Values

EL Expression	Description
#{node.propertyMap['myProp'].value.binaryValue}	Returns custom attributes for a binary property type or attachment. Attributes available on binaryValue are: • contentType –Returns the mime type of the binary content (for example, text or html). • name –Returns the filename of the binary content (for example, index.html). • size –Returns the size of the binary content, or -1 if the size is unavailable.
#{node.propertyMap['myProp'].value.booleanValue}	Returns the value of this property as java.lang.Boolean.
#{node.propertyMap['myProp'].value.calendarValue}	Returns the value of this property as java.util.Calendar.
#{node.propertyMap['myProp'].value.doubleValue}	Returns the value of this property as java.lang.Double.
#{node.propertyMap['myProp'].value.longValue}	Returns the value of this property as java.lang.Long.
#{node.propertyMap['myProp'].value.orderedPosition}	Returns the "index" of the property when the property is multi-valued. **Example**:#{node.propertyMap['address[0]'].value.orderedPosition},

Table 10-5   (Cont.) EL Expressions for Content Item Node Property Values

EL Expression	Description
`#{node.propertyMap['myProp'].value.string Value}`	Returns the value of this property as `java.lang.String`.
	Example:`#{node.propertyMap['firstName'].value.stringValue}`

10.3.8.3 How to Work with Content Item Icons and URLs

Use the the EL expressions described in this section to work with icons and URLs associated with content items and properties.

Table 10-6   EL Expressions for Content Item Node or Property Icons

EL Expression	Description
`#{node.icon.largeIcon}`	Returns a URL to an image resource for a large icon.
`#{node.propertyMap['myDoc'].icon.largeIcon}`	**Example**: `<af:image source="#{node.propertyMap['projectFolder'].largeIcon}"/>`
`#{node.icon.smallIcon}`	Returns a URL to an image resource for a small icon.
`#{node.propertyMap['myDoc'].icon.smallIcon}`	**Example**: `<af:image source="#{node.icon.smallIcon}" />`

Table 10-7   EL Expressions for Content Item Node URLs

EL Expression	Description
`#{node.url.downloadUrl}`	Creates a URL to the binary content. Forces a download, and the underlying operating system renders the content based on the content type.
	Example: `<af:goLink destination="#{node.url.downloadUrl}" targetFrame="_blank"/>`
`#{node.url.renderUrl}`	Creates a URL to the binary content. Allows the browser to render the content based on the content type.
	By default, `#{node.url}` resolves to `#{node.url.renderUrl}`.
	Example: `<af:goLink destination="#{node.url.renderUrl}" targetFrame="_blank"/>`

10.3.8.4 How to Work with Image Renditions

This section lists the EL expressions available to retrieve information about specific renditions of image documents and Site Studio region elements, including the URL to render the image using that rendition.

For more information about image renditions, see How to Use Image Renditions in Content Presenter Display Templates.

Table 10-8   EL Expressions for Image Renditions of Image Documents

EL Expression	Description

`#{node.renditionsMap['renditionName:url']}`	Returns the URL of the image rendition. For example:
	
	`http://mymachine.example.com:8888/cs/groups/`
	`personalspaces/@pewebcenter/`
	`@799c4d7d-255c-46c8-80f5-0d06c848dd65/documents/`
	`document/awrf/mdax/~edisp/~extract/`
	`ID_001642~3~staticrendition/preview.gif`
	
	Wherever possible, `url` points to a static rendition URL for the image rendition on Oracle WebCenter Content Server. If the static URL cannot be determined, the `url` uses the WebCenter Portal `showProperty` servlet.

`#{node.renditionsMap['renditionName:name']}`	Returns the name of the rendition, for example `web`, `thumbnail`, or `highres`. For `web` and `thumbnail` renditions, `name` returns the name of the native file, not of the web or thumbnail rendition.
`#{node.renditionsMap['renditionName:type']}`	Returns the file type of the image rendition, for example, `preview`. The `type` property is not valid for `web` and `thumbnail` renditions.
`#{node.renditionsMap['renditionName:width']}`	Returns the width of the image rendition in pixels, for example 250. The `width` property is not valid for `web` and `thumbnail` renditions.
`#{node.renditionsMap['renditionName:height']}`	Returns the height of the image rendition in pixels, for example 34. The `height` property is not valid for `web` and `thumbnail` renditions.
`#{node.renditionsMap['renditionName:resolution']}`	Returns the resolution (DPI) of the image rendition, for example `96 dpi`. The `resolution` property is not valid for `web` and `thumbnail` renditions.
`#{node.renditionsMap['renditionName:size']}`	Returns the file size of the image rendition, for example, 3133.
`#{node.renditionsMap['renditionName:contentType']}`	Returns the MIME type of the image rendition, for example. `image/gif`.

Table 10-9   EL Expressions for Image Renditions of Site Studio Region Elements

Table 10-9   (Cont.) EL Expressions for Image Renditions of Site Studio Region Elements

EL Expression	Description

`#{node.propertyMap['`*regionDefinitionName*`:`*elementName*`'].asTextHtml}`	Returns an HTML `img` tag with the `src` element set to the URL of the image defined in the specified element. For example:
	
	`<img src="http://mymachine.example.com:9400`
	`/cs/idcplg?IdcService=GET_FILE&dDocName=`
	`id_038497&RevisionSelectionMethod=LatestRelea`
	`sed"`
	`alt="S3-1" class="">`
	
	Use this EL to reference a Site Studio region element that defines a specific rendition of an image.
	
	Wherever possible, the URL points to a static rendition URL for the image rendition on WebCenter Content Server. If the static URL cannot be determined, the `url` uses the WebCenter Portal `showProperty` servlet.

`#{node.propertyMap['`*regionDefinitionName*`:`*elementName*`/`
`Rendition:`*renditionName*`'].asTextHtml}`
	Returns an HTML `img` tag with the `src` element set to the URL of the specified image rendition of the image defined in the specified element. For example:
	
	`<img src="/cs/groups/wc081512c/`
	`@sb5cdcaa4610341a8bb8387effdf21790/documents/`
	`document/`
	`awrf/mdm4/~edisp/~extract/`
	`ID_038497~1~staticrendition/preview.gif"`
	`alt="S3-1"`
	`class="">`
	
	If the specified image rendition does not exist for the image, then the base image URL is returned.
	
	Use this EL to reference a Site Studio region element that defines the base image for which you want to display the specified *renditionName*.
	
	Wherever possible, the URL points to a static rendition URL for the image rendition on WebCenter Content Server. If the static URL cannot be determined, the `url` uses the WebCenter Portal `showProperty` servlet.

10.3.8.5 How to Work with Group Portal Information

This section lists the EL expressions you can use to work with group portal information in your Content Presenter display templates.

Table 10-10   EL Expressions for Basic Group Portal Information

EL Expression	Description

`#{spaceContext.currentSpace.metadata.crea`
`tedBy}`
`#{spaceContext.space[portalName].metadata`
`.createdBy}`
	Returns the user who created the current or specified group portal.

Table 10-10   (Cont.) EL Expressions for Basic Group Portal Information

EL Expression	Description

`#{spaceContext.currentSpace.metadata.creationDate}`
 	
`#{spaceContext.space[portalName].metadata.creationDate}`
	Returns a `java.util.Calendar` object representing the date and time on which the current or specified portal was created.

`#{spaceContext.currentSpace.metadata.customAttributes[attributeName]}`
 	
`#{spaceContext.space[portalName].metadata.customAttributes[attributeName]}`
	Returns the value of a specific custom attribute of the name `attributeName` for the portal.

`#{spaceContext.currentSpace.metadata.defaultLanguage}`
 	
`#{spaceContext.space[portalName].metadata.defaultLanguage}`
	Returns the default language for the current or specified portal.

`#{spaceContext.currentSpace.metadata.description}`
 	
`#{spaceContext.space[portalName].metadata.description}`
	Returns the description associated with the current portal with the display name in the language in which the portal was created. If the portal name has been translated, the translated name is not shown.
	
	Example:
	`#{spaceContext.space['FinanceProject'].metadata.description}`
	
	Evaluates to conglomeration of all teams involved in financial activities.

`#{spaceContext.currentSpace.metadata.displayName}`
 	
`#{spaceContext.space[portalName].metadata.displayName}`
	Returns the display name associated with the current or specified portal in the language in which the portal was created. If the portal name has been translated, the name in which the portal was created will be shown.
	
	Example:
	If a group portal called "Web20Portal" has the display name "Web 2.0 Portal", then:
	`#{spaceContext.space['Web20Portal'].metadata.displayName}`
	
	will evaluate to "Web 2.0 Portal".

`#{spaceContext.currentSpace.metadata.guid}`
 	
`#{spaceContext.space[portalName].metadata.guid}`
	Returns the unique ID associated with the current or specified portal.

`#{spaceContext.space[portalName].metadata.icon}`
 	
`#{spaceContext.currentSpace.metadata.icon}`
	Returns a URL to the icon associated with the current or specified portal.

`#{spaceContext.currentSpace.metadata.keywords}`
 	
`#{spaceContext.space[portalName].metadata.keywords}`
	Returns a comma-separated list of searchable keywords associated with the current or specified portal.

Table 10-10   (Cont.) EL Expressions for Basic Group Portal Information

EL Expression	Description

`#{spaceContext.currentSpace.metadata.last UpdatedDate}`
 	
`#{spaceContext.space[portalName].metadata .lastUpdatedDate}`
	Returns the `java.util.Calendar` object representing the date-time on which the current or specified portal was last updated.

`#{spaceContext.space[portalName].metadata .logo}`
 	
`#{spaceContext.currentSpace.metadata.logo }`
	Returns the path to the image associated with the logo of the current or specified portal.

`#{spaceContext.currentSpace.metadata.name }`
 	
`#{spaceContext.space[portalName].metadata .name}`
	Returns the name of the current or specified portal used generally at the back end of the portal and is used with the pretty URL.

`#{spaceContext.currentSpace.metadata.grou pSpaceURI}`
 	
`#{spaceContext.space[portalName].metadata .groupSpaceURI}`
	Returns the pretty URL of the current or specified portal.

`#{spaceContext.currentSpace.metadata.clos ed}`
 	
`#{spaceContext.space[portalName].metadata .closed}`
	Returns Boolean value indicating whether the portal has been kept closed.

`#{spaceContext.currentSpace.metadata.disc overable}`
 	
`#{spaceContext.space[portalName].metadata .discoverable}`
	Returns Boolean value indicating whether users will be able to discover the existence of the portal by searching for it or getting it listed in My Portals.

`#{spaceContext.currentSpace.metadata.offl ine}`
 	
`#{spaceContext.space[portalName].metadata .offline}`
	Returns Boolean value indicating whether the portal has been taken offline.

`#{spaceContext.currentSpace.metadata.publ ishRSS}`
 	
`#{spaceContext.space[portalName].metadata .publishRSS}`
	Returns Boolean value, indicating whether the portal publishes RSS feeds.

`#{spaceContext.currentSpace.metadata.self Registration}`
 	
`#{spaceContext.space[portalName].metadata .selfRegistration}`
	Returns a Boolean value indicating whether users are allowed to subscribe themselves to the portal.

`#{spaceContext.currentSpace.metadata.unsu bscriptionApprovalRequired}`
 	
`#{spaceContext.space[portalName].metadata .unsubscriptionApprovalRequired}`
	Returns a Boolean value representing whether approval is required to unsubscribe from the current or specified portal.

Table 10-11   EL Expressions for Portal UI Information

Table 10-11   (Cont.) EL Expressions for Portal UI Information

EL Expression	Description

`#{spaceContext.space[portalName].metadata.uiMetadata.gsSiteTemplateId}`
 	
`#{spaceContext.currentSpace.metadata.uiMetadata.gsSiteTemplateId}`
	Returns the ID of the page template associated with the current or specified portal.

`#{spaceContext.space[portalName].metadata.uiMetadata.rcForGSPages}`
 	
`#{spaceContext.currentSpace.metadata.uiMetadata.rcForGSPages}`
	Returns the ID of the resource catalog associated with the current or specified portal.

`#{spaceContext.space[portalName].metadata.uiMetadata.rcForGSSiteTemplates}`
 	
`#{spaceContext.currentSpace.metadata.uiMetadata.rcForGSSiteTemplates}`
	Returns the ID of the resource catalog associated with the page template of the current portal.

`#{spaceContext.space[portalName].metadata.uiMetadata.gsSiteStructureId}`
 	
`#{spaceContext.currentSpace.metadata.uiMetadata.gsSiteStructureId}`
	Returns the ID of the navigation model associated with the current portal.

`#{spaceContext.space[portalName].metadata.uiMetadata.skin}`
 	
`#{spaceContext.currentSpace.metadata.uiMetadata.skin}`
	Returns the ADF Faces skin family associated with the current portal.

`#{spaceContext.space[portalName].metadata.uiMetadata.footerHidden}`
 	
`#{spaceContext.currentSpace.metadata.uiMetadata.footerHidden}`
	Returns the Boolean value representing whether the footer of the portal is hidden.

`#{spaceContext.space[portalName].metadata.uiMetadata.copyrightMessage}`
 	
`#{spaceContext.currentSpace.metadata.uiMetadata.copyrightMessage}`
	Returns the copyright message used by the portal.

`#{spaceContext.space[portalName].metadata.uiMetadata.privacyPolicyUrl}`
 	
`#{spaceContext.currentSpace.metadata.uiMetadata.privacyPolicyUrl}`
	Returns the URL to the privacy policy document followed.

10.3.9 How to Discover Content Type Property Names

As a Content Presenter display template developer, you will need to know the names of the properties defined for the associated content type so that you can define how to display the selected content item(s) on the page.

Each content item is associated with a specific *content type* defined in the Oracle WebCenter Content Server repository. Content types can map to WebCenter Content Server profile definitions and Site Studio region definitions. Types are created in WebCenter Content Server and define the properties of the content item. The property names of a content item's content type, however, are different than the display names, and need to be obtained from WebCenter Content Server. For more information, see Defining Content Types in *Managing Oracle WebCenter Content*.

> **Note:**
>
> You can also use REST services to obtain content type property names. For more information see Using the WebCenter Portal REST APIs.

10.3.10 How to Reference External Files in Display Templates

In some cases, a display template needs to reference an external file, like a CSS file. All such references must be either an absolute path or a path that is relative to the root of the web application.

For example:

- **absolute path** –`http://host:port/mypath/file.css`

- **relative path** –`/webcenter/mypath/file.css`

Do not use local references to external files. Local references to external files do not work because they are not included when you publish a Content Presenter display template as an asset, as explained in Publishing a Content Presenter Display Template.

10.3.11 How to Reference Site Studio Region Elements in a Custom View

You can use custom display templates to display Site Studio Region Definition elements.

For example, you might have a Site Studio Region Definition called `RD_NEWS` with four elements: `TITLE`, `LEAD`, `IMAGE`, and `BODY`. A Content Presenter display template can reference these elements using the node property EL expression like this:

`#node.propertyMap['RD_NEWS:LEAD'].asTextHtml}`

The following example illustrates how these Site Studio Region elements can be included in a `contentTemplateDef` definition:

```
<dt:contentTemplateDef var="node">
   <af:panelGroupLayout layout="vertical" id="pgl3">
     <af:panelGroupLayout layout="horizontal" valign="top" inlineStyle="background-
color:#FFF; padding:10px;" id="pgl4">
         <af:panelGroupLayout layout="vertical" id="pgl2" valign="top">
           <af:outputText
value="#{node.propertyMap['dInDate'].value.calendarValue}" id="ot3"
styleClass="bodytext" converter="javax.faces.DateTime"/>
         </af:panelGroupLayout>
         <af:spacer width="10px;" id="s1" inlineStyle="background-color:#DDD;
color:white;"/>
         <af:panelGroupLayout layout="vertical" id="pgl1" valign="top">
           <af:outputText value="#{node.propertyMap['xTargetGroup'].value}"
id="ot12" inlineStyle="background-color:#0A9FC0; color:white; text-align:left;
padding:5px;"/>
           <af:goLink text="#{node.propertyMap['RD_NEWS:TITLE'].asTextHtml}"
id="gil1"
               destination="#{'/faces/home/news-viewer?
news_id='}#{node.propertyMap['dDocName'].value}" styleClass="newstitle"/>
           <af:outputText value="#{node.propertyMap['RD_NEWS:LEAD'].asTextHtml}"
id="ot2" styleClass="bodytext"/>
         </af:panelGroupLayout>
       <af:panelGroupLayout layout="vertical" id="pgl32" valign="top"
styleClass="newsimage">
         <af:outputText value="#{node.propertyMap['RD_NEWS:IMAGE'].asTextHtml}"
```

```
                escape="false" id="ot1"  inlineStyle="max-width:100px;"/>
            </af:panelGroupLayout>
        </af:panelGroupLayout>
        <af:panelGroupLayout layout="horizontal" id="aaaa">
        </af:panelGroupLayout>
    </af:panelGroupLayout>
</dt:contentTemplateDef>
```

For a complete, end-to-end example illustrating how to reference Site Studio Region elements in multiple templates, see the WebCenter Architecture Team blog entry at:

http://www.ateam-oracle.com/content-presenter-cmis-complete

10.4 Publishing a Content Presenter Display Template

After creating a Content Presenter display template and editing the page fragment, the next step is to publish and test the template in WebCenter Portal.

For instructions on how to publish a Content Presenter display template as a shared asset, or to a specific portal as a portal asset, see Publishing WebCenter Portal Assets.

10.5 Optimizing Performance for Content Presenter Display Templates

When content nodes are retrieved for display in a Content Presenter display template, most content item node property values are retrieved immediately along with the node, but some are loaded later only if needed. Other than the performance difference, the selective loading of node property values is transparent to the user or developer.

As a Content Presenter display template developer, you can optimize performance of your template if you are aware when different property values are loaded. For example, a typical list display template will render faster if you refer only to properties that are immediately loaded when the node is first retrieved and avoid properties that are loaded later when needed.

A secondary consideration is dependent on how the node is retrieved: through search versus fetched by parent ID. A property that may be loaded immediately on node retrieval for searches (such as Results of a Query) may be loaded later for other retrieval methods (such as Contents Under a Folder). Table 10-12 shows whether or not node properties are loaded immediately upon node retrieval, by retrieval mechanism.

For information about the node properties, see Configuration Reference for Oracle WebCenter Content.

Table 10-12   Loading of Node Properties By Node Retrieval Mechanism

OCS Global Profile Properties	Loaded When Node Is First Retrieved?		
	GET BY PARENT ID (Contents Under a Folder)	SEARCH (Results of a Query)	GET BY UUID (Single Content Item and List of Items)
VaultFileSize	**N**	Y	Y
dCheckoutUser	Y	**N**	Y
dCreateDate	Y	Y	Y
dDocAccount	Y	Y	Y
dDocAuthor	Y	Y	Y

Table 10-12   (Cont.) Loading of Node Properties By Node Retrieval Mechanism

OCS Global Profile Properties	Loaded When Node Is First Retrieved?		
	GET BY PARENT ID (Contents Under a Folder)	SEARCH (Results of a Query)	GET BY UUID (Single Content Item and List of Items)
dDocName	Y	Y	Y
dDocTitle	Y	Y	Y
dDocType	Y	Y	Y
dFormat	Y	Y	Y
dID	Y	Y	Y
dInDate	Y	Y	Y
dIsCheckedOut	Y	**N**	Y
dOutDate	Y	Y	Y
dReleaseDate	Y	**N**	Y
dReleaseState	Y	**N**	Y
dRevClassID	Y	**N**	Y
dRevLabel	Y	Y	Y
dRevRank	Y	**N**	Y
dRevisionID	Y	Y	Y
dSecurityGroup	Y	Y	Y
dStatus	Y	**N**	Y
dWebExtension	Y	Y	Y
dWorkflowState	**N**	**N**	Y
idcPrimaryFile	Y	Y	Y
idcRenditions	**N**	**N**	Y
xCollectionID	Y	Y	Y
xComments	Y	Y	Y
xForceFolderSecurity	Y	Y	Y
xHidden	Y	Y	Y
xInihibitUpdate	Y	Y	Y
xReadOnly	Y	Y	Y

10.6 Using Content Presenter (Tips, Tutorials, and Examples)

The following supplementary tutorials and examples further illustrate how Content Presenter can be used.

- *UCM, Site Studio and Templates* - describes the components (Content Server, Site Studio, and Content Presenter) that drive content interactions in WebCenter Portal.

  http://www.ateam-oracle.com/portal-and-content-components-part-1-ucm-site-studio-and-templates-2-of-7

- *Content Presenter-CMIS-Complete* - steps you through using Content Presenter-based pages and complex CMIS and custom templates to create a filterable content list viewer.

  http://www.ateam-oracle.com/content-presenter-cmis-complete

- *Portal and Content - Content Integration - Best Practices* - highlights some of the best practices to consider when integrating content into your portal.

  http://www.ateam-oracle.com/portal-and-content-content-integration-best-practices/

Part III

Working with Portlets

This part contains the following chapters:

- Introduction to Portlets

- Creating Portlets from JSF Applications Using the Oracle JSF Portlet Bridge

- Building Standards-Based Java Portlets Using JSR 286

11

Introduction to Portlets

This chapter provides an overview of portlets and describes their uses. It explains portlet anatomy and the resources you can use 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 topics:

- About Portlets

- About Portlet Anatomy

- About Portlet Resources

- About Portlet Development

11.1 About Portlets

A portlet is a reusable web component that can draw content from many different sources. Portlets can contain anything from static HTML content to Java controls to complex web services and process-heavy applications.

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 several different portlets can be placed on a single page, users benefit from a single-source experience even though, in reality, the content may be derived from multiple sources.

Figure 11-1   Lottery Sample Portlet

Portlets can communicate with each other using events and other techniques. A single portlet can also have multiple instances—in other words, it can appear on a variety of different pages within a single portal, or even across multiple portals. Page designers can customize portlets to meet the needs of their specific audience. With the correct privileges, individual end users can further personalize portlets for their own particular requirements.

WebCenter Portal supports the development of portlets using JSR 286, an industry standard. You can also create portlets from existing JSF applications using the Oracle JSF Portlet Bridge. For more information, see About Portlet Resources.

11.2 About Portlet Anatomy

Portlet anatomy is the visual representation of the portlet on a page.

Figure 11-2 shows some aspects of portlet anatomy that you might expect to see in a typical portlet in WebCenter Portal. Note that the same portlet displayed in a different application could look different.

Figure 11-2   Sample Portlet Showing Typical Portlet Anatomy

Aspects of portlet anatomy (some, but not all, of which are illustrated in Figure 11-2) include:

- **Portlet chrome**—A collective term for the different types of decoration that surround the portlet, including the header, shadow, resize handle, Actions menu, and icons.

- **Portlet header**—The area of the portlet that displays the portlet icon, title, **Actions** menu, and **Customize** and **Personalize** icons.

- **Portlet icon**—A small image in the portlet header used to visually identify the portlet.

- **Portlet title**—Text in the portlet header that indicates the purpose of the portlet. End users may be able to personalize this title at runtime to make it more meaningful to their particular usage.

- **Personalize icon**—An icon in the portlet header that enables end users to make changes to the portlet that only they can see.

  The **Personalize** icon is displayed only to authenticated users (that is, users who are logged in). It does not display to public users or unauthenticated users. You must implement some form of application security for users to be able to personalize their portlet views.

- **Customize icon**—An icon in the portlet header that enables application administrators to make changes to the default settings of the portlet at runtime. These changes are visible to all users.

  A typical customization setting is the portlet title. At runtime, the application administrator can determine what title should appear in the portlet header.

- **Actions icon**—An icon in the portlet header that displays the **Actions** menu when clicked.

- **Actions menu**—A menu that lists various actions that users can perform on the portlet. The actions listed depend on whether the user is logged in, what privileges that user has, the functionality provided by the portlet, and the options specified when the portlet was added to the page at design time. Actions include **Move**, **Remove**, **Refresh**, **Move Up**, and **Move Down**. The **Actions** menu also provides access to other portlet modes available for the portlet, such as About, Help, Configure, and Print.

  If the user who added the portlet to the page chose to not display the portlet header, the **Actions** menu is displayed on a fade in/fade out toolbar that displays on mouse rollover.

- **Resize handle**—An area at the bottom right of the portlet that enables users to change the size of the portlet.

- **Scroll bars**—Display when the portlet content does not fit the width and height specified for the portlet, providing access the content that is not initially displayed.

- **Portlet content**—The actual content of the portlet as determined by the portlet logic.

The portlet anatomy rendered on the page is controlled by two factors:

- The portlet's own logic, as determined by the portlet developer. This includes which portlet modes are supported by the portlet.

- The attributes of the portlet tag that binds the portlet to the page, as determined by the application developer who added the portlet to the page at design time.

For example, when designing the portlet, the portlet developer may implement Help mode for the portlet. When a page designer adds the portlet to the page, he or she can determine, using the portlet property `isHelpModeAvailable`, whether or not to include the **Help** command in the portlet's **Actions** menu. If the page designer sets `isHelpModeAvailable` to false, the **Help** command is not included in the **Actions** menu even though it is provided in the portlet logic. Conversely, if the portlet developer has not implemented Help mode for a portlet, the **Help** command is not displayed in the **Actions** menu, even if `isHelpModeAvailable` is set to `true`.

11.3 About Portlet Resources

The portlets in your applications can come from a variety of sources, including other Oracle products and third-party vendors.

Portlet resources also include custom built Java portlets built using WebCenter Portal's Java portlet wizard for JSR 286. Each of these resources offers different product features and are targeted toward users with different levels of experience.

This section includes the following topics:

- About Prebuilt Portlets

- About Java Server Faces (JSF) Portlets

- About Custom Java Portlets

- About OmniPortlet

- Portlet Technologies

- Portlets Versus Task Flows

11.3.1 About Prebuilt Portlets

Oracle provides integration with other Oracle applications by enabling you to expose functionality as portlets:

What Are They?

Supported applications include:

- **Peoplesoft**—You can expose Peoplesoft applications as WSRP portlets. For more information, see Integrating PeopleSoft Applications in *Administering Oracle WebCenter Portal*.

- **JD Edwards**—You can expose JD Edwards standalone regions as portlets. For more information, see Integrating JD Edwards Applications in *Administering Oracle WebCenter Portal*.

- **Oracle E-Business Suite**—Oracle E-Business Suite provides several prebuilt portlets, such as Applications Navigator, Favorites, and Worklist, that you can add to WebCenter Portal. For more information, see Integrating E-Business Suite Applications in *Administering Oracle WebCenter Portal*.

- Other prebuilt portlets are available through Oracle's partnerships with leading system integrators, software vendors, and content providers.

Who Is the Intended User?

Fully developed, downloadable portlets are best suited for use by portal developers who understand how to download, install, and register producers in WebCenter Portal. They are available for use by all levels of experience.

When Should They Be Used?

Use prebuilt portlets when your needs are met by the functions the portlets offer and the level of personalization readily available is sufficient to complete the desired task.

Consider alternatives when you want 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.

11.3.2 About Java Server Faces (JSF) Portlets

JSF portlets are created from JSF applications using the Oracle JSF Portlet Bridge.

What Are They?

The Oracle JSF Portlet Bridge enables application developers to expose their existing JSF applications and task flows as JSR 286 Java portlets. The Oracle JSF Portlet Bridge simplifies the integration of JSF applications with WSRP portlet consumers, such as WebCenter Portal.

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 portletized JSF application, the JSF portlets are also deployed with it. Therefore, using the Oracle JSF Portlet Bridge eliminates the requirement to store, maintain, and deploy your portlets separately from your application.

For more information about creating JSF portlets, see Creating Portlets from JSF Applications Using the Oracle JSF Portlet Bridge.

Who Is the Intended User?

Application developers with knowledge of Faces.

When Should They Be Used?

JSF portlets are best suited when application developers intend to display content from a JSF application or from individual task flows within the 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 the same as any WSRP producer.

11.3.3 About Custom Java Portlets

Custom Java portlets are portlets that you write yourself, in Java, using the Java Portlet Specification 2.0 (JSR 286).

What Are They?

WebCenter Portal provides a declarative wizard in JDeveloper for simplifying the creation of standards-based JSR 286 Java portlets. This wizard assists in the construction of the framework within which you create the portlet.

> **Note:**
>
> You can consume legacy JSR 168 and WSRP 1.0 portlets in your portals, for example from Oracle WebLogic Portal. There is no need to convert these portlets to JSR 286 first. However, when creating new Java portlets we recommend using JSR 286 to take advantage of features such as public render parameters and portlet events.
>
> You can also consume legacy Oracle PDK-Java portlets in your portals.

Who Is the Intended Audience?

Use of the wizard is easy, but creation of the portlet logic is best performed by experienced and knowledgeable Java developers who are comfortable with JSR 286 and who understand the configuration of producers.

When Should They Be Used?

Consider using custom Java portlets when you cannot find a prebuilt portlet or existing JSF application to address your requirements.

Use custom Java portlets when you have very specialized business rules or logic, or when you require personalized authentication, granular processing of dynamic results, or complete user interface control.

For more information about creating custom Java portlets, see Building Standards-Based Java Portlets Using JSR 286.

11.3.4 About OmniPortlet

OmniPortlet is a declarative portlet-building tool that enables you to build portlets against a variety of data sources.

What Is It?

OmniPortlet data sources include XML files, character-separated value files (CSV, for example, spreadsheets), web services, and databases. 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.

Who Is the Intended User?

Business users with a minimum knowledge of the URLs to their targeted data and limited programming skills 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.

For more information about OmniPortlet, see Working with OmniPortlet in *Building Portals with Oracle WebCenter Portal*

As an alternative to OmniPortlet, you should consider using data integration and presentation. For more information, see Working with Data Visualizations in *Building Portals with Oracle WebCenter Portal*.

If you want to include data from another web page, consider using a clipper pagelet using Oracle WebCenter Portal's Pagelet Producer's Pagelet Producer. For more information, see How to Configure the Clipper.

11.3.5 Portlet Technologies

When creating new portlets, there are several different technologies you can use. The technology you use depends on several factors.

Figure 11-3 provides a decision tree to enable you to determine which portlet implementation technology best fits your environment and requirements. The list below the figure presents the same information in a non-graphical format and provides additional notes.

Figure 11-3   Decision Tree for Portlet Technologies
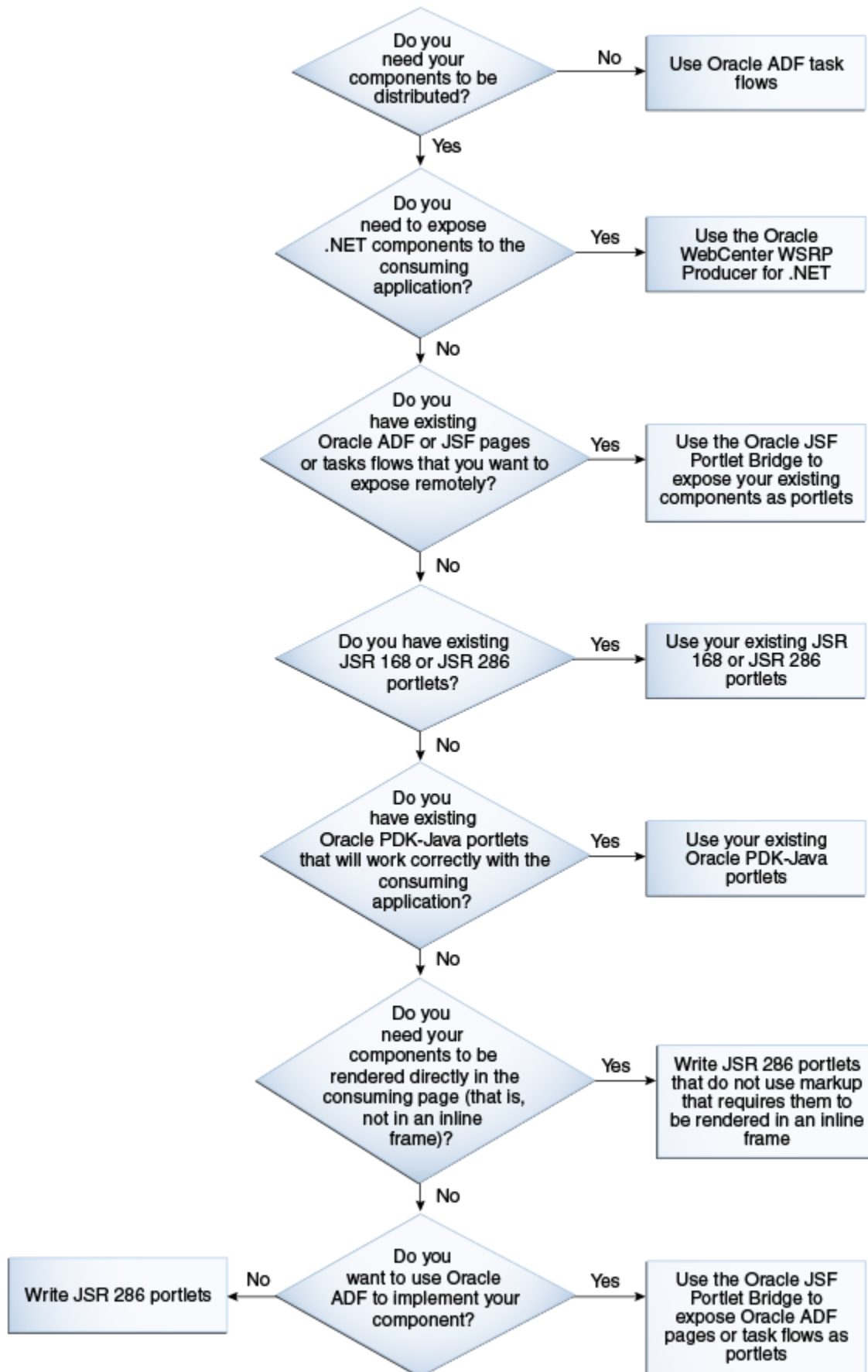
1. Do you need your components to be distributed?

   - **Yes**—Go to Question 2.

   - **No**—You don't need portlets. Use Oracle ADF task flows (or some other technology) that are local to your consuming application. For more information, see Creating ADF Task Flows in *Developing Fusion Web Applications with Oracle Application Development Framework*.

   Having a distributed environment results in a more complicated deployment and potentially makes it more difficult to track down issues. However, it also means that you can spread the processing load across multiple servers and enables your components to be used by several different consumers.

2. Do you need to expose .NET components to the consuming application?

   - **Yes**—Use the Oracle WebCenter WSRP Producer for .NET. For more information see *Developer's Guide for Oracle WebCenter WSRP Producer for .NET*.

   - **No**—Go to Question 3.

   The Oracle WebCenter WSRP Producer for .NET allows any .NET application to act as a WSRP producer.

3. Do you have existing Oracle ADF or JSF pages or task flows that you want to expose remotely?

   - **Yes**—Use the Oracle JSF Portlet Bridge to expose your existing components. For more information, see Creating Portlets from JSF Applications Using the Oracle JSF Portlet Bridge.

   - **No**—Go to Question 4.

   Using existing Oracle ADF components in this way provides a very convenient way of developing portlets, but be aware of the implications of using Oracle ADF to implement portlets, as outlined in Table 11-1.

4. Do you have existing JSR 168 or JSR 286 portlets?

   - **Yes**—Use your existing JSR 168 or JSR 286 portlets. For more information, see Registering WSRP Producers.

   - **No**—Go to Question 5.

   WebCenter Portal exposes JSR 168 and JSR 286 portlets over WSRP, which are fully compatible with the WebCenter consumer, assuming such portlets do not make assumptions that are specific to a particular consumer.

5. Do you have existing Oracle PDK-Java portlets that will work correctly with the WebCenter Portal consumer?

   - **Yes**—Use your existing PDK-Java portlets. For more information, see Registering an Oracle PDK-Java Portlet Producer

   - **No**—Go to Question 6.

6. Do you have a requirement for your components to be rendered directly in the consumer page (that is, not in an inline frame)?

- **Yes**—Write JSR 286 portlets that do not use markup that requires them to be rendered in an inline frame. For more information, see Building Standards-Based Java Portlets Using JSR 286.

- **No**—Go to Question 7.

7. Do you want to use Oracle ADF to implement your component?

- **Yes**—Use the Oracle JSF Portlet Bridge to expose Oracle ADF pages or task flows as portlets. For more information, see Creating Portlets from JSF Applications Using the Oracle JSF Portlet Bridge.

- **No**—Write JSR 286 portlets. For more information, see Building Standards-Based Java Portlets Using JSR 286.

The decision of whether to use Oracle ADF for implementing portlets is broadly the same as whether you would choose to use it for implementing a normal web application. However, the additional network hop (for markup, resource, and PPR) and consumer processing that is inherent when using remote portlets may make Oracle ADF a less attractive option than it is in the non-portlet case.

Also, see the Oracle ADF Overview white paper on OTN for specific benefits of using Oracle ADF to develop Java EE applications. Another benefit of using an Oracle ADF task flow is that you can later run the same task flow locally if you encounter any issues running in an inline frame.

Table 11-1 outlines the implications of choosing a particular portlet technology for implementing components for use with WebCenter Portal against various criteria.

Table 11-1   Portlet Building Technologies Comparison Matrix

Local Oracle ADF Pages and Task Flows	WebCenter WSRP Producer for .NET	JSR 286 Portlets	JSF Portlets
Distributed Architecture			
No.	Yes, using WSRP SOAP/HTTP.	Yes, using WSRP SOAP/HTTP.	Yes, using WSRP SOAP/HTTP.
Data Access Support			
Full support for working with databases, web services, and legacy systems.	No restrictions. This leverages the underlying .NET framework, which provides data access to a wide variety of data repositories.	No restrictions, but must be implemented by the portlet developer, possibly using a third party framework.	Full support for working with databases, web services, and legacy systems.
Navigation/Control Flow Support			
Provides comprehensive support for declarative navigation.	Provides comprehensive support for declarative navigation.	No restrictions, but must be implemented by the portlet developer, possibly using a third party framework.	Provides comprehensive support for declarative navigation.
Richness of UI			

Table 11-1   (Cont.) Portlet Building Technologies Comparison Matrix

Local Oracle ADF Pages and Task Flows	WebCenter WSRP Producer for .NET	JSR 286 Portlets	JSF Portlets
Provides a rich, web-based UI with many AJAX-enabled components.	No restrictions. Supports both standard AJAX and ASP.NET AJAX toolkit.	No restrictions, but must be implemented by the portlet developer, possibly using a third party framework.	Provides a rich, web-based UI with many AJAX-enabled components.
Security			
Fully integrated with the application	User identity propagated using WS-Security or asserted by consumer using WSRP. Authorization checks must be implemented by the developer. User profile information can be passed using WSRP user profile items.	User identity propagated using WS-Security or asserted by consumer using WSRP. Authorization checks must be implemented by the developer. User profile information can be passed using WSRP user profile items.	User identity propagated using WS-Security. Authorization is implemented by Oracle ADF.
Inter-Component Communication (Events)			
ADF contextual events enable task flows to communicate.	We provide samples to show how it can be accomplished client-side. Current version does not support WSRP server-side eventing model.	Full support for JSR 286 events, including automatic event delivery between portlets and event mapping between portlet events and ADF contextual events in the consumer.	As per JSR 286 portlets in addition to ADF contextual events allowing task flows to communicate as in the local case.
Consumer to Component Interaction (Parameters)			
Task flow input parameters can take values from various sources.	Full support for WSRP portlet parameters. All parameters can be passed from the consumer application.	Full support for JSR 286 parameters, including passing public parameter values to and from the consumer and automatic sharing of parameter values between portlets.	As per JSR 286 portlets in addition to task flow input parameters taking values from various sources in the consumer application as in the local case.
Performance and Caching			

Table 11-1   (Cont.) Portlet Building Technologies Comparison Matrix

Local Oracle ADF Pages and Task Flows	WebCenter WSRP Producer for .NET	JSR 286 Portlets	JSF Portlets
No additional network hop since the component runs on the local server.	Remote component requires additional network hop. Configuration parameters allow for resources (CSS, JavaScript, and `.axd` resource files) to be proxied via the consumer or direct access to reduce load and take advantage of browser caching.	Remote component requires additional network hop. Resources should be proxied by the consumer. Component performance depends on implementation. Expires and validation based caching are supported and controlled by the portlet developer.	Remote component requires additional network hop. Many resources are proxied via the consumer. Heavyweight nature of framework means simple applications are likely to suffer performance-wise versus handcrafted JSR 286 equivalents. However, for applications offering more complex functionality, the performance benefits of using the framework (for example, partial page rendering) become more apparent. JavaScript, CSS, and other resources are cached in the browser. Caching is controlled by the framework.
Concurrency			
Multiple task flows are invoked sequentially.	Portlets are invoked in parallel.	Portlets are invoked in parallel.	Portlets are invoked in parallel.
Geometry Management			
Components are rendered directly in the containing page.	Portlets are automatically rendered in a proxied inline frame and resized accordingly. The slider bars are hidden from view.	Portlet geometry and inline versus framed rendering is controlled by the portlet developer. Complex UIs are more likely to require rendering in an inline frame.	Portlets must be rendered in an inline frame.
Use of JavaScript			
JavaScript used heavily as a fundamental part of the framework to provide a rich user experience. Developer does not generally need to write JavaScript.	JavaScript used heavily as a fundamental part of the framework to provide a rich user experience. Developer does not generally need to write JavaScript, but samples are provided to show how JavaScript can be used. All scripts can be proxied via the consumer.	Level of JavaScript use and packaging are controlled by the portlet developer. Developers must be aware of special considerations regarding the use of JavaScript if portlets are to be rendered inline.	JavaScript used heavily as a fundamental part of the framework to provide a rich user experience. Developer does not generally need to write JavaScript. All scripts must be proxied via the consumer.

Table 11-1   (Cont.) Portlet Building Technologies Comparison Matrix

Local Oracle ADF Pages and Task Flows	WebCenter WSRP Producer for .NET	JSR 286 Portlets	JSF Portlets
Skills Requirements/Ease of Development			
Requires knowledge of ADF and possibly Java. ADF provides an integrated framework across all layers. JDeveloper provides a fully integrated environment for developing ADF applications. This includes declarative options for all layers of the framework.	Requires knowledge of .NET web parts development or ASP.NET web application development.	Requires knowledge of Java, Java portlets, and any other frameworks used to implement components. Development time comparable with that required to develop a servlet-based component.	Requires knowledge of ADF and possibly Java. Development time is likely to be reduced (versus handcrafted portlets) for all but the most simple of applications due to the rich UI and control framework provided by ADF. ADF provides an integrated framework across all layers. JDeveloper provides a fully integrated environment for developing ADF applications. This includes declarative options for all layers of the framework.
Debugging			
JDeveloper provides a comprehensive solution for debugging application code.	.NET framework provides a comprehensive solution for debugging application code and network trace for SOAP and HTTP traffic. Log4net plug-in for WSRP consumer-producer debug messages.	Made more difficult by remote nature of deployment. Producer provides tool for monitoring consumer-producer communication. JDeveloper provides a comprehensive solution for debugging application code.	Made more difficult by remote nature of deployment. Producer provides tool for monitoring consumer-producer communication. JDeveloper provides a comprehensive solution for debugging application code.
Deployment			
Components are deployed as part of the client application. Entire application can be packaged into a single EAR file and deployed.	Deployment requires copying .NET applications under a particular folder hierarchy and filling out a configuration XML file. Components can easily be run standalone under this folder hierarchy if required.	Made more complex by distributed nature of deployment and multiple applications.	Made more complex by distributed nature of deployment and multiple applications. Components can easily be run standalone if required.
Level of Support			

Table 11-1   (Cont.) Portlet Building Technologies Comparison Matrix

Local Oracle ADF Pages and Task Flows	WebCenter WSRP Producer for .NET	JSR 286 Portlets	JSF Portlets
The ADF framework is an Oracle product and is supported as such.	The WebCenter WSRP Producer for .NET is an Oracle product and is supported as such.	The JSR 286 producer is an Oracle product and is supported as such.	The ADF framework, Oracle JSF Portlet Bridge, and JSR 286 producer are Oracle products and are supported as such.

11.3.6 Portlets Versus Task Flows

A common question asked when creating portal is when to use portlets versus task flows.

Portlets are reusable Web components that can draw content from many different sources. Portlets can manage and display anything from static HTML to Java controls to complex web services and process-heavy applications.

A single portlet can also have multiple instances—in other words, it can appear on a variety of different pages within a single portal, or even across multiple portals if the portlet is enabled for Web Services for Remote Portlets (WSRP). You can customize portlets to meet the needs of specific users or groups.

Task flows let you build customizable applications with reusable units of business logic. They represent a modular approach for defining control flow in an application. Each task flow represents a reusable piece of business logic. With isolated memory and transaction scopes, task flows are decoupled from the surrounding application. This decoupling allows task flows to be included in Oracle WebCenter Portal's page editor, for instance.

A task flow encapsulates control flow rules, activities, and managed beans to implement business modules. Task flows are easily reusable within portals built with WebCenter Portal. You can drag and drop task flows onto any customizable WebCenter Portal portal page. In this sense, you can think of a task flow as a "local portlet" that can be reused and dropped into any ADF application.

While task flows are high-level implementations of business processes, they do not host data-access or business logic. Task flows must interact with data controls and declarative bindings for these purposes. These data controls and bindings in turn interact with the ADF Business Components framework.

Following the Oracle-submitted standard JSR 329, you can expose your task flows as standards based portlets. 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 329, WebCenter Portal provides the Oracle JSF Portlet Bridge. For more information about the Oracle JSF Portlet Bridge, see Creating Portlets from JSF Applications Using 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. For more information, see Getting Started with ADF Task Flows in the *Developing Fusion Web Applications with Oracle Application Development Framework.*

For more information about when to use portlets versus task flows, see the "Oracle WebCenter & ADF Architecture Team" blog.

11.4 About Portlet Development

You can develop your portlet producers in Oracle JDeveloper.

This section includes the following topics:

- About Portlet Producer Applications
- How to Create a Portlet Producer Application
- Portlet Modes
- Interportlet Communication
- Portlet Personalization and Customization
- Portlet Performance
- Multilanguage Portlets
- Portlet Deployment

11.4.1 Portlet Producer Applications

This section includes the following topics:

- About Portlet Producer Applications
- How to Create a Portlet Producer Application

11.4.1.1 About Portlet Producer Applications

A portlet producer application is an application specifically configured for building and deploying Java portlets.

To readily find and use the appropriate components in your portlet producer application, you must ensure that the appropriate technology scopes are set, tag libraries added, and required Java classes are added to the class path. Once you do this, relevant components are included in the Component Palette and relevant context menus become available in JDeveloper.

JDeveloper provides an application template to ensure that scopes are set properly and the right tag and Java libraries are added to create portlet producer applications. The WebCenter Portlet Producer Application template provides a way of easily creating an application with a single project scoped for the creation of Java portlets.

11.4.1.2 How to Create a Portlet Producer Application

To create portlets, you must first create a portlet producer application.

To create an application using the WebCenter Portlet Producer Application template:

1. Access the New Gallery dialog in any of the following ways:

   - From the **File** menu, choose **New**, then **Application**.
   - From the **Application** menu, choose **New**.
   - If you have an existing application open, in the Application Navigator, click the application name and choose **New Application**.

- If you have an existing application open, then in the Application Navigator, right-click the application name and choose **New**, then **Application**.

2. In the New Gallery dialog, select **WebCenter Portlet Producer Application**, and then click **OK**.

3. In the Name your application page of the Create WebCenter Portlet Producer Application wizard, in the **Application Name** field, enter a name for the application.

4. 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\myApplication
```

Optionally, click the **Browse** button to navigate to the desired directory.

5. If required, in the **Application Package Prefix** field, enter a prefix to use for packages to be created within the application.

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

Figure 11-4 shows a newly created portlet producer application, with its single Portlets project, in the Application Navigator.

Figure 11-4   New Portlet Producer Application in the Application Navigator
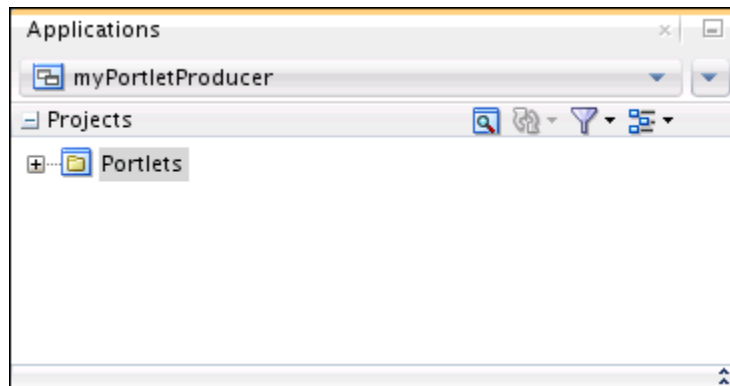
Options exposed in the Oracle JDeveloper user interface are limited to those appropriate for developing portlets.

For information about how to create portlets within this application, see Building Standards-Based Java Portlets Using JSR 286.

11.4.2 Portlet Modes

This section includes the following topics:

- About Portlet Modes

- View Mode

- Edit Mode

- Edit Defaults Mode

- Help Mode

- About Mode

11.4.2.1 About Portlet Modes

Portlet modes exhibit the runtime functionality seen by users. WebCenter Portal supports the following standard portlet modes:

- View mode

- Edit mode

- Edit Defaults mode

- Help mode

- About mode

WebCenter Portal defines an extended set of modes that it supports. Different modes are available to JSR 286 portlets. For example, the Create JSR 286 Java Portlet Wizard includes Print mode, which you can use to provide a printer-friendly version of the portlet.

If you are coding portlets to JSR 286, then you can also declare your own custom portlet modes in the `portlet.xml` file. You can use these to accommodate any other functionality you may want the portlet to provide.

Defining custom portlet 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.

Note:

Third party portal products may have their own set of extended modes (JSR 286 custom modes) that producers can offer. Arbitrary custom modes that a third party or custom portlet producer offers are ignored and therefore are not supported.

11.4.2.2 View Mode

A portlet uses *View mode* to appear on a page with other portlets. This is the mode most people think about when they envision a portlet. A JSR 286 portlet must have a View mode, other modes 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 an HTML table cell. 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 end user settings, the browser width, and the amount and style of content in the portlet.

11.4.2.2.1 HTML Guidelines for Rendering Portlets

Plain HTML is the most basic way to render portlets and provides a great deal of flexibility to portlet developers. You can use almost any standard HTML paradigm, such as links, forms, images, and 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 to not appear at all. Ensure that you adhere to the following rules:

- **Use standard HTML**—The official HTML specification is available from the W3C. For more information, see `http://www.w3c.org/`.

- **Avoid lengthy, complex HTML**—The HTML you use 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. If portlets must render complex HTML or wait for external resources, such as third party applications, then it can greatly slow down the rendering of the page.

- **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.

- **Consider restrictions imposed by container objects**—If your portlet is contained inside 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 cannot 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.

- **Do not create fixed-width HTML tables in portlet**—You have no way to tell how wide a column your portlet has on a user's page. If your portlet requires more room than given, then it might overlap with another portlet in some browsers.

- **Avoid long, unbroken lines of text**—The result is similar to what happens with fixed-width tables. Your portlet might overlap other portlets in some 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**—End users may choose whatever font size they want and they can change it at any time. Your portlet should handle these situations gracefully.

11.4.2.2.2 CSS 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 Cascading Style Sheets (CSS). 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, WebCenter Portal uses a very basic subset of this specification to allow for consistent fonts and colors. CSS implementation levels should not affect the consistency of your pages across browsers. Consider the following guidelines for using CSS:

- **Use CSS instead of hard coding**—Hard coding fonts and colors is not advised. If you hard code fonts and colors, then your portlet may look out of place if the end user changes the page style settings. Since you have no way of knowing the end

user's font and color preferences, 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's content will not be visible to that user.

- **Avoid using CSS for absolute positioning**—Since users may be able to personalize pages, you cannot guarantee that your portlet can appear in a particular spot on the page.

- **Follow accessibility standards**—You should ensure that you code your style sheets according to existing accessibility standards. For more information, see `http://www.w3c.org/TR/WCAG10-CSS-TECHS`.

11.4.2.3 Edit Mode

A portlet uses *Edit mode* to enable users to *personalize* the behavior of the portlet. Personalizations are visible only to the user who performs them, not to other users. 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 else that affects the appearance or content of the portlet.

Users typically access a portlet's Edit mode by clicking the **Personalize** icon in the portlet header (Figure 11-5).

Figure 11-5   Personalize Icon

When users click **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.

11.4.2.3.1 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 end 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 from the cache. If you do not do this, the user may see incorrect content.

- **Do not use Edit mode as an administrative tool**—Edit mode is meant to give end users a way of changing the behavior of portlets. If you want to change producer settings or perform other administrative tasks, then you should create secured portlets specifically for these tasks.

11.4.2.3.2 Guidelines for Buttons in Edit Mode

For consistency and user convenience, Edit mode should implement the following buttons in the order listed:

- **OK**—Saves the end user's personalizations and returns the portlet to View mode.

- **Apply**—Saves the end user's personalizations and reloads the current page.

- **Cancel**—Returns the portlet to View mode without saving changes.

11.4.2.4 Edit Defaults Mode

A portlet uses *Edit Defaults mode* to enable application administrators to customize the default behavior of a particular portlet instance at runtime. Edit Defaults mode provides a list of settings that the application administrator can change. These settings may include the title, type of content, formatting, amount of information, defaults for form elements, or anything that affects the appearance or content of the portlet.

Whereas personalizations affect a portlet instance for an individual user, *customizations* made in Edit Defaults mode affect the portlet instance 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.

Application administrators access Edit Defaults mode, when editing a page, by choosing **Customize** from the component's View Actions menu (Figure 11-6).

Figure 11-6   Customize Portlet
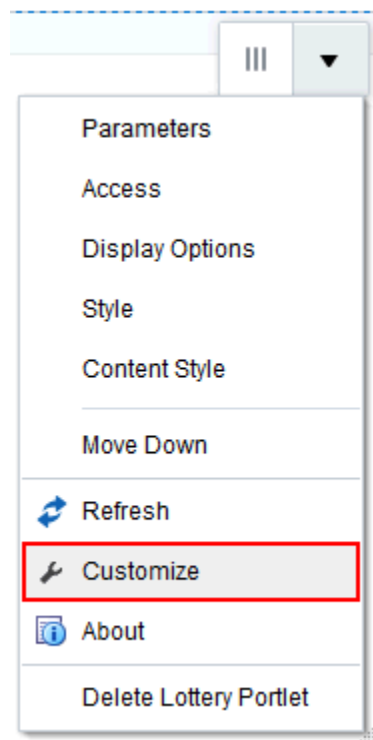
When users click **Customize**, a new page appears in the same browser window. The portlet typically creates a web page representing a dialog to customize the portlet instance settings. After applying the settings, users are automatically returned to the original page.

11.4.2.4.1 Guidelines for Edit Defaults Mode Operation

The following guideline should govern what you expose to application administrators in Edit Defaults mode:

- **Do not use Edit Defaults mode as an administrative tool**—Edit Defaults mode gives application administrators a way of changing the behavior of their portlets. If you want to change producer settings or do other administrative tasks, then you should create secured portlets specifically for those tasks.

11.4.2.4.2 Guidelines for Buttons in Edit Defaults Mode

For consistency and user convenience, Edit Defaults mode should implement the following buttons in the order listed:

- **OK**—Saves the application administrator's customizations and returns the portlet to View mode.

- **Apply**—Saves the application administrator's customizations and reloads the current page.

- **Cancel**—Returns the portlet to View mode without saving changes.

11.4.2.4.3 Guidelines for Rendering Customization Values

When you show the forms used to change customization settings, you should default the values so that the application administrator does not have to constantly reenter settings. When rendering customization values, use the following sequence to provide consistent behavior:

- **Instance preferences**—Query and display system defaults for the portlet instance.

- **Portlet defaults**—If no system default customizations are found, then display general portlet defaults, which may be blank. General portlet defaults are sometimes hard coded into the portlet but should be overwritten by system defaults.

11.4.2.5 Help Mode

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 contents, and its capabilities with this mode.

Users access a portlet's Help mode by choosing **Help** from the **Actions** menu (Figure 11-7).

Figure 11-7   Help Option in the Portlet Actions Menu
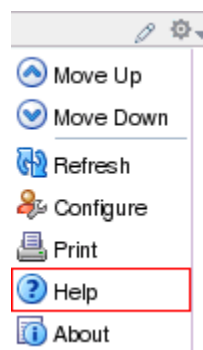
11.4.2.5.1 Guidelines for Help Mode

The following guideline should govern what you expose in Help mode:

- **Describe how to use the portlet**—Users may not be able to ascertain all the features your portlet offers just from its interface. Describe the features and how to get the most out of them.

11.4.2.6 About Mode

A portlet uses *About mode* to provide users with information such as the version of the portlet that is currently running, its publication and copyright information, and how to contact the portlet developer. Portlets that require registration could also use About mode to link to a web-based registration application or contact information.

Users access a portlet's About mode by choosing **About** from the **Actions** menu (Figure 11-8).

Figure 11-8   About Option in the Portlet Actions Menu
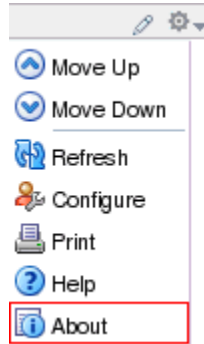
A new page appears in the same browser window. The portlet can either generate the content for this page or take the user to an existing page or application.

11.4.2.6.1 Guidelines for About Mode

The following guideline should govern what you expose to users in About mode:

- **Display relevant copyright, version, and developer 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.

11.4.3 Interportlet Communication

Interportlet communication enables portlets to share data and react to events. WebCenter Portal supports interportlet communication through parameters and events:

- **Parameters**—Public render parameters enable interportlet communication by sharing parameter values between portlets. For example, if a Map portlet and a Weather portlet are both configured to use a Zipcode parameter, entering a zip code in the Map portlet updates the same parameter value in the Weather portlet.

  For information about public render parameters, see How to Use Public Render Parameters in JSR 286 Portlets.

- **Events**—Events enable interportlet communication by providing portlets with the ability to respond to actions that occur outside of the portlet itself, for example, an action performed on the page that contains the portlet or on another portlet on the same page. Portlet events can be cascaded so that a portlet may respond to an event by triggering an event of its own, which in turn affects other portlets on the page.

  For information about portlet events, see How to Use Portlet Events in JSR 286 Portlets.

In WebCenter Portal, interportlet communication happens automatically by default, as long as parameters and events share the same name or define an appropriate alias. You do not need to supply any extra code to enable this communication.

For interportlet communication guidelines specific to portlets created using the Oracle JSF Portlet Bridge, see Using Events to Link JSF Portlets with Other Portlets.

11.4.4 Portlet Personalization and Customization

Portlet preferences enable end users to personalize or customize portlets at runtime. Personalizations are visible only to the user that performs them; not to other users. Customizations are visible to all users who have not personalized the portlet.

Portlet preferences greatly enhance the reusability of portlets. The same portlet can be instantiated on multiple pages, or multiple times on the same page. Using portlet preferences, each instance of the portlet can behave in a different way, but still use the same code and user interface.

For example, by default when you create a JSR 286 portlet using the JDeveloper wizard, a portlet preference is created to enable users to change the title of the portlet at runtime. You can create additional portlet preferences, either during portlet creation or by editing the `portlet.xml` file after portlet creation, to enable users to perform other modifications on the portlet at runtime.

Portlet preferences are stored in the persistence store, along with information about the registration and portlet instances, such as registration and portlet handles. Portlet producers can use one of three types of persistence store:

- **Consumer**—A consumer persistence store ties the producer metadata to the consumer application. It is recommended that you should use a consumer persistence store in your production environment.

- **Database**—A database persistence store persists data using a relational database.

- **File**—A file-based persistence store persists data using the file system.

A file-based or consumer persistence store may be used for testing, since it removes the dependency on a database. But, for production configurations, it is recommended that you use a consumer persistence store.

By default, JSR 286 and JSF portlets use a consumer persistence store. For information about how to change this default setting, see Setting Up a Persistence Store for a WSRP Producer.

For information about migrating the persistence store, for example, when moving from a test to production environment, see Migrating a WSRP Producer Persistence Store.

11.4.5 Portlet Performance

Caching is a common technique for enhancing the performance of web sites that include a great deal of dynamic content. The overhead involved in retrieving data and generating the output for dynamic content can be significantly reduced by proxying requests through a local agent backed by a large, low-latency data store, known as a cache. The cache agent responds to a request in one of two ways:

- If a valid version of the requested content exists in the cache, then the agent simply returns the existing cache copy, thus skipping the costly process of content retrieval and generation. This condition is known as 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 known as a *cache miss*.

Portlet producers generate dynamic content (that is, portlets) and they reside remotely from the WebCenter Portal 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. Portlet caching is key to rapid response to user requests.

For JSR 286 portlets there are two different caching methods. The methods differ mainly in how they determine whether content is still valid:

- **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.

  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.

  For information about how to implement expiry-based caching for your portlets, see Implementing Expiry-Based Caching in JSR 286 Portlets.

- **Validation-based caching**—When a producer receives a render request, it stamps a response with a version identifier (or ETag). The response goes into the cache, but before the consumer can reuse the cached content, it must determine whether the cached content 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 consumer without any content that indicates that 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 consumer 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 cached content, rather than relying on a fixed period.

  Consider using validation-based caching for portlets with dynamic content that changes frequently or unpredictably.

  For information about how to implement validation-based caching for your portlets, see Implementing Validation-Based Caching in JSR 286 Portlets.

Caching rules can be specified at a portlet's container level, encoded in the portlet's own logic, or established through the portlet wizard.

JSF portlets and custom Java portlets support expiry- and validation-based caching.

At the application level, WebCenter Portal uses Coherence for the establishment of application-level caching rules. The WebCenter Portal portlet client provides a default cache configuration file for portlets:

```
<?xml version="1.0"?>


<cache-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns="http://xmlns.oracle.com/coherence/coherence-cache-config"
               xsi:schemaLocation="http://xmlns.oracle.com/coherence/coherence-cache-
config
coherence-cache-config.xsd">
   <defaults>
      <scope-name>portlets</scope-name>
   </defaults>
   <caching-scheme-mapping>
      <cache-mapping>
         <cache-name>portlet-consumer-content-cache-*</cache-name>
         <scheme-name>portlet-thin-direct</scheme-name>
      </cache-mapping>
   </caching-scheme-mapping>
   <caching-schemes>
      <distributed-scheme>
         <scheme-name>portlet-thin-direct</scheme-name>
         <local-storage>true</local-storage>
      </distributed-scheme>
   </caching-schemes>
</cache-config>
```

You can create your own cache configuration file to use for the portlet consumer cache and update the portlet client configuration information in adf-config.xml to point to your file. For more information, see Editing the Portlet Client Configuration in *Administering Oracle WebCenter Portal*. Base your file on the default cache configuration file to ensure that it defines a mapping for the cache name portlet-consumer-content-cache-*. For more information about Coherence, see Improving Data Caching Performance in *Administering Oracle WebCenter Portal*.

11.4.6 Multilanguage Portlets

There are three steps involved in making your portlets available in multiple languages:

- Declare the locales supported by the portlet using the <supported-locale> element in the portlet.xml file. For example:

```
<portlet id="1328624289503">
 ...
 <supported-locale>en</supported-locale>
 <supported-locale>fr</supported-locale>
 <supported-locale>es</supported-locale>
 ...
</portlet>
```

- Make the portlet metadata that is displayed in the portlet (for example, the portlet title, description, keywords, and so on) translatable.

  You can include the translations directly in the portlet.xml file, for example:

```
<portlet id="1328624289503">
 <portlet-name>LotteryPortlet</portlet-name>
 <display-name xml:lang="en">Lottery Numbers</display-name>
 <display-name xml:lang="fr">Numéros de Loterie</display-name>
 <display-name xml:lang="es">Números de la Lotería</display-name>
 ...
 <supported-locale>en</supported-locale>
```

```
    <supported-locale>fr</supported-locale>
    <supported-locale>es</supported-locale>
    ...
</portlet>
```

Alternatively, you can provide translations in a separate resource bundle. You must declare the resource bundle in the `portlet.xml` file, for example:

```
<portlet id="1328624289503">
 <portlet-name>LotteryPortlet</portlet-name>
 <display-name>Lottery Numbers</display-name>
 ...
 <supported-locale>en</supported-locale>
 <supported-locale>fr</supported-locale>
 <supported-locale>es</supported-locale>
 <resource-bundle>portlet.resource.LotteryPortletBundle</resource-bundle>
 ...
```

Then, provide the translations in the resource bundle, for example:

```
# English Resource Bundle
#
# filename: LotteryPortletBundle_en.properties
javax.portlet.display-title=Lottery Numbers

# French Resource Bundle
#
# filename: LotteryPortletBundle_fr.properties
javax.portlet.display-title=Numéros de Loterie

# Spanish Resource Bundle
#
# filename: LotteryPortletBundle_es.properties
javax.portlet-title=Números de la Lotería
```

- Make the portlet content translatable the same way as you would for any other web application.

11.4.7 Portlet Deployment

Before a portlet can be registered with and consumed by an application, you must first deploy it. JSR 286 portlets are deployed to a *WSRP producer*, which is remote and communicates with the consumer application through Web Services for Remote Portlets (WSRP). JSF portlets are deployed as JSR 286 portlets, that is, to a WSRP producer, either from JDeveloper or manually.

WebCenter Portal supports WSRP versions 1.0 and 2.0. This standard provides support for interportlet communication and export or import of portlet customizations.

WSRP is a communication protocol between portlet consumer applications 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 286, .NET, or 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.

Figure 11-9 illustrates the basic architecture of WSRP producers.

Figure 11-9   WSRP Producer Architecture
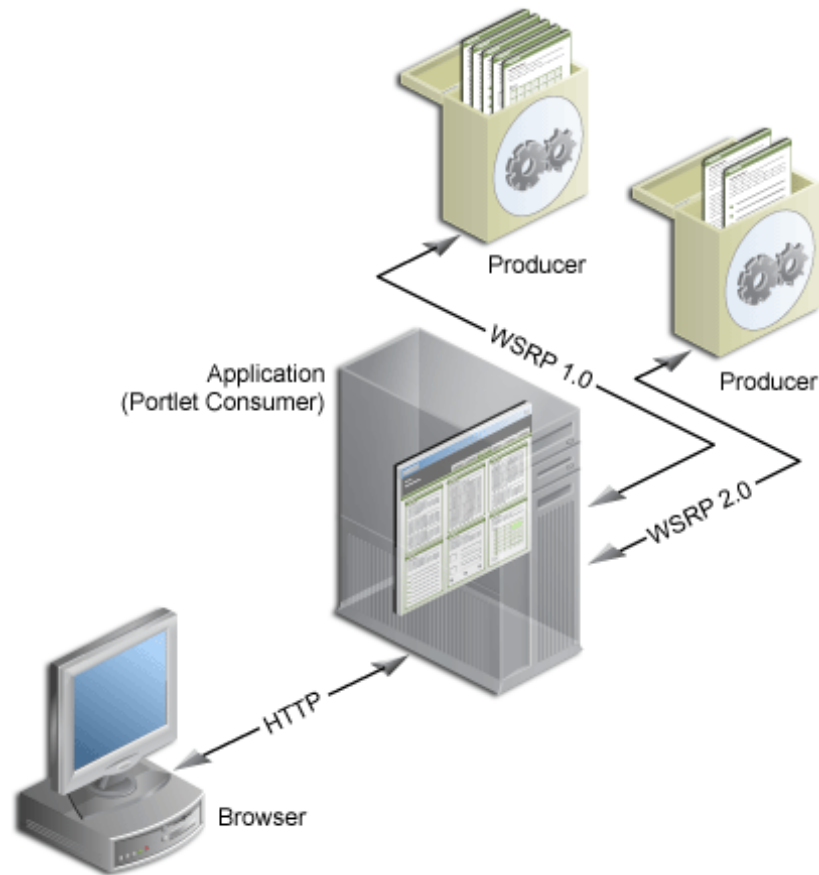
When end users display a page in their web browsers, the flow of the request works as follows:

1. The end user requests a page from the web browser by entering a URL in the browser's Location field.

2. The browser transmits the request to the consumer application over HTTP.

3. The application contacts the portlet producers that provide the portlets that display on the requested page.

    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.

4. The producers return the portlet content back to the consumer application using the WSRP 1.0 or 2.0 protocol:

5. The consumer application transmits the portlet content to the browser over HTTP.

To make standards-based portlets available to a consumer application in WebCenter Portal, you must package them in a Portlet Producer application and deploy them to a WSRP container.

12

Creating Portlets from JSF Applications Using the Oracle JSF Portlet Bridge

This chapter describes how to use the Oracle JSF Portlet Bridge to expose a JSF application as a portlet.

This chapter includes the following topics:

- About Creating Portlets from JSF Applications Using the Oracle JSF Portlet Bridge
- Creating a Portlet from a JSF Application
- Updating the Portlet Entry for a Portletized Page or Task Flow
- Testing a Portletized Page or Task Flow in JDeveloper
- Testing a JSF Portlet Using the Integrated WebLogic Server
- Deploying JSF Portlets to a WebLogic Managed Server
- Using Events to Link JSF Portlets with Other Portlets

> **Note:**
>
> The Oracle JSF Portlet Bridge extends the Apache Reference implementation of JSR 329. JSR 329 is the standards effort to define the functionality for the Portlet 2.0 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=329

12.1 About Creating Portlets from JSF Applications Using the Oracle JSF Portlet Bridge

The Oracle JSF Portlet Bridge enables application developers to quickly and easily expose their existing JSF applications and Oracle ADF applications and task flows as JSR 286 portlets.

> **Note:**
>
> Unless otherwise noted, the term JSF applications encompasses Oracle ADF applications.

The Oracle JSF Portlet Bridge:

- Simplifies portlet development by enabling you to provide portlet functionality using JSF rather than relying on the JSR 286 portlet APIs.

- Simplifies exposing your JSF application to JSR 286 portlet consumers, such as WebCenter Portal.

- Eliminates the requirement 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 JSR 286 portlets, this also enables you to use your task flows in a distributed environment.

Note:

The Oracle JSF Portlet Bridge uses WSRP extensions that may prevent JSF portlets working as intended with third party WSRP consumers.

12.2 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 286 portlet. As part of this configuration, you indicate the initial JSF view (or task flow) to 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.

This section includes the following topics:

- How to Create a JSF Portlet Based on a Page

- How to Create a JSF Portlet Based on a Task Flow

- What You May Need to Know When Creating a JSF Portlet

12.2.1 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. In JDeveloper, open the application that contains the JSF page that you want to portletize.

2. In the Application Navigator, right-click the JSF page and choose **Create Portlet Entry.**

Note:

You cannot generated a portlet based on a page that contains a portlet.

3. In the Create Portlet Entry dialog, in the **Portlet Name** field, enter a name for the portlet.

4. In the **Display Name** field, enter a descriptive name for your portlet.

5. 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.

6. 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.

7. In the **Description** field, enter a description for your portlet.

8. Select **Create portlet events for contextual events** to create portlet events in the `portlet.xml` file for any contextual events exposed by the page. This option is selected by default.

   Portlet events enable a portlet to communicate with the page on which it resides and with other portlets on that page.

9. Click **OK.**

When you create a JSF portlet based on a page, a `portlet.xml` file is created (or updated if it already exists) to contain the portlet entry (see example below) for the page. The file is opened ready for viewing or editing. By default, the file is opened in Design view. To view or edit the source code, click the **Source** tab.

```
<portlet id="adf_jsf__testPage_jspx">
 <description>PortletBridgeApplication_testPage_jspx</description>
 <portlet-name>PortletBridgeApplication_testPage_jspx</portlet-name>
 <display-name>PortletBridgeApplication_testPage_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>PortletBridgeApplication_testPage_jspx</title>
 <short-title>PortletBridgeApplication_testPage_jspx</short-title>
 </portlet-info>
 <supported-processing-event id="DepartmentSelectedEvent">
 <qname xmlns:x="http://xmlns.oracle.com/adfm/contextualEvent">
 x:DepartmentSelectedEvent
```

```
          </qname>
          <supported-publishing-event id="DepartmentSelectedEvent">
          <qname xmlns:x="http://xmlns.oracle.com/adfm/contextualEvent">
          x:DepartmentSelectedEvent
          </qname>
          </supported-publishing-event>
          <supported-public-render-parameter>
          _adf_event_DepartmentSelectedEvent
          </supported-public-render-parameter>
          <container-runtime-option>
          <name>com.oracle.portlet.requireIFrame</name>
          <value>true</value>
          </container-runtime-option>
          <container-runtime-option>
          <name>com.oracle.portlet.minimumWsrpVersion</name>
          <value>2</value>
          </container-runtime-option>
     </portlet>
```

The page you selected 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 and extended portlet modes supported by WebCenter Portal:

- Edit mode: `javax.portlet.faces.defaultViewId.edit`

- Help mode: `javax.portlet.faces.defaultViewId.help`

- About mode: `javax.portlet.faces.defaultViewId.about`

- Config mode: `javax.portlet.faces.defaultViewId.config`

- Edit Defaults mode: `javax.portlet.faces.defaultViewId.edit_defaults`

- Preview mode: `javax.portlet.faces.defaultViewId.preview`

- Print mode: `javax.portlet.faces.defaultViewId.print`

Note:

The value for the `defaultViewId` is relative to the application context root and must always start with a `/`. In the example above, the value for `defaultViewId.view` is `/myPage.jspx`.

If you add `defaultViewId` for other portlet modes, then ensure that you also add the mode to the `<supports>` tag. For example, `<portlet-mode>HELP</portlet-mode>`.

For information about portlet modes, see Portlet Modes.

12.2.2 How to Create a JSF Portlet Based on a Task Flow

This section includes the following topics:

- About Creating Portlets from Tasks Flows

- Creating a Portlet From a Task Flow Using the Create Portlet Entry Dialog

- Creating a Portlet From a Task Flow Using the Manage Portlet Entries of Task Flows Dialog

12.2.2.1 About Creating Portlets from Tasks Flows

An advantage of using Oracle ADF is the existence of task flows, which 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. Bounded task flows can use pages or page fragments, however only those that use page fragments can be portletized.

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, see *Developing Fusion Web Applications with Oracle Application Development Framework*.

12.2.2.2 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. In JDeveloper, open the JSF application that contains the task flow from which you want to make a portlet.

2. In the Application Navigator, right-click the task flow that you want to portletize, and choose **Create Portlet Entry.**

3. In the Create Portlet Entry dialog, in the **Portlet Name** field, enter a name for the portlet.

4. If the task flow is unbounded, the **Entry Point View** drop-down list displays all the view activities of the task flow. From this drop-down list, choose the view activity to use as the entry point for the portlet. By default, the first view activity in the task flow is chosen.

   If the task flow is bounded, there is only a single possible entry point, so you do not see the **Entry Point View** drop-down list.

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 portlet events for contextual events** to create portlet events in the `portlet.xml` file for any contextual events exposed by the task flow. This option is selected by default.

   Portlet events enable a portlet to communicate with the page on which it resides and with other portlets on that page.

10. Click **OK.**

When your portlet, or portlets, have been created, you should receive a message that says:

```
New portlet has been successfully created
```

In addition, the `portlet.xml` file is created. The `portlet.xml` file contains the portlet entry (see example below) and is opened ready for viewing or editing.

```
<portlet id="adf_taskflow_WEB_INF_department_xml">
 <description>PortletBridgeApplication department</description>
 <portlet-name>PortletBridgeApplication_department</portlet-name>
 <display-name>PortletBridgeApplication department</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?adf.tfDoc=/WEB-INF/adfp-portlet-bridge-container.xml
 &amp;adf.tfId=adfp-portlet-bridge-container&amp;_fragmentTaskFlowDoc=/WEB-INF/
department.xml&amp;_fragmentTaskFlowId=department
 </value>
 </init-param>
 <supports>
 <mime-type>text/html</mime-type>
 <portlet-mode>VIEW</portlet-mode>
 </supports>
 <supported-locale>en</supported-locale>
 <portlet-info>
 <title>PortletBridgeApplication department</title>
 <short-title>PortletBridgeApplication department</short-title>
 </portlet-info>
 <supported-publishing-event id="DepartmentSelectedEvent">
 <qname xmlns:x="http://xmlns.oracle.com/adfm/contextualEvent">
 x:DepartmentSelectedEvent
 </qname>
 </supported-publishing-event>
 <supported-public-render-parameter>
 _adf_event_DepartmentSelectedEvent
 </supported-public-render-parameter>
 <container-runtime-option>
 <name>com.oracle.portlet.requireIFrame</name>
```

```
<value>true</value>
</container-runtime-option>
<container-runtime-option>
<name>com.oracle.portlet.minimumWsrpVersion</name>
<value>2</value>
</container-runtime-option>
</portlet>
```

> **Note:**
>
> The `portlet.xml` file can include multiple portlets and can have a combination of pages and task flows exposed as portlets. The file can also include standard JSR 286 (non bridge) portlets.

12.2.2.3 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** and then **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.

4. Select **Create portlet events for contextual events** to create portlet events in the `portlet.xml` file for any contextual events exposed by the task flows. This option is selected by default.

    Portlet events enable a portlet to communicate with the page on which it resides and with other portlets on that page.

5. Click **OK** to create portlets for the selected task flows.

When your portlet, or portlets, have been created, you should receive a message that says:

```
New portlet has been successfully created
```

In addition, the `portlet.xml` file is created. The `portlet.xml` file contains the portlet entry (see example below) and is opened ready for viewing or editing.

```
<portlet id="adf_taskflow_WEB_INF_department_xml">
 <description>PortletBridgeApplication department</description>
 <portlet-name>PortletBridgeApplication_department</portlet-name>
 <display-name>PortletBridgeApplication department</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?adf.tfDoc=/WEB-INF/adfp-portlet-bridge-container.xml
&amp;adf.tfId=adfp-portlet-bridge-container&amp;_fragmentTaskFlowDoc=/WEB-INF/
department.xml&amp;_fragmentTaskFlowId=department
</value>
</init-param>
<supports>
<mime-type>text/html</mime-type>
<portlet-mode>VIEW</portlet-mode>
</supports>
<supported-locale>en</supported-locale>
<portlet-info>
<title>PortletBridgeApplication department</title>
<short-title>PortletBridgeApplication department</short-title>
</portlet-info>
<supported-publishing-event id="DepartmentSelectedEvent">
<qname xmlns:x="http://xmlns.oracle.com/adfm/contextualEvent">
x:DepartmentSelectedEvent
</qname>
</supported-publishing-event>
<supported-public-render-parameter>
_adf_event_DepartmentSelectedEvent
</supported-public-render-parameter>
<container-runtime-option>
<name>com.oracle.portlet.requireIFrame</name>
<value>true</value>
</container-runtime-option>
<container-runtime-option>
<name>com.oracle.portlet.minimumWsrpVersion</name>
<value>2</value>
</container-runtime-option>
</portlet>
```

> **Note:**
>
> The `portlet.xml` file can include multiple portlets and can have a combination of pages and task flows exposed as portlets. The file can also include standard JSR 286 (non bridge) portlets.

12.2.3 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 286 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, which naturally render markup in a style that is either compatible with JSF portlets or can be automatically interpreted by the Oracle JSF Portlet Bridge into a compatible style (for example, by rendering the portlet in an inline frame).

The sections that follow provide some guidance on how to avoid common issues that you may encounter as you code Oracle ADF pages that you later choose to publish as portlets.

This section includes the following topics:

- General Guidelines for Creating a JSF Portlet

- Portlet Guidelines

- Security Guidelines

- JSF Guidelines

- Oracle ADF Guidelines

12.2.3.1 General Guidelines for Creating a JSF Portlet

Consider the following general guidelines when creating a JSF portlet:

- **Application Must Run as a Web Application.** 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 Integrated WLS. If it cannot run as a regular web application, it cannot run as a portlet producer either.

- **The Portlet Is Contained by a Different Page.** One area that you should consider when creating task flows that will be exposed as portlets is that the page on which you render the task flow as a portlet is different to the one in the original producer application.

  Some common errors that occur include:

  – Expecting to access JavaScript in the consuming page.

  – Expecting to find managed beans or objects added to scopes as part of the consuming page.

  – Expecting components in the JSF component tree from the consumer to be present.

- **Servlet Dependencies.** 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 method shown in the example below to check if the application runs 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;
 }
}
```

- **Secured Applications.** If your application is secured, ensure that you have secured identity propagation.

- **Deep Links.** For normal interactions with a portlet, there is a session maintained between the consumer and the producer. This is done by storing a session cookie in the consumer and sending it to the producer on each request. On the original request, the session is established and on subsequent requests, the portlet sends the session cookie to the producer and the session is reused.

  For deep links, that is, links from the portlet to the producer application, the request is from the browser direct to the producer application. In this case, the session cookie cannot be sent so a new session is established. If a shared session is required for deep links, then this must be implemented by the developer. A common implementation is to write this state to a common location, accessible from the producer and the application, and pass a reference to this state in the deep link.

- **Java EE Authentication.** 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 286. In the JSR 286 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 must not contain login links that trigger Java EE authentication.

- **Timeout Period.** 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`.

12.2.3.2 Portlet Guidelines

Consider the following portlet guidelines when creating a JSF portlet:

- Portlet Sizing. When consuming a JSF portlet, the consumer application automatically renders the portlet content within an inline frame. This means that any inline popup is then confined within the inline frame. You should take this into consideration when specifying the size of the portlet.

  If the JSF portlet is a portletized page, then you have full control of the content and can decide how that content fills the inline frame window, utilizing exactly the same techniques as you would to determine how the content would fill the browser window.

  If the portlet is being stretched by its ancestor in the consumer application, then the `maximized` attribute is set to `true` on the `af:document` component in the document produced by the Oracle JSF Portlet Bridge. This means that components that support it, such as `panelStretchLayout`, will fill the entire inline frame.

- **Resources and Links.** 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 Portal run remotely and are accessed using a SOAP protocol (WSRP). The latter means that the regular web application concept of

request path does not apply in a JSR 286 container. The JSR 286 specification reflects this by mandating that all resource URLs either be absolute or context path relative.

- **Redirecting Requests.** Do not redirect or forward a request within your JSP. JSR 286 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.

- **Memory Consumption.** The Oracle JSF Portlet Bridge stores the content of the request scope between requests so that the request scope is preserved between a portlet Action and a portlet Render (which are two separate requests). Because of this, to minimize overall memory consumption, you must especially avoid storing too much data in the request scope when the application contains JSF portlets.

- **WSRP Version.** The Oracle JSF Portlet Bridge makes use of Portlet 2.0 features. Consequently, it should be used only with WSRP V2. The portlet's `minimumWsrpVersion` container runtime option should be set to `2`. For more information, see How to Customize the Runtime Environment for JSR 286 Portlets.

12.2.3.3 Security Guidelines

Consider the following security guidelines when creating a JSF portlet:

- **Oracle ADF Secured Applications.** 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 two 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>
    ```

- **Role Based Authorization.** 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 if the propagated user does not have permission to view the task flow.

12.2.3.4 JSF Guidelines

Consider the following JSF guidelines when creating a JSF portlet:

- **Using h:commandLink.** When using the `h:commandLink` JSF standard HTML component ensure that you set the `externalizeJavaScript` parameter in `web.xml` so that the JSF Reference Implementation does not generate any external JavaScript resource (see example below). This is to work around 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>
```

12.2.3.5 Oracle ADF Guidelines

Consider the following ADF guidelines when creating a JSF portlet:

- **Task Flow Returns.** When an ADF bounded task flow completes it may use a task flow return to send control back to the calling task flow. In a portletized task flow, there is no calling task flow, so the task flow return does not make any sense in this context. This may cause what appears to be an empty portlet to display as the task flow navigates to a state where there is no view selected. The task flow will stay in this state until it is refreshed either when the input parameters change or the session on the producer is reset.

  If you plan to portletize your task flow, you should avoid using task flow returns. If you must use a task flow return, you can provide a dummy task flow with a default activity of the task flow that you want to portletize. When the task flow return is executed, it returns to the dummy task flow. The dummy task flow then executes its default activity, which is to call back to the original task flow and thus restart the task flow when the portlet is next called.

- **Mismatched Style Sheets.** In general, when you consume a task flow using the Oracle JSF Portlet Bridge, the portlet producer tries to use the skin of the consumer page. When the producer cannot match the consumer skin, the portlet generates and references its own style sheet. This is known as a *mismatched skin*. A mismatched skin can occur when:

  - The consumer and producer use different versions of ADF Faces. In this case, the producer still uses the consumer skin.

  - The consumer has additional `skin-additions` that the producer does not have, or vice versa. In this case, the producer still uses the consumer skin.

  - The producer does not have the consumer skin. In this case, the producer uses the portlet skin.

  Mismatched skins can cause performance problems because the browser has to fetch an extra style sheet and the generated portlet markup uses uncompressed styles resulting in larger markup.

If a mismatched skin occurs, the producer log includes the following:

```
The skin skinName specified on the requestMap will not be used because the
styleSheetDocument id on the requestMap does not match the local skin's
styleSheetDocument's id. It could mean the jars are not identical. For example,
one might have trinidad-skins.xml skin-additions in a jar file on the class path
that the other does not have.
```

The portlet markup inside the inline frame also includes a link tag to the portlet style sheet resource, for example:

```
<link rel=\"stylesheet\" charset=\"UTF-8\" type=\"text/css\" href=\"http:.../
resourceproxy/portletId...252525252Fadf%252525252Fstyles%252525252Fcache
%252525252Fblafplus-rich-portlet-d1062g-en-ltr-gecko.css...
```

You can use an HTTP monitoring tool to see that a request is made to the portlet style sheet resource.

There are a number of reasons for mismatched skins. For a skin to match, the producer and consumer must be the same for all the following conditions:

– The ADF Faces version; a different version of ADF Faces may have different style selectors.

– The style compression, defined in `web.xml`, for example:

```
<context-param>
 <param-name>
 org.apache.myfaces.trinidad.DISABLE_CONTENT_COMPRESSION
 </param-name>
 <param-value>false</param-name>
</context=param>
```

– Tonal style or themes, defined in `web.xml`, for example:

```
<context-param>
 <param-name>xyz</param-name>
 <param-value>xyz</param-value>
</context-param>
```

– Availability of skin additions. Skin additions are defined in `META-INF/trinidad-skins.xml` in a JAR file. These are then aggregated from all the JAR files in the class path. If there are any JAR files that exist on the producer but not on the consumer, or vice versa, you get a mismatch.

```
<skin-addition>
 <skin-id>blafplus-rich.desktop</skin-id>
 <style-sheet-name>
 adf/styles/flexComponents-blafplus-rich-desktop.css
 </style-sheet-name>
</skin-addition>
```

To determine if the JAR files match, turn on Trinidad logging to show which `skin-addition` it is processing. To do this, update the `logging.xml` log level of both the producer and consumer WebLogic Servers to `FINEST` then restart the servers:

```
<logger name="org.apache.myfaces.trinidad.skin.SkinUtils" level=FINEST"/>
```

When you run the consumer page, any skin entries that do not match in the consumer and producer log files are the cause of the skin mismatch.

- **ADF Faces Dialog Framework.** ADF Faces Dialog Framework is not supported. If your application includes any buttons or icons that launch secondary browser windows, then 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>`

- **Composer Components.** Portletization of pages that contain Composer components is not supported.

- **The fileDownloadActionListener Component.** The `<af.fileDownloadActionListener>` component is not supported.

- **The prepareModel Phase.** Oracle ADF components/code that handle `prepareModel` must be idempotent. Any code running during the `prepareModel` phase must be rerunnable without effect to the underlying data/ business logic. When running in the portlet environment, `prepareModel` is called twice during the complete JSF life cycle as opposed to being called once when running within a regular web application.

    The reason for this difference is that the Oracle JSF Portlet Bridge runs 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.

- **Accessing Request Parameters.** 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 lifecycle in a single request, the Oracle JSF Portlet Bridge runs 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}"
/>
```

- **Referencing Page Parameters.** Do not reference page parameters in the `prepareModel` phase. This issue relates to the same problem described for request parameters in the previous guideline. 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 fails. Instead, move any dependency on a page parameter value into the model before JSF transitions from its execute phases to its render phase.

- **Using ADF Faces Client-Side APIs to Find Components.** 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 Producer 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 Using Naming Containers in the *Developing Web User Interfaces with Oracle ADF Faces*.

- **Encoded URLs.** By default, URLs in the portletized page or task flow are encoded such that they are proxied by the consumer application. In some circumstances, you may not want to encode these URLs, for example, when requesting JavaScript libraries from content delivery networks or accessing resources directly from the user's browser to make use of a locally configured HTTP proxy.

  Another example is if your task flow includes an inline frame (for example, using an `af:inlineFrame` tag). In this case you need to ensure that the inline frame URL does not get encoded to go via the portlet resource proxy in the consumer. If you do not do this, any relative URLs in the content inside the inline frame will not

work because they will be relative to the URL to the resource proxy rather than their correct location relative to the original URL for the inline frame.

To bypass the URL encoding, set the `_xEncodeUrl` parameter to `false` on URLs that are passed to the Oracle JSF Portlet Bridge for encoding. This parameter is picked up by the Oracle JSF Portlet Bridge, which ensures that the URL does not get encoded via the resource proxy. The parameter is ignored if the task flow is not running as a portlet.

For example, replace:

```
<af:inlineFrame source="http://myiframe.example.com">
```

with:

```
<af:inlineFrame source="http://myiframe.example.com?_xEncodeUrl=false">
```

> **Note:**
>
> Do not use this parameter to access resources that reside on the producer server. These resources must be accessed using the portlet client's proxy.

- **Issues with goLink.** The `adf:goLink` component encodes the link URL as an action for the current JSF application. In other words, it encodes the link using `ExternalContext.encodeActionURL(...)`. This is different to the JSF `h:outputLink` component, which encodes the URL as a resource URL using `ExternalContext.encodeResourceURL(...)`. If you are using `goLink` to link to a resource that is not a Faces View, then the URL must be encoded as a Resource URL. To do this, you must tell the Oracle JSF Portlet Bridge the link is not a Faces View by adding the following parameter to the URL:

```
javax.portlet.faces.ViewLink=false
```

For example:

```
<af:goLink text="goLink 1" id="gl1" destination="/mynonfaceslink?
javax.portlet.faces.ViewLink=false"/>
```

- **In-Protocol Resource Requests.** Within a JSF application running under the Oracle JSF Portlet Bridge, when a resource URL is created using `ExternalContext.encodeResourceURL(...)`, that resource URL is encoded in such a way that it is executed as an out-of-protocol resource request. This means that when the portlet consumer accesses that resource it does so by making a direct call to the target URL. It is also possible for resource URLs to be in-protocol. This means that the portlet consumer accesses the resource by making a WSRP web service call to the producer application, which in turn ultimately leads to the `serverResource` method being called on the portlet. It is then the portlet's responsibility to provide the resource which is sent back in the WSRP response. When that portlet is the Oracle JSF Portlet Bridge, it serves the resource by using a request dispatcher to forward to the target URL.

  Generally out-of-protocol resource requests are to be preferred as they do not incur the additional overhead of the web service call. In-protocol requests can also cause large amounts of memory to be allocated as the WSRP SOAP response is constructed, if the resource being served is large. In the case of the Oracle JSF Portlet Bridge, the only time that you would want to use in-protocol resources is when you need to share the portlet scoped HTTP session between the resource requests and the other portlet requests, such as portlet action or render requests.

To cause the Oracle JSF Portlet Bridge to encode resource URLs as in-protocol resource requests, add the following parameter to the URL, before `ExternalContext.encodeResourceURL(...)` is called:

```
javax.portlet.faces.InProtocolResourceLink=true
```

For example:

```
<af:goLink text="goLink 1" id="gl1" destination="/mynonfaceslink?
javax.portlet.faces.ViewLink=false&javax.portlet.faces.InProtocolResourceLink=true
"/>
```

12.3 Updating the Portlet Entry for a Portletized Page or Task Flow

If you edit a page or task flow that has been previously portletized, if those edits include updates to the task flow or page's parameters or events, then you must update the portlet entry for the page or task flow to keep the portlet in sync with the underlying page or task flow.

> **Note:**
>
> If you delete a page that has been portletized, you must manually remove the portlet entry for that page from the `portlet.xml` file.

To update the portlet entry for a portletized page or task flow:

1. In the Application Navigator, open the JSF application that contains the portletized page or task flow that you want to update.

2. Right-click the page or task flow, and choose **Update Portlet Entry.**

3. If the portlet entry is for an unbounded task flow, then a dialog prompts you to select the view activity that was used to create the portlet.

When you update the portlet entry for a portletized page or task flow, the portlet entry is updated as follows:

*   The **Create portlet events for contextual events** option is set to true.

*   No updates are made to the portlet name, display name, title, short title, or description.

*   Any `event-definition` elements (at the portlet application level) are created as necessary.

*   None of the existing `event-definition` elements are deleted or updated.

*   Any `public-render-parameter` elements (at the portlet application level) are created as necessary.

*   None of the existing `public-render-parameter` elements are deleted or updated.

*   The list of `supported-processing-event`, `supported-publishing-event`, and `supported-public-render-parameter` elements in the portlet entry are re-created to reflect the current events and input parameters of the page or task flow. This means that some existing portlet events and public parameters may be deleted and new ones added.

12.4 Testing a Portletized Page or Task Flow in JDeveloper

After you have portletized a page or task flow, you must test it to ensure that it is working correctly. Testing a portletized page or task flow requires the following steps:

1. Portletize the page or task flow to create a portlet entry in the `portlet.xml` file.

2. Deploy the portletized application to a portlet server.

3. Create an application to consume the portletized page or task flow.

4. Register the portletized application with the consumer application.

5. Create a page in the consumer application and add the portletized page or task flow to it.

6. Deploy the consumer application and display the page that contains the portletized page or task flow.

12.5 Testing a JSF Portlet Using the Integrated WebLogic Server

When you have created your JSF portlet you can test it using the Integrated WebLogic Server that comes packaged with JDeveloper.

Before You Begin

Before you test your portletized page or task flow, it is a good idea to first test that the page or task flow works correctly in the running ADF application. You can do this, using the Integrated WebLogic Server, by running the page or, for task flows, adding the task flow to a page and running that page.

For more information, see Running an ADF application in Integrated WebLogic Server in the *Developing Fusion Web Applications with Oracle Application Development Framework*.

To test a JSF portlet:

1. From the main menu, choose **Run** and then **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:

   ```
   IntegratedWebLogicServer started.
   ```

   For more information, see Managing the Integrated WebLogic Server.

2. You are now ready to run your Portlet Producer application on the Integrated WLS. Because your web application and Portlet Producer application are one and the same (the Portlet Producer application is your existing web application with additional portlet artifacts), you can run your web application as you normally would (see Running an ADF application in Integrated WebLogic Server in *Developing Fusion Web Applications with Oracle Application Development Framework*) or you can run your portlet following the instructions in How to Run a WSRP Portlet Producer on Integrated WebLogic Server.

3. When the application is deployed, you can view the WSRP Producer Test Page by going to:

   ```
   http://host:port/context-root/info
   ```

where:

- *host* is the server to which the application 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.

4. Once you have successfully deployed the application containing the portlet, you can register it as a portlet producer with any other application.

> **Note:**
>
> Because the Oracle JSF Portlet Bridge uses WSRP 2.0 features, you should register the producer using the **WSRP v2 WSDL** URL listed in the WSRP Producer Test Page.

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.

12.6 Deploying JSF Portlets to a WebLogic Managed Server

When you are satisfied that the application works correctly both as a web application and a portlet application, you can deploy it to your production Oracle WebLogic Managed Server.

Because your web application and portlet application are one and the same (the portlet application is your existing web application with additional portlet artifacts), deploying the web application also deploys the portlet producer application.

For information about how to deploy an application to a WebLogic Managed Server, see Deploying Fusion Web Applications in the *Developing Fusion Web Applications with Oracle Application Development Framework*.

> **Note:**
>
> Ensure that you deploy your application to a Java EE container with the Oracle Portlet Container installed. The Integrated WLS has the container 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
```

12.7 Using Events to Link JSF Portlets with Other Portlets

This section includes the following topics:

- About Linking JSF Portlets with Other Portlets

- How to Link a JSF Portlet to Another JSF Portlet

- How to Link a JSF Portlet to a JSR 286 Portlet

- How to Link a JSR 286 Portlet to a JSF Portlet

- What You May Need to Know About Portlet Events in the Oracle JSF Portlet Bridge

12.7.1 About Linking JSF Portlets with Other Portlets

If you portletize an Oracle ADF task flow that triggers a contextual event, the Oracle JSF Portlet Bridge creates a JSR 286 portlet event to wrap the contextual event. This means that when the portletized task flow is consumed on a page, it can be linked to other JSF portlets or to JSR 286 portlets.

The examples in this section use the following portlets:

- **Departments portlet**, a portletized ADF task flow that displays a list of departments. The list of departments can be restricted by location ID. This portlet raises a contextual event, departmentSelected, when a department row is selected in the portlet. The payload of the departmentSelected event is an object of type hr.Department (a Java object representing one row of the Departments table).

- **Employees portlet**, a portletized ADF task flow that displays a list of employees. The list of employes can be restricted by department ID.

- **Department Address portlet**, a JSR 286 portlet that displays the address of a specified department.

- **Department Locations portlet**, a JSR 286 portlet that displays a list of locations. This portlet raises a portlet event, named locationId with the namespace http://xmlns.oracle.com/adfm/contextualevent, when one of the listed locations is selected.

12.7.2 How to Link a JSF Portlet to Another JSF Portlet

The Departments task flow raises a contextual event, departmentSelected, when a department row is selected within the task flow. The event is ultimately raised by an eventBinding which is triggered from a row selection listener (see example below).

The payload of the event (an object of type hr.Department, a Java object representing one row of the departments table) is generated by using the customPayload attribute which points to the current row from the departments table iterator.

For more information about standard ADF contextual event techniques, like the ones used in this example, see About Creating Contextual Events in the *Developing Fusion Web Applications with Oracle Application Development Framework*.

```
<eventBinding Listener="org.apache.myfaces.trinidad.event.SelectionListener"
 id="DepartmentSelectedEvent">
 <events xmlns="http://xmlns.oracle.com/adfm/contextualEvent">
 <event name="departmentSelected"
 customPayLoad="#{bindings.departments.currentRow.dataProvider}"/>
 </events>
</eventBinding>
```

When the Departments task flow is portletized, using the steps described in How to Create a JSF Portlet Based on a Task Flow, a `portlet.xml` file is created containing a definition for a portlet event that is used to transfer the corresponding contextual event during portlet communication with the portlet over WSRP:

```
<portlet id="adf_taskflow_WEB_INF_departments_xml">
 ...
 <supported-publishing-event id="departmentSelected">
 <qname xmlns:x="http://xmlns.oracle.com/adfm/contextualEvent">
 x:departmentSelected
 </qname>
 </supported-publishing-event>
 ...
</portlet>
```

There is also a corresponding event definition in `portlet.xml` for the Departments portlet's `departmentSelected` event:

```
<event-definition id="departmentSelected">
 <qname xmlns:x="http://xmlns.oracle.com/adfm/contextualEvent">
 x:departmentSelected
 </qname>
 <value-type>
 oracle.portlet.bridge.adf.lifecycle.ADFmPayloadWrapper
 </value-type>
</event-definition>
```

> **Note:**
>
> The namespace for contextual events that are converted into portlet events is always:
>
> `http://xmlns.oracle.com/adfm/contextualEvent`
>
> The payload type is always:
>
> `oracle.portlet.bridge.adf.lifecycle.ADFmPayloadWrapper`
>
> The `ADFmPayloadWrapper` is used to ensure that any serializable contextual event payload is wrapped in a JAXB marshallable wrapper. It also ensures that a fixed type can be declared for the portlet event, even though it is not possible to know the payload type until the event is raised. The payload is unwrapped when extracting the contextual event from the portlet event.

The Employees task flow uses standard ADF techniques to handle a `departmentSelected` event. This is in the form of a `methodAction` binding that invokes a method on a data control, delivering the event payload from the `departmentSelected` event as a parameter to that method. The example below shows the definition of the `methodAction` binding.

```
<methodAction id="updateTableForDepartmentId" RequiresUpdateModel="true"
 Action="invokeMethod" MethodName="updateTableForDepartmentId"
 IsViewObjectMethod="false" DataControl="EmployeesDataControl"
 InstanceName="EmployeesDataControl.dataProvider">
 <NamedData NDName="departmentIdPayLoad" NDType="hr.Department"/>
</methodAction>
```

The `updateTableForDepartmentId` method takes the `hr.Department` object, passed as the payload of the `departmentSelected` event, and filters the data

queried from the `EmployeesDataControl` so that it shows only employees who are in the selected department. It also takes steps to ensure that the iterator is re-executed and that the table view component is marked as needing to be re-rendered in this request.

The Employees task flow includes an event map to specifically map the incoming event onto the appropriate `methodAction` when the task flow is portletized:

```
<eventMap xmlns="http://xmlns.oracle.com/adfm/contextualEvent">
 <event name="departmentSelected">
 <producer region="*">
 <consumer region="" handler="updateTableForDepartmentId"
 handleCondition="${ADFPortletBridge.bridgeRequest}">
 <parameters>
 <parameter name="p1" value="#{payLoad}"/>
 </parameters>
 </consumer>
 </producer>
 </event>
</eventMap>
```

Note:

The `<producer region="*">` attribute accepts events from any region. This is because when the task flow is running as a portlet, with the event coming into the portlet from a remote application, the remote application does not have any knowledge of the producer region.

The EL expression within `handleCondition="$
{ADFPortletBridge.bridgeRequest}"` evaluates to `true` if the current request is running under the Oracle JSF Portlet Bridge. This ensures that this event map is used only when the task flow is running as a portlet. It is not essential to have this `handleCondition`. However it can be useful to be able to define a separate `eventMap` which is used only when the task flow is running as a portlet.

When the two portletized task flows are added to a page, as long as the name of the event triggered by the Departments portlet is the same as that expected by the Employees portlet, no further coding is required; the portlets are automatically linked and when a department is selected in the Departments portlet, the Employees portlet is updated to display employees belong to the selected department.

Figure 12-1   Linked JSF Portlets

You may choose to not use automatic event listening, for example, if you have multiple portlets on the same page that use the same events and you wish to control the flow of events between them. Or perhaps the names of the contextual events used by the JSF portlets that you want to link do not match. In this case you must explicitly create an event map in the page definition of the page containing the portletized task flows. This event map links the portlet events of the portlets that you want to wire together. The example below shows how this would work in our example if the Employees task flow was expecting an event named `departmentId` instead of `departmentSelected`.

```
<eventMap xmlns="http://xmlns.oracle.com/adfm/contextualEvent">
 <event name="departmentSelected">
 <producer region="EventSampleProducerdepartments1_1">
 <consumer handler="EventSampleProduceremployees1_1.
 {http://xmlns.oracle.com/adfm/contextualEvent}departmentId"/>
 </producer>
 </event>
</eventMap>
```

Note:

In the case of portletized task flows, the `handler` attribute must specify not only the specific binding object that is to handle the event, it must also specify the full QName of the portlet event, using the format `<portletBindingId>.<eventQName>`. The namespace does not have to be the ADFm namespace; it can be any portlet event, provided the payload is compatible.

12.7.3 How to Link a JSF Portlet to a JSR 286 Portlet

As well as linking JSF portlets to other JSF portlets, you can also link a JSF portlet to a regular JSR 286 portlet. To do this, you must ensure that the JSR 286 portlet explicitly declares that it can to receive contextual events.

For example, we can link the Departments task flow from our previous example to a Department Address portlet, a JSR 286 portlet, to display the address of a selected department.

For automatic event wiring to work, the Department Address portlet must declare that it can receive the contextual event from the Department portlet. It can do this in two ways:

- The Department Address portlet directly supports the `departmentSelected` contextual event:

```
<supported-processing-event id="departmentSelected">
 <qname xmlns:x="http://xmlns.oracle.com/adfm/contextualEvent">
 x:departmentSelected
 </qname>
</supported-processing-event>
```

- The event definition for the Department Address portlet's event, `department`, provides an alias to associate it with the `departmentSelected` contextual event:

```
<event-definition id="department">
 <qname xmlns:x="http://xmlns.oracle.com/portlet/EventSample">
 x:department
 </qname>
 <alias xmlns:x="http://xmlns.oracle.com/adfm/contextualEvent">
 x:departmentSelected
 </alias>
 <value-type>hr.Department</value-type>
</event-definition>
```

Note:

The namespace for portlet events used to carry contextual event is always:

```
http://xmlns.oracle.com/adfm/contextualEvent
```

The namespace is used in the event declarations or alias given above. This namespace indicates to the consumer that the portlet wishes to receive a contextual event with the given name as a portlet event. This is essential for automatic event wiring to work.

The Department Address portlet includes code in its `processEvent` implementation to handle the event. This method looks for either the `departmentSelected` or the `department` event with the alias as defined above. The payload type for both of these events is `hr.Department`. However, the method shown in the example below illustrates how to use the `ADFmPayloadWrapper` to unwrap the payload. `ADFmPayloadWrapper` is the event type that the Oracle JSF Portlet Bridge uses by default when raising events.

```
public void processEvent(EventRequest request, EventResponse response)
 throws PortletException, IOException
 {
```

```
response.setRenderParameters(request);

Event event =request.getEvent();
 String eventName =event.getName();
 if ("departmentSelected".equals(eventName) ||
"department".equals(eventName))
 {
Object payload =event.getValue();
 if (payload instanceof ADFmPayloadWrapper)
 {
payload =((ADFmPayloadWrapper)payload).getPayload();
 }
if (payload instanceof Department)
 {
Department department =(Department)payload;
 int locationId =department.getLocation();
 response.setRenderParameter("locationId", Integer.toString(locationId));
 }
}
}
```

When these two portlets are displayed together on a page, selecting a department in the Departments portlet automatically updates the Department Address portlet to display the address of the selected department.

Figure 12-2   JSF Portlet Linked to a JSR 286 Portlet
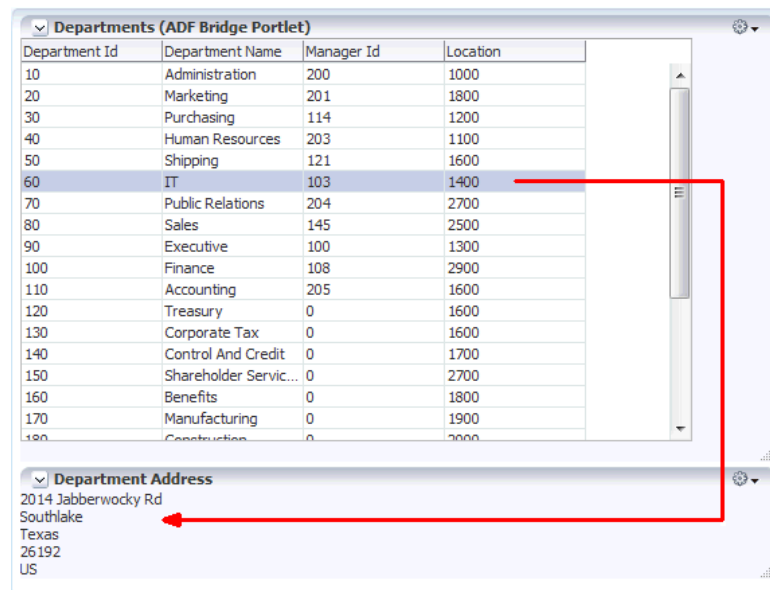
If the Department Address portlet does not explicitly support the `departmentSelected` event or define an appropriate alias, you can still link the two portlets by creating an event map in the page definition:

```
<eventMap xmlns="http://xmlns.oracle.com/adfm/contextualEvent">
 <event name="departmentSelected">
 <producer region="EventSampleProducerdepartments1_1">
 <consumer handler="DepartmentAddress1_1.
 {http://xmlns.oracle.com/portlet/EventSample}department"/>
 </producer>
 </event>
</eventMap>
```

The result of this event mapping is that when the `departmentSelected` contextual event is raised by the `EventSampleProducerdepartments1_1` portlet binding, the event is delivered to the `DepartmentAddress1_1` portlet binding, which in turn sends the event on to the portlet using the portlet event, `department`.

> **Note:**
>
> The payload of the `departmentSelected` contextual event is a `hr.Departments` object. This is the same payload type that the portlet's `department` portal event is expecting, so that matches and everything works correctly. It is also possible for the portlet to declare that it expects a payload of type `ADFmPayloadWrapper`. In this case, the portlet consumer detects this and automatically wraps the contextual event payload in an `ADFmPayloadWrapper` object.

12.7.4 How to Link a JSR 286 Portlet to a JSF Portlet

You can also send portlet events from a JSR 286 portlet to a JSF portlet. The portlet event is converted into a contextual event by the Oracle JSF Portlet Bridge and is then delivered into the producer ADF application to be consumed using standard contextual event techniques.

The Departments task flow contains a `methodAction` binding that binds to a method on the `DepartmentsDataControl`:

```
<methodAction id="updateTableForLocationId" RequiresUpdateModel="true"
 Action="invokeMethod" MethodName="updateTableForLocationId"
 IsViewObjectMethod="false"
 DataControl="DepartmentsDataControl"
 InstanceName="DepartmentsDataControl.dataProvider">
 <NamedData NDName="locationId" NDType="int"/>
</methodAction>
```

The `updateTableForLocationId` method receives, as a parameter, a location ID that causes the list of departments returned by the data control to be filtered by that location ID.

The task flow contains an event map to map the contextual event onto the appropriate `methodAction`:

```
<eventMap xmlns="http://xmlns.oracle.com/adfm/contextualEvent">
 <event name="locationId">
 <producer region="*">
 <consumer region="" handler="updateTableForLocationId"
 handleCondition="${ADFPortletBridge.bridgeRequest}">
 <parameters>
 <parameter name="p1" value="#{payLoad}"/>
 </parameters>
 </consumer>
 </producer>
 </event>
</eventMap>
```

In the Department Locations portlet, when a location is selected, the `processAction` method raises a portlet event, `locationId`, giving the location ID of the selected location:

```
public void processAction(ActionRequest request, ActionResponse response)
 throws PortletException
 {
String locationId =request.getParameter("locationId");
 Location location =getLocation(Integer.parseInt(locationId));

if (location != null)
 {
//QName matches the event declared as a supported-publishing-event in
 //portlet.xml for this portlet.

//LocationId event. Raised in the ContextualEvent namespace makes it readily
 //consumable by ADF Bridge portlets. Likewise we wrap with the
 //ADFmPayloadWrapper payload object that the ADF Bridge expects.
 response.setEvent(new QName("http://xmlns.oracle.com/adfm/contextualEvent",
 "locationId"),
 new ADFmPayloadWrapper(Integer.valueOf(location.getLocationId())));
 }
}
```

When you drop the portlets on a page, selecting a location in the Department Location portlet automatically updates the Departments portlet to display the departments at the selected location.

Figure 12-3   JSR 286 Portlet Linked to a JSF Portlet
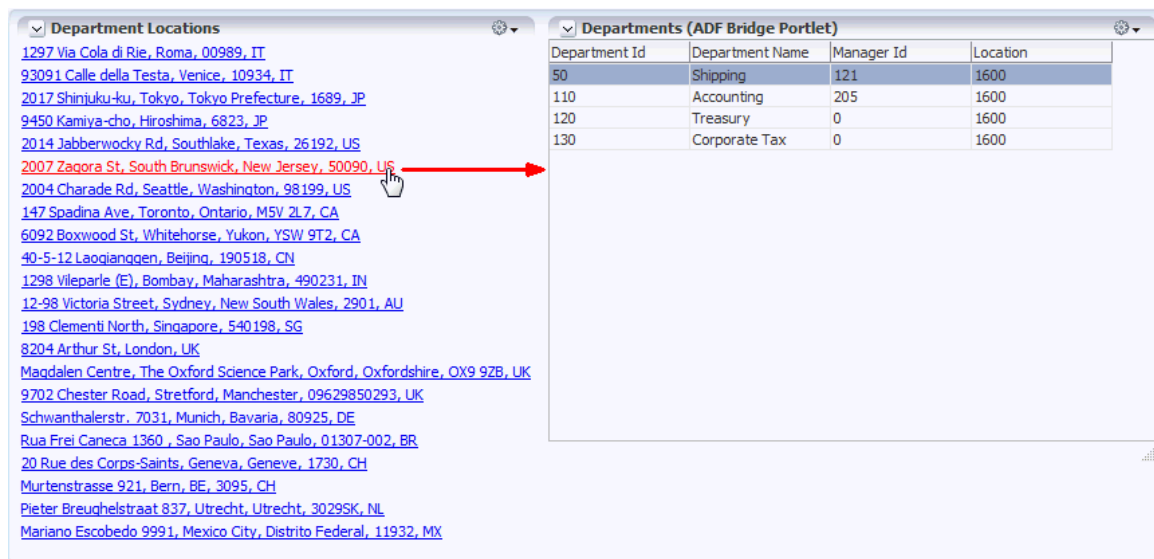
If the two portlets do not declare events with matching names (for example the contextual event from the Departments portlet is called `locId`), or if automatic event listening is disabled, you can still link the two portlets by creating an event map in the page definition to map the contextual event from the Departments portlet to the portlet event from the Department Location portlet:

```
<eventMap xmlns="http://xmlns.oracle.com/adfm/contextualEvent">
 <event name="locationId">
 <producer region="DepartmentLocations1_1">
 <consumer handler="DepartmentsADFBridgePortlet1_1.
 {http://xmlns.oracle.com/adfm/contextualEvent}locId">
 </consumer>
 </producer>
 </event>
</eventMap>
```

12.7.5 What You May Need to Know About Portlet Events in the Oracle JSF Portlet Bridge

This section includes the following topics:

- What You May Need to Know About Using a Serialized Type for the Contextual Event Payload

- What You May Need to Know About Raising Undeclared Events

- What You May Need to Know About Partial Page Rendering

12.7.5.1 What You May Need to Know About Using a Serialized Type for the Contextual Event Payload

Part of the JSR 286 event's payload is the wrapped contextual event payload, which is serialized. Therefore, while contextual event payloads can be any object type, when you are sending contextual events via WSRP, only serialized types are supported.

When this wrapped contextual event is delivered, the contextual event payload must be deserialized on the consumer before the contextual event can be forwarded into the consumer application. To deserialize the event, the payload class must be available on

the consumer as well as on the producer. This condition is automatically met when the original payload is a Java Runtime Environment class, such as `java.lang.String`.

12.7.5.2 What You May Need to Know About Raising Undeclared Events

If your Oracle ADF task flow includes ADFm events that do not have corresponding portlet events declared in the `portlet.xml` file, you can enable the Oracle JSF Portlet Bridge to raise these undeclared events.

To raise undeclared events, set the following container runtime option for the portletized task flow:

```
<portlet id="Application5untitled1jspx1_1">
 ...
 <container-runtime-option>
 <name>oracle.portlet.bridge.adf.raiseUndeclaredContextualEvents</name>
 <value>true</value>
 </container-runtime-option>
 ...
</portlet>
```

Setting this option to `true` means that any ADFm event raised is forwarded on as a portlet event. If this option is not provided or is set to `false`, then only events with corresponding portlet event declarations are forwarded. For information about setting container runtime options, see Setting Container Runtime Options for Individual Portlets.

You can also enable undeclared portlet events to automatically be forwarded on from the portlet binding in the consumer application as ADFm events.

In the portlet binding, set the `raiseUndeclaredContextualEvent` attribute to `true`, for example:

```
<portlet id="Application5untitled1jspx1_1"
 portletInstance="/oracle/adf/portlet/WsrpPortletProducer4/ap/
 Application5untitled1jspx_2943bd23_012e_1000_8004_0aa7c0849010"
 class="oracle.adf.model.portlet.binding.PortletBinding"
 retainPortletHeader="false"
 listenForAutoDeliveredPortletEvents="true"
 listenForAutoDeliveredParameterChanges="true"
 raiseUndeclaredContextualEvents="true"
 xmlns="http://xmlns.oracle.com/portlet/bindings"/>
```

Setting this option to `true` means that if a portlet event is received, a corresponding ADFm event is raised, even if there is no event declaration in the portlet binding. The name of the ADFm event is directly taken from the QName of the portlet event. This is so that ADFm events raised from the Oracle JSF Portlet Bridge have the same name as that used when the event was raised on the remote application.

12.7.5.3 What You May Need to Know About Partial Page Rendering

The Oracle JSF Portlet Bridge uses portlet events to transport contextual events to and from the consumer application. However, partial page rendering (PPR) requests are implemented using Portlet 2.0 resource requests. It is not possible to send or receive portlet events in resource requests. Therefore proprietary mechanisms are used to transport the contextual events from the JSF Portlet to the consumer application in the case of PPR requests.

13

Building Standards-Based Java Portlets Using JSR 286

This chapter describes how to build standards-based Java portlets using the Java Portlet Specification, JSR 286.

This chapter includes the following topics:

- About Building Standards-Based Java Portlets Using JSR 286
- Creating a JSR 286 Java Portlet
- Developing JSR 286 Java Portlets
- Testing JSR 286 Portlets
- Migrating WebLogic Portal Portlets to WebCenter Portal
- Files Related to JSR 286 Portlets

13.1 About Building Standards-Based Java Portlets Using JSR 286

WebCenter Portal supports standards-based Java portlets built using the JSR 286 standard.

JSR 286 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 286 defines container services that provide:

- A portlet API for coding portlet functionality
- The URL-rewriting mechanism for creating user interaction within a portlet container
- The security and personalization of portlets
- Interportlet communication using portlet events and public render parameters

For more information, see the JSR 286 specification at:

http://jcp.org/en/jsr/detail?id=286

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:

- 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

WSRP provides communication between a portlet client and a portlet container running on a remote server. JSR 286 describes the Java Portlet API for running portlet producer applications. 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 Resource Palette or Application Resources pane of the Application Navigator.

Figure 13-1 shows the architecture of the WSRP specification with JSR 286 portlets.

Figure 13-1   WSRP Specification Architecture
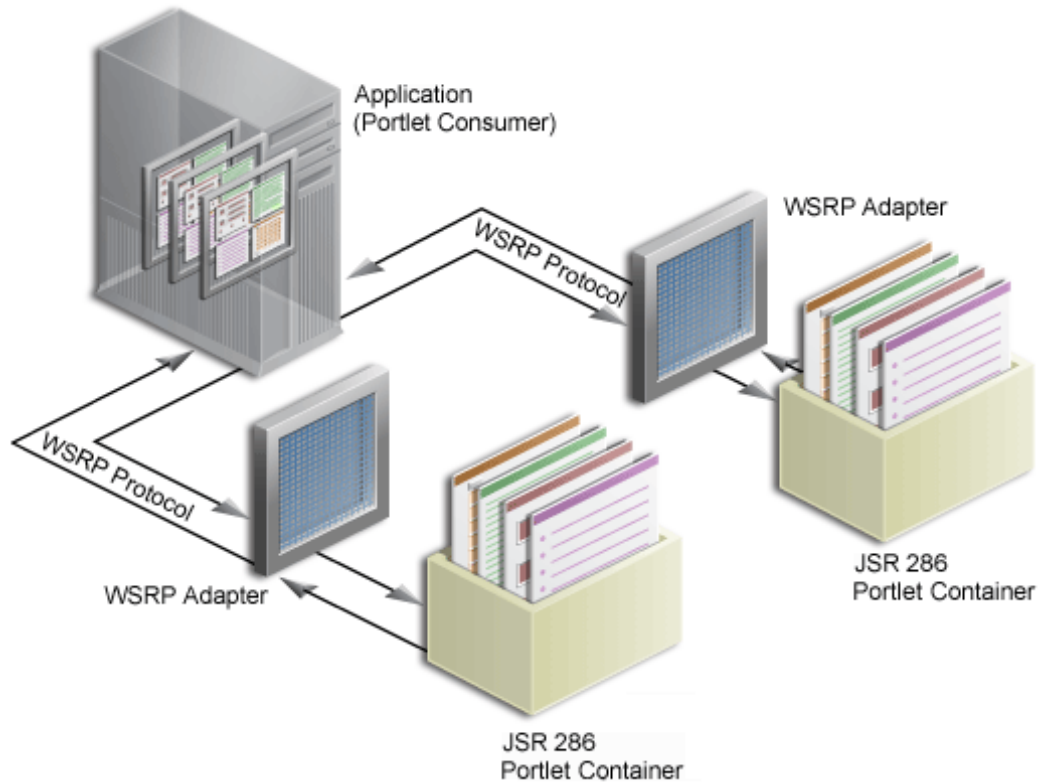
13.2 Creating a JSR 286 Java Portlet

WebCenter Portal provides a wizard, available in JDeveloper, for quickly and easily creating the initial framework of your JSR 286 standards-based Java portlets.

In the Create JSR 286 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 286 Java portlet using the JDeveloper wizard:

1. In JDeveloper, open the portlet producer application under which you want to create your portlet, or create a new portlet producer application.

   For information about how to create a portlet producer application, see Portlet Producer Applications.

> **Note:**
>
> If you do not use the WebCenter Portal - Portlet Producer Application template to create the portlet producer application, you must manually add the appropriate portlet building technology scopes to the application.

2. Right-click the project under which you want to create your portlet (for example **Portlets**), and choose **New.**

3. In the New Gallery (Figure 13-2), expand **Web Tier,** select **Portlets** and then **Standards-based Java Portlet (JSR 286),** and click **OK.**

Figure 13-2   Standards-based Java Portlet (JSR 286) Option in the New Gallery
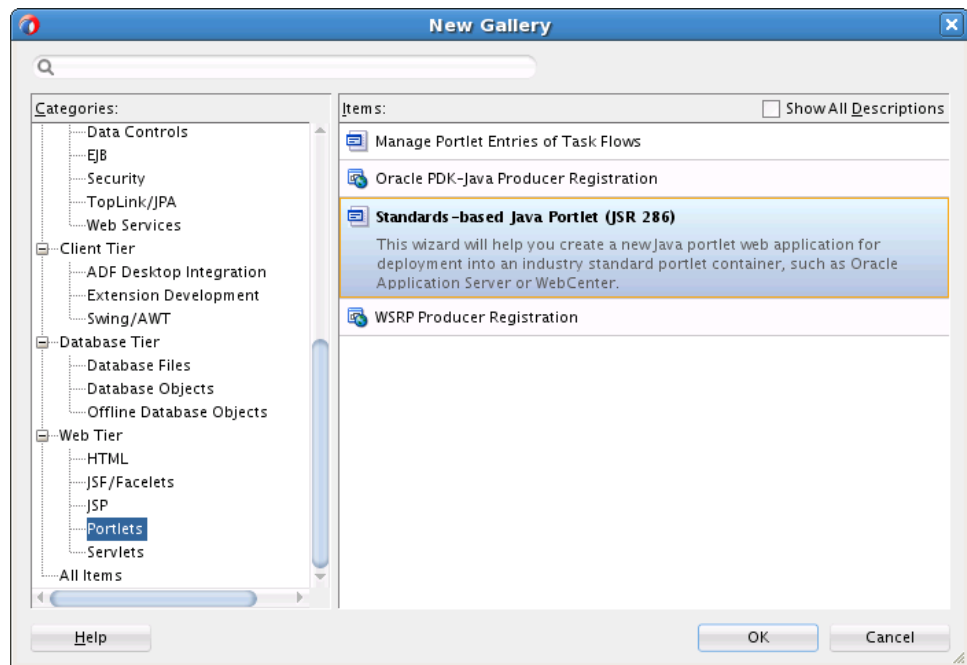
> **Tip:**
>
> If the project already contains JSR 286 portlets, you can also create a new portlet by:
>
> - Right-clicking `portlet.xml` and choosing **Add Portlet**.
>
> - Opening `portlet.xml`, clicking the **Design** tab, and clicking the **Add** icon on the **Portlets** tab.

4. On the General Portlet Information page of the Create JSR 286 Java Portlet wizard, replace the default name provided in the **Name** field with one that better describes the purpose of the portlet.

5. 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.

6. From the **Package** drop-down list, select the package in which to create the class, or enter a package name.

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.

7. From the **Language** drop-down list, select the default language that your portlet supports. The wizard uses English by default.

8. 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.

   Edit mode enables users to personalize the portlet at runtime. For more information, see Edit Mode.

   If you select this option, you can specify implementation details for the portlet's Edit mode later on in the wizard.

9. Click **Next**.

10. On the Additional Portlet Information page, in the **Portlet Title** field, enter a descriptive title for your portlet.

    The portlet title is displayed in the resource catalog, 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.

    > **Tip:**
    >
    > The **Display Name**, **Short Title**, **Description**, and **Keyword** attributes are not implemented in WebCenter Portal. You do not need to enter any values for these fields unless your portlet is likely to be consumed by other applications.

11. 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.

12. On the Content Types and Portlet Modes page, in the **Content Types and Portlet Modes** list, select **view**.

13. In the Implementation Method section, select how you want to implement View mode for your portlet (for more information about View mode, see View Mode):

    - 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 JSF page in the corresponding field, or accept the default.

      If you choose this option, the portlet implementation class is created as a subclass of `oracle.portlet.bridge.adf.application.ADFBridgePortlet`, instead of as a subclass of `javax.portlet.GenericPortlet`. That is, the wizard generates a portlet producer application which uses the Oracle JSF Portlet Bridge. For more information about the Oracle JSF Portlet Bridge, see Creating Portlets from JSF Applications Using the Oracle JSF Portlet Bridge.

- Select **Map to Path** to map the portlet mode to a web resource in the portlet producer 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 necessary.

  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 through 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.

  Tip:

  If you want this portlet mode to support a different content type, for example `text/xml`, see Step 16.

14. If you selected **Enable users to edit portlet content** on the first page of the wizard, select **edit** and then select the implementation method for Edit mode (for information about Edit mode, see Edit Mode), as described in step 13.

15. To implement another portlet mode for your portlet:

    a. In the **Content Types and Portlet Modes** list, select an existing mode (for example, **view**) under the appropriate content type (for example, **text/html**).

    b. Click **Add**.

    c. In the Portlet Modes dialog, move the required mode or modes to the **Selected** list and click **OK**.

       The Portlet Modes dialog lists the standard modes supported by JSR 286 and the extended modes supported by WebCenter Portal.

       For more information, see Portlet Modes.

    d. Select each of the portlet modes and specify the implementation method to use for rendering content, as described in step 13.

16. The content type identifies what type of content the portlet supports. Portlets can support multiple content types. By default, portlets created using the Create JSR 286 Java Portlet wizard support the `text/html` content type.

    To add a different content type:

    a. In the **Content Types and Portlet Modes** list, select an existing content type (for example, **text/html**).

    b. Click **Add**.

    c. In the Content Types dialog, move the required content types to the **Selected** list and click **OK**.

       - **text/html**—The portlet supports text encoded with HTML. This is the default selection.

       - **text/xml**—The portlet supports text encoded with XML.

- **text/plain**—The portlet supports plain, unencoded text.

- **text/vnd.oracle.mobilexml**—The portlet supports text encoded with Oracle Mobile XML. Note, however, that WebCenter Portal does not support Oracle Mobile XML.

- **application/xhtml+xml**—The portlet supports text encoded with XHTML.

- **application/xml**—The portlet supports any XML content, including XHTML.

17. Click **Next.**

If you selected **Enable users to edit portlet content** on the General Portlet Information page earlier in the wizard, then you can create portlet preferences to enable users of the portlet to specify values for the portlet at runtime. Go to step 18.

If you did not select this option, go to step 24.

18. On the Customization Preferences page, click **Add** to add a new portlet preference to the portlet.

By default, the wizard includes a portlet preference to enable users to customize the portlet title.

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 and cannot contain spaces.

20. In the **Default Values** field, enter one or more default values for the new preference. Separate multiple values with commas.

21. Select the **Translate Preference** check box if you want to be able to make the preference available in different languages.

For example, if the portlet is likely to be consumed by a multilingual portal you want to ensure that the preference displays in the appropriate language.

If you enable this option, then entries for the preference are added to the resource bundle class that JDeveloper generates for the portlet.

> **Tip:**
>
> Edit the resource bundle to provide translations for the preference name and default values. The name will almost always require translating, but the default values may not, for example, if the value is an integer.

22. Click **OK**.

23. Repeat the preceding steps to add more preferences. When you are done click **Next.**

24. On the Security Roles page, 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`).

You can create new security roles for the application by editing `web.xml`. For more information, see the JDeveloper online help.

25. Click **Next.**

26. On the Caching Options page, select **Cache Portlet** to enable expiry-based caching for your portlet.

For more information about expiry-based caching, see Portlet Performance and Implementing Expiry-Based Caching in JSR 286 Portlets.

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 information about how to implement validation-based caching, see Implementing Validation-Based Caching in JSR 286 Portlets.

27. If you chose to cache the portlet, in the Default Expiry Conditions section, select:

- **Cache Content Expires After []seconds** to expire the cached portlet content after a certain amount of time. Specify the time limit in the corresponding 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. On the Initialization Parameters page, you can add initialization parameters to your portlet.

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.

For example, you might want to turn on some kind of debugging mode for the portlet or display different menu options in different environments.

a. Click **New** to add a new initialization parameter to the portlet.

b. In the newly added row, double-click each field to provide a **Name**, default **Value**, and **Description** for the parameter.

    c. Repeat these steps to add more initialization parameters.

    d. When you are done, click **Finish** to create the portlet.

When you use the Create JSR 286 Java Portlet wizard, JDeveloper generates a default implementation of the portlet. Specifically, the following files are created:

- Java classes:

  - *portletName*.java is invoked by the portlet container and contains all the methods required by the portlet standards.

  - *portletName*Bundle.java contains all the translation strings for the portlet.

  - *portletName*Backing.java contains the JSF backing bean for the JSF pages that implement portlet modes. This class is created if you implement portlet modes as ADF-Facelets JSP pages.

- portlet.xml is the portlet deployment descriptor file for the application.

- web.xml is the web deployment descriptor 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 JSF (.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 are created, but the code for the portlet mode resides in the portlet's Java class.

- faces-config.xml is a configuration file that registers an application's resources, such as custom validators and managed beans and describes the page flow of the application. This file is created if you implement portlet modes as ADF-Facelets JSF pages.

You can see all the files in the Application Navigator, as shown in .

Figure 13-3   Files Generated for a JSR 286 Java Portlet
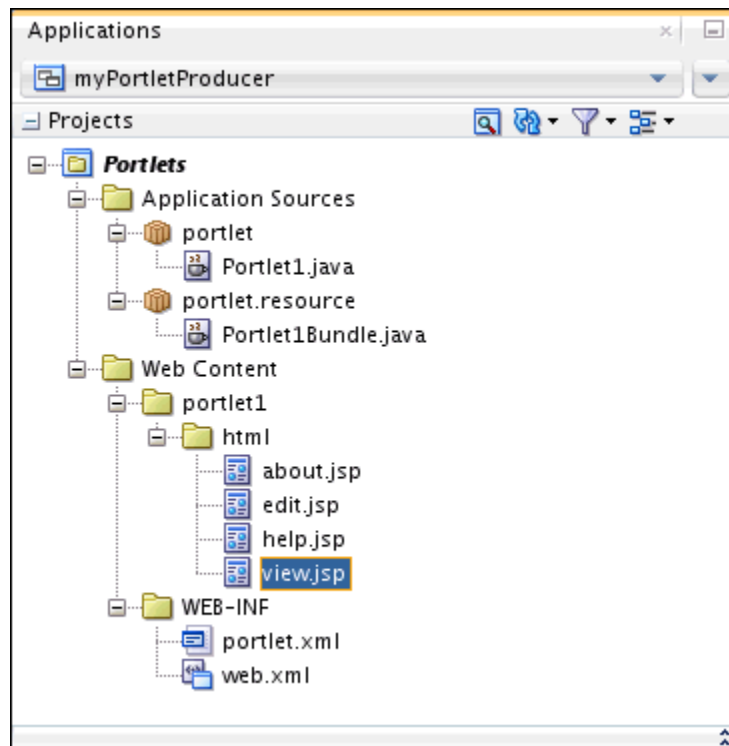
13.3 Developing JSR 286 Java Portlets

When you have built your initial portlet implementation using the Create JSR 286 Java Portlet wizard, the next step is to create the code that drives the portlet content and behavior. Because JSR 286 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, including:

http://jcp.org/en/jsr/detail?id=286

This section includes the following topics:

- Example Portlet Deployment Descriptor File

- How to Edit the Portlet Deployment Descriptor File

- Portlet Modes for JSR 286 Portlets

- How to Add Custom Portlet Modes to JSR 286 Portlets

- How to Access User Information in JSR 286 Portlets

- How to Customize the Runtime Environment for JSR 286 Portlets

- How to Use Public Render Parameters in JSR 286 Portlets

- How to Use Portlet Events in JSR 286 Portlets

- How to Add Portlet Preferences to JSR 286 Portlets

- How to Use Portlet Filters in JSR 286 Portlets

- How to Implement Interportlet Communication Across Different Pages

- How to Enhance JSR 286 Portlet Performance with Caching

- How to Implement Rewritten URLs for Resource Proxy

- How to Implement Stateless Resource Proxying

- How to Manage the Persistence Store for JSR 286 Portlets

13.3.1 Example Portlet Deployment Descriptor File

The following example shows an example portlet deployment descriptor file.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<portlet-app version="2.0"
 xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd
 http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd"
 xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <portlet id="1328624289503">
 <portlet-name>myPortlet</portlet-name>
 <display-name xml:lang="en-US">Portlet1</display-name>
 <display-name xml:lang="en">English Display Name</display-name>
 <portlet-class>portlet.Portlet1</portlet-class>
 <expiration-cache>300</expiration-cache>
 <supports>
 <mime-type>text/html</mime-type>
 <portlet-mode>edit</portlet-mode>
 </supports>
 <supported-locale>en-US</supported-locale>
 <resource-bundle>portlet.resource.Portlet1Bundle</resource-bundle>
 <portlet-info>
 <title>This Is My Portlet</title>
 <short-title>My Portlet</short-title>
 <keywords/>
 </portlet-info>
 <portlet-preferences>
 <preference>
 <name>portletTitle</name>
 </preference>
 </portlet-preferences>
 </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>
```

13.3.2 How to Edit the Portlet Deployment Descriptor File

The portlet deployment descriptor file for the application, `portlet.xml`, specifies the portlet resources in the application. After you have created your JSR 286 portlet using the wizard, you can edit the `portlet.xml` file to edit the portlet resources.

When you open the `portlet.xml` file in JDeveloper, you can edit the source code or you can use the Overview Editor to edit portlet resources without having to manually modify it in the Source view.

To edit the portlet deployment descriptor file:

1. In the Application Navigator, open the portlet producer application.

2. Expand the portlet project (for example, **Portlets**).

3. Expand the **Web Content** node and then the **WEB-INF** node.

4. Right-click **portlet.xml** and choose **Open**.

5. Click the **Design** tab to open the Overview Editor for `portlet.xml`.

   > **Tip:**
   >
   > If you prefer to edit the source code directly, click the **Source** tab.

6. The Overview Editor for `portlet.xml` contains the following tabs:

   - **Application**—Use to specify general information for the portlet producer application. These properties apply to all portlets within the application.

   - **Portlets**—Use to specify information for individual portlets within the application. For example, you can specify the portlet events and public render parameters supported by a portlet.

   - **Events**—Use to create and manage portlet events for use by all portlets in the application. For more information, see How to Use Portlet Events in JSR 286 Portlets.

   - **Parameters**—Use to create and manage public render parameters for use by all portlets in the application. For more information, see How to Use Public Render Parameters in JSR 286 Portlets.

   - **Filters**—Use to specify filter information for the portlets in the application. For more information, see How to Use Portlet Filters in JSR 286 Portlets.

   - **Filter Mappings**—Use to assign portlet filters to individual portlets. For more information, see How to Use Portlet Filters in JSR 286 Portlets.

13.3.3 Portlet Modes for JSR 286 Portlets

The standard modes supported by JSR 286 are:

- View

- Edit

- Help

WebCenter Portal supports the following additional custom modes for JSR 286 portlets:

- About

- Config

- Edit Defaults

- Preview

- Print

13.3.4 How to Add Custom Portlet Modes to JSR 286 Portlets

In the Create JSR 286 Java Portlet wizard, you add portlet modes by adding them to a list on the Content Types and Portlet Modes page. For more information about using the wizard, see Creating a JSR 286 Java Portlet.The wizard enables you to implement the standard modes supported by JSR 286 and the extended modes provided by WebCenter Portal.

After the initial creation of the portlet, you can also define your own custom portlet modes.

The principles of implementing portlet modes are the same for all modes.

To add a custom portlet mode:

1. Open the Overview Editor for the `portlet.xml` file.

   For more information, see How to Edit the Portlet Deployment Descriptor File.

2. Click the **Application** tab, if necessary.

3. Expand the Custom Modes section, if necessary.

   This section lists all the custom modes available to the portlets within the application. It includes the WebCenter Portal extended portlet modes.

4. Click the **Add Custom Mode** icon to create a new row for the portlet mode.

5. In the **Portlet Mode** field, enter the name of the portlet mode.

   The name must be unique within the application.

6. In the **Description** field, enter a brief description to explain the purpose of the mode.

7. Deselect the **Portal Managed** check box if applications that consume the portlet do not need to be aware of the portlet mode, that is, the mode is used internally by the portlet.

8. Save the `portlet.xml` file.

9. After adding the custom portlet mode to the application, you must then create the code for the mode in the portlet implementation file.

13.3.5 How to Access User Information in JSR 286 Portlets

You can create user attributes in the portlet producer application to access commonly used user information, such as `user.login.id` or `user.name.family`. At runtime,

these user attributes are mapped to the actual attributes of the current user. Portlets can use this information to obtain information about the current user.

To access user information:

1. Open the Overview Editor for the `portlet.xml` file.

   For more information, see How to Edit the Portlet Deployment Descriptor File.

2. Click the **Application** tab, if necessary.

3. Expand the User Attributes section, if necessary.

4. Click the **Add User Attribute** icon to create a new row for the attribute.

5. In the **Name** field, enter the name of the user attribute, for example `user.login.id`.

   For a list of attribute names, see User Information Attribute Names appendix in the Java Portlet Specification.

   > **Tip:**
   >
   > For a list of the user information attributes supported by WebCenter Portal, see Mapping User Attributes to LDAP Directories in *Securing Applications with Oracle Platform Security Services*.

6. In the **Description** field, enter a description for the user attribute.

7. Save the `portlet.xml` file.

   At runtime, the portlet container maps the defined user attributes to the appropriate values. For example, if the current user is John Doe, then the `user.name.family` user attribute is mapped to Doe. Portlets can obtain a `Map` object containing the user attributes from the `PortletRequest` interface.

13.3.6 How to Customize the Runtime Environment for JSR 286 Portlets

When a portlet is run, the portlet container provides the runtime environment and provides an interface between the portlet producer application and the portlet.

Container runtime options provide a way to customize the behavior of the portlet container and therefore customize the runtime environment.

This section includes the following topics:

- Supported Container Runtime Options
- Setting Container Runtime Options for All Portlets in an Application
- Setting Container Runtime Options for Individual Portlets

13.3.6.1 Supported Container Runtime Options

Table 13-1 lists the container runtime options supported by the WebCenter Portal portlet container.

For more information about the JSR 286 container runtime options, see the JSR 286 specification at:

http://jcp.org/en/jsr/detail?id=286

Table 13-1   Supported Container Runtime Options

Container Runtime Option	Supported by	JSR 286 Standard
javax.portlet.actionScopedRequestAttributes	Application Portlet	Yes
javax.portlet.escapeXml	Application Portlet	Yes
java.portlet.renderHeaders	Not supported	Yes
javax.portlet.servletDefaultSessionScopejavax.portlet.servletDefaultSessionScope	Application Portlet	Yes
com.oracle.portlet.allowEventPayloadsWithoutJAXBBinding	Application	No
com.oracle.portlet.allowWsrpExport	Application	No
com.oracle.portlet.compatibilityMode	Application Portlet	No
com.oracle.portlet.defaultProxiedResourceRequiresWsrpRewrite	Application Portlet	No
com.oracle.portlet.defaultServedResourceRequiresWsrpRewrite	Application Portlet	No
com.oracle.portlet.disallowResourceServing	Application Portlet	No
com.oracle.portlet.escapeXmlEncodeUrls	Application Portlet	No
com.oracle.portlet.eventPayloadsXmlType	Application	No
com.oracle.portlet.excludedActionScopeRequestAttributes	Application Portlet	No
com.oracle.portlet.externalScopeRequestAttributes	Application Portlet	No
com.oracle.portlet.importCssToIFrame	Application Portlet	No
com.oracle.portlet.minimumWsrpVersion	Application Portlet	No
com.oracle.portlet.offerPortletOverWsrp	Application Portlet	No

Table 13-1   (Cont.) Supported Container Runtime Options

Container Runtime Option	Supported by	JSR 286 Standard
com.oracle.portlet.portalInfo Provider	Application Portlet	No
com.oracle.portlet.redirectAft erAction	Application Portlet	No
com.oracle.portlet.renderHea ders	Application Portlet	No
com.oracle.portlet.requireIFr ame	Application Portlet	No
com.oracle.portlet.streaming Optimized	Application Portlet	No
com.oracle.portlet.suppressW srpOptimisticRender	Application Portlet	No
com.oracle.portlet.trapWsrpR enderExceptions	Application Portlet	No
com.oracle.portlet.trimEncod eUrls	Application Portlet	No
com.oracle.portlet.useWsrpU serContextForUserAuthentic ation	Application	No
com.oracle.portlet.wsrpHead erMode	Application Portlet	No
com.oracle.portlet.wsrpLegac yPortletHandle	Portlet	No
com.oracle.portlet.wsrpPortle tHandle	Portlet	No
oracle.portlet.bridge.adf.raise UndeclaredContextualEvents	Portlet	No

13.3.6.1.1 javax.portlet.actionScopedRequestAttributes

Specifies whether to store action-scoped request attributes so that they are available to portlets until a new action occurs.

You can use the WebCenter Portal-specific `excludedActionScopeRequestAttributes` container runtime option to limit which request attributes are stored in the scopes.

Valid values:

- `true`—store request attributes starting at `processAction` until the next `processAction`. You can specify a second value of `numberOfCachedScopes`

and a third value indicating the number of scopes to be cached by the portlet container.

- `false`—do not store request attributes.

13.3.6.1.2 javax.portlet.escapeXml

Specifies whether to XML encode URLs returned from `actionUrl`, `renderUrl`, and `resourceUrl` JSR 286 tag library tags.

You can override this option on a per-tag basis using the `encodeXml` attribute.

Valid values:

- `true`—(default) the URLs returned by the `actionUrl`, `renderUrl`, and `resourceUrl` tags are XML encoded (using ampersand entities for parameter separation).

- `false`—the URLs returned by the actionUrl, renderUrl, and resourceUrl tabs are not XML encoded (and use just ampersand characters).

> **Note:**
>
> This option does not have any effect on the return value of `PortletURL.toString()` or `ResourceURL.toString()`. In both these cases, for JSR 286, the output uses only ampersand characters. To use ampersand entities or to be able to specify the XML encoding to use when generating URLs not using the tag libraries, see the `BaseURL.write()` methods.

13.3.6.1.3 javax.portlet.servletDefaultSessionScope

Specifies the scope of the session object provided to servlets or JSPs that are included or forwarded from a Java portlet.

Valid values:

- `APPLICATION_SCOPE`—(default) map the portlet session with application scope.

- `PORTLET_SCOPE`—map the portlet session with portlet scope.

13.3.6.1.4 com.oracle.portlet.allowEventPayloadsWithoutJAXBBinding

Allows event payload types declared in `portlet.xml` and event payloads sent from JSR 286 portlets to bypass the JSR 286 spec requirement that these types have a valid JAXB binding.

Valid values:

- `true`—event payloads without valid JAXB bindings are allowed.

- `false`—event payloads without valid JAXB bindings are not allowed, per the JSR 286 specification.

13.3.6.1.5 com.oracle.portlet.allowWsrpExport

Specifies whether the WSRP export-portlets operation should be supported for the web application.

This option is used mainly for backward-compatibility; it is preferred to control the export-portlets operation through the `wsrp-producer-config.xml` setting.

Valid values:

- `true`—(default) export-portlets is allowed.

- `false`—export-portlets is not allowed. This overrides the setting in `WEB-INF/wsrp-producer-config.xml`.

13.3.6.1.6 com.oracle.portlet.compatibilityMode

Set to `owc168` to invoke the WebCenter Portal JSR 286 container JSR 168 compatibility mode.

This also automatically sets the `disallowResourceServing` runtime option to `true` if no other value for that option is specified.

13.3.6.1.7 com.oracle.portlet.defaultProxiedResourceRequiresWsrpRewrite

Specifies the default WSRP `requiresRewrite` flag to use when encoding URLs for resources not served by the portlet. This setting is used for all URLs returned by the `PortletResponse.encodeURL()` method, unless overridden by the presence of the `oracle.portlet.server.resourceRequiresRewriting` request attribute when the `PortletResponse.encodeURL()` method is called.

Valid values:

- `true`—(default) the `requiresRewrite` flag is set to `true`, indicating that the resource should be rewritten by the consumer.

- `false`—the `requiresRewrite` flag is set to `false`, indicating that the resource does not necessarily need to be rewritten by the consumer.

13.3.6.1.8 com.oracle.portlet.defaultServedResourceRequiresWsrpRewrite

Specifies the default WSRP `requiresRewrite` flag to use when generating resource URLs for portlet-served resources.

This setting is used for all resource URLs created by the portlet, unless overridden by the presence of the `resourceRequiresRewriting` request attribute when the resource URL methods `write()` or `toString()` are called. This setting is also used to specify the WSRP `requiresRewriting` flag on the served resource response, but can be overridden by the presence of the `resourceRequiresRewriting` request attribute when the portlet's `serveResource()` method returns.

Valid values:

- `true`—the `requiresRewrite` URL flag and `requiresRewriting` response flag are set to `true`, indicating that the resource should be rewritten by the consumer

- `false`—the `requiresRewrite` URL flag and `requiresRewriting` response flag are set to `false`, indicating that the resource does not necessarily need to be written by the consumer, although the consumer may choose to rewrite the URL.

If unspecified, the `requiresRewrite` URL flag is not given a value, and the `requiresRewriting` response flag for a `serveResource` operation is based on the MIME type of the response.

13.3.6.1.9 com.oracle.portlet.disallowResourceServing

Specifies whether portlets are allowed to serve resources. This is useful if JSR 168 portlets are run in the 286 container, as any JSR 168 portlet extending `javax.portlet.GenericPortlet` automatically inherits the JSR 286 functionality, which automatically forwards resource requests to a file in the web application named after the resource ID, creating a potential security problem. For the same security reason, JSR 286 portlets that do not serve resources are safest to disallow resource serving.

Valid values:

- `true`—portlets are not allowed to serve resources.

- `false`—portlets are allowed to serve resources.

13.3.6.1.10 com.oracle.portlet.escapeXmlEncodeUrls

Specifies whether the JSR 286 container should XML-encode URLs generated from the `PortletResponse.encodeURL()` method.

Valid values:

- `true`—URLs generated by the `PortletResponse.encodeURL()` method are XML-encoded.

- `false`—(default) URLs generated by `PortletResponse.encodeURL()` are not XML-encoded.

13.3.6.1.11 com.oracle.portlet.eventPayloadsXmlType

Specifies optional XML schema types for JAXB-bindable Java object event payloads if events are sent over WSRP. This is a multi-valued runtime option; each value consists of a Java class name and QName pair, separated by a colon (`:`). The QName should use the standard Java String representation of a QName (`{namespace}localpart`). For each value specified, the QName is used as the XML schema type to annotate WSRP event payloads of the specified Java object type after the object has been marshalled to XML. This may be useful when using events to communicate across portlets on multiple producers.

13.3.6.1.12 com.oracle.portlet.excludedActionScopeRequestAttributes

This is a multi-valued property with each value being a regular expression. Request attributes which match any of the regular expressions are not stored as action-scoped request attributes if the `javax.portlet.actionScopedRequestAttributes` container runtime option is used, in addition to any request parameters whose values match the regular expressions defined in the `com.oracle.portlet.externalScopeRequestAttributes` container runtime option.

13.3.6.1.13 com.oracle.portlet.externalScopeRequestAttributes

This is a multi-valued property with each value being a regular expression. Request attributes which match any of the regular expressions are considered outside of portlet scope, and are shared with the underlying portal request.

If the `javax.portlet.actionScopedRequestAttributes` option is used, any request attributes matching the regular expressions declared in `externalScopeRequestAttributes` are not stored in the action scope request attributes.

13.3.6.1.14 com.oracle.portlet.importCssToIFrame

Specifies to a portal consumer whether the CSS file should be imported to an IFRAME portlet.

Valid values:

- `false`—(default) nothing is done.

- `true`—the CSS file from the consumer is applied to an IFRAME portlet.

13.3.6.1.15 com.oracle.portlet.minimumWsrpVersion

Specifies minimum required WSRP version for the portlet to work. If the WSRP version being used is less than the value specified, the portlet will not be included in a WSRP `GetServiceDescription`, `GetPortletDescription`, `GetMarkup` and other WSRP responses.

Valid values:

- `1`

- `2`

13.3.6.1.16 com.oracle.portlet.offerPortletOverWsrp

Specifies whether the portlet is offered in the WSRP producer's service description.

Any `offerRemote` setting in a `.portlet` file referencing the JSR 168/286 portlet overrides this container runtime option.

Valid values:

- `true`—(default) the portlet is offered in the WSRP producer's service description.

- `false`—the portlet is not included in the service description.

If not specified at all, the default value specified in the `WEB-INF/producer-config.xml` is used.

13.3.6.1.17 com.oracle.portlet.portalInfoProvider

Specifies the class name of an optional implementation of `com.bea.portlet.container.IPortalInfo` interface. This implementation defines the value returned by the JSR 286 container's `request.getPortalContext().getPortalInfo()`. The default implementation returns the producer/local server name and version, but a custom implementation could pass both the consumer server information and the producer server information in a WSRP scenario.

13.3.6.1.18 com.oracle.portlet.redirectAfterAction

Causes the JSR 286 container to send a redirect to the browser to the portlet's render URL after a `processAction` is run (and after events are handled) so that reloading the resulting page will not result in another `processAction`.

Valid values:

- `true`—a redirect is issued after every portlet action.

- `false`—no redirect is not automatically issued after portlet actions.

13.3.6.1.19 com.oracle.portlet.requireIFrame

Specifies whether the portlet must be rendered inside an IFRAME.

Valid values:

- `true`—renders the portlet inside an IFRAME.

- `false`—does not force the portlet to be rendered inside an IFRAME.

13.3.6.1.20 com.oracle.portlet.streamingOptimized

Indicates the portlet is optimized to run in streaming mode, which may enhance performance.

Valid values:

- `true`—the portlet is optimized to run in streaming mode.

- `false`—the portlet is not optimized to run in streaming mode.

13.3.6.1.21 com.oracle.portlet.suppressWsrpOptimisticRender

Suppresses the optimistic render of a portlet after the action and/or event lifecycles if the portlet is being run over WSRP.

Valid values:

- `true`—optimistic render is always suppressed.

- `false`—optimistic render may be performed.

13.3.6.1.22 com.oracle.portlet.trapWsrpRenderExceptions

Specifies whether the JSR 286 container should send exceptions during render as WSRP SOAP faults, or render an exception message for the portlet markup instead.

Valid values:

- `true`—exceptions are not sent as SOAP faults but instead as rendered exception stack traces.

- `false`—(default) exceptions generated by the portlet during render are treated as SOAP faults.

13.3.6.1.23 com.oracle.portlet.trimEncodeUrls

Specifies whether the JSR 286 container should trim whitespace from URLs passed to the `PortletResponse.encodeURL()` method.

Valid values:

- `true`—(default) leading and trailing whitespace is trimmed from the URL passed into `PortletResponse.encodeURL()`.

- `false`—whitespace is not trimmed from the URL passed into `PortletResponse.encodeURL()`.

13.3.6.1.24 com.oracle.portlet.useWsrpUserContextForUserAuthentication

Specifies whether the PortletRequest methods `getRemoteUser()`, `getUserPrincipal()` and `isUserInRole()` are based on the WSRP user context information or on standard J2EE security.

Valid values:

- `true`—the user information is based on the WSRP user context, if the portlet is run over WSRP. This can be a security problem so use this option with care.

- `false`—(or the portlet is not being run over WSRP) the user information is based on the J2EE authenticated user.

13.3.6.1.25 com.oracle.portlet.wsrpHeaderMode

Used only when portlets are being rendered as WSRP remote portlets, to indicate where cookies and headers should be put in the WSRP SOAP response as a hint to the WSRP consumer for the header or cookie's intended final destination.

When portlets are run locally (not over WSRP), headers and cookies set by portlets are always assumed to go to the client. Setting this container runtime option sets a default value for the `PortletRequest` attribute `com.oracle.portlet.wsrpHeaderMode`, which can still be overridden by the portlet at runtime on a per-header basis.

Valid values:

- `client`—headers and cookies set by the portlet are directed to go to the client (for example, the browser).

- `consumer`—headers and cookies set by the portlet are directed to go to the consumer and not passed on to the client.

- `both`—headers and cookies set by the portlet are directed to go to the client and the consumer.

If not set, the portlet container assumes the default value for the producer as specified in the `WEB-INF/wsrp-producer-config.xml` file.

13.3.6.1.26 com.oracle.portlet.wsrpLegacyPortletHandle

Allows the specification of a legacy WSRP portlet handle to be used for the portlet, which must be unique within the web application. This is useful for backward-compatibility with WebCenter Portal consumers that use the legacy, portlet-position-based WSRP portlet handles.

If specified, the legacy portlet handle is published in the WSRP serviceDescription as a legacy-handle extension on the portlet, and consumers are able to access the portlet using the legacy portlet handle, although the portlet will not be published in the WSRP serviceDescription under the legacy portlet handle.

13.3.6.1.27 com.oracle.portlet.wsrpPortletHandle

Allows the specification of the WSRP portlet handle to be used for the portlet, which must be unique within the web application.

13.3.6.1.28 oracle.portlet.bridge.adf.raiseUndeclaredContextualEvents

Allows the Oracle JSF Portlet Bridge to raise ADFm events that do not have corresponding portlet events declared in the `portlet.xml` file.

Valid values:

- `true`—any ADFm event raised is forwarded on as a portlet event.

- `false`—only events with corresponding portlet event declarations are forwarded.

13.3.6.2 Setting Container Runtime Options for All Portlets in an Application

Setting container runtime options at the application level affects all the portlets in the application.

To set application-level container runtime options:

1. Open the Overview Editor for the `portlet.xml` file.

   For more information, see How to Edit the Portlet Deployment Descriptor File.

2. Click the **Application** tab, if necessary.

3. Expand the Container Runtime Options section, if necessary.

4. Click the **Add Container Runtime Option** icon to display a list of available options.

   The list includes all container runtime options supported by the WebCenter Portal portlet container, which are listed in Table 13-1.

   If you are using a different portlet container that supports additional container runtime options that are not listed, select **Customize** and enter the name of the option in the **Name** field.

   If you specify a container runtime option that is not supported by the portlet container, it is ignored.

5. In the **Value** field, enter a value for the container runtime option.

   The container runtime option is added to the `portlet.xml` code, as shown below.

   ```
   <container-runtime-option>
    <name>com.oracle.portlet.requireIFrame</name>
    <value>true</value>
   </containter-runtime-option>
   ```

6. Save the `portlet.xml` file.

13.3.6.3 Setting Container Runtime Options for Individual Portlets

You can set container runtime options at the individual portlet level to override the application-wide settings.

To set portlet-level container runtime options:

1. Open the Overview Editor for the `portlet.xml` file.

   For more information, see How to Edit the Portlet Deployment Descriptor File.

2. Click the **Portlets** tab, if necessary.

3. Click the **Advanced** tab.

4. Click the **Add Container Runtime Option** icon to display a list of available options.

   The list includes all container runtime options supported by the WebCenter Portal portlet container, which are listed in Table 13-1.

If you are using a different portlet container that supports additional container runtime options that are not listed, select **Customize** and enter the name of the option in the **Name** field.

If you specify a container runtime option that is not supported by the portlet container, it is ignored.

5. In the **Value** field, enter a value for the container runtime option.

The container runtime option is added to the `portlet.xml` code, as shown below.

```
<container-runtime-option>
 <name>com.oracle.portlet.requireIFrame</name>
 <value>false</value>
</containter-runtime-option>
```

6. Save the `portlet.xml` file.

13.3.7 How to Use Public Render Parameters in JSR 286 Portlets

This section includes the following topics:

- About Public Render Parameters

- Declaring a Public Render Parameter at the Application Level

- Defining a Public Render Parameter for a Portlet

- Using Public Render Parameters: An Example

13.3.7.1 About Public Render Parameters

Public render parameters enable portlets to share parameter values, allowing a form of interportlet communication.

For example, if a Map portlet and a Weather portlet are both configured to use a Zipcode public render parameter, entering a zip code in the Map portlet updates the same parameter value in the Weather portlet.

Any public render parameters supported by a portlet must be declared in the application section of the portlet deployment descriptor file (`portlet.xml`). Public render parameters defined at the application level in this way are available to all the portlets in the application.

Individual portlets within the application can then specify which of these public render parameters they want to use.

If you declare and define public render parameters as described below, interportlet communication happens automatically, without any further coding required on your part.

For more information about public render parameters and how to implement them, see the JSR 286 specification at:

http://jcp.org/en/jsr/detail?id=286

13.3.7.2 Declaring a Public Render Parameter at the Application Level

For a public render parameter to be available to a portlet, it must first be declared in the application section of the portlet deployment descriptor file (`portlet.xml`).

To define a public render parameter at the application level:

1. Open the Overview Editor for the `portlet.xml` file.

For more information, see How to Edit the Portlet Deployment Descriptor File.

2. Click the **Parameters** tab.

3. Click the **Add** icon to create a new public render parameter.

4. From the drop-down list, select whether you are specifying a fully qualified name (QName) for the parameter, or just the local part of the name.

- **Qualified Name**—A *QName* uniquely identifies the public render parameter across applications by specifying a namespace for the parameter as well as a local name. Within the portlet code, a public render parameter is accessed by its identifier, which identifies the parameter uniquely within the application. However, a page typically contains multiple portlets which may come from different applications. Using QNames ensures that public render parameters from one portlet do not unintentionally interfere with the other portlets on a page regardless of where those portlets come from.

- **Unqualified Name**—If the namespace for the parameter is the same as the application default namespace, you can omit the namespace when defining the parameter by specifying an unqualified name. If the application default namespace has not been defined, a parameter with an unqualified name uses the XML default namespace.

  Tip:

  To check whether an application default namespace has been defined, click the **Application** tab and see if a value has been entered in the **Default Namespace** field.

5. Enter a name for the parameter.

- If you selected **Qualified Name**—in the **Namespace** field, enter the namespace for the parameter and in the **Name** field, enter the local part of the parameter name.

- If you selected **Unqualified Name**—in the **Name** field, enter the local part of the parameter name. The parameter uses the application default namespace as the namespace for the parameter (or the XML default namespace if no default namespace is defined for the application).

6. In the **Identifier** field, enter a name to identify the public render parameter within the application.

The identifier must be unique within the application and is used to identify the parameters used by a portlet and in the portlet code to access the parameter. Using the identifier means that portlet developers do not need to be aware of the parameter's fully qualified name; they simply need to know the simpler application-specific identifier.

7. In the **Description** field, enter a description for the public render parameter. The description should provide enough information so that portlet developers can determine how to use this parameter in their portlets.

8. (Optional) In the Aliases panel, you can provide a list of aliases so that the parameter can accept values from other public render parameters even if they have different QNames.

    a. Click the **Create** icon to create a new alias for the parameter.

    b. In the **Namespace** field, enter the namespace of the parameter to use for the alias.

    c. In the **Name** field, enter the local part of the parameter to use for the alias.

    Tip:

    When creating aliases for public render parameters, it is a good idea to set up reciprocal aliases. So if a parameter in portlet A has an alias to a parameter in portlet B, you should also, if possible, create an alias for the parameter in portlet B to the parameter in portlet A.

9. The public render parameter is added to the application definition section of the `portlet.xml` file, as shown below.

```
<public-render-parameter>
 <description>Parameter for zip code</description>
 <identifier>Zip</identifier>
 <qname xmlns:x="http://example.com/params">x:Zip</qname>
 <alias xmlns:x="http://example.com/params">x:ZipCode</alias>
</public-render-parameter>
```

10. Save the `portlet.xml` file.

13.3.7.3 Defining a Public Render Parameter for a Portlet

If you want a portlet to support a public render parameter, add it to the portlet.

To add a public render parameter to a portlet:

1. Open the Overview Editor for the `portlet.xml` file.

For more information, see How to Edit the Portlet Deployment Descriptor File.

2. If the public render parameter has not yet been defined within the Portlet Producer application, you must do this first.

For more information, see Declaring a Public Render Parameter at the Application Level.

3. Click the **Portlets** tab and select the portlet in which you want to use the public render parameter.

4. Click the **Parameters** tab.

In the Publishing Events panel, the **Available** list shows all the portlet events that have been defined for the application.

5. Click the **Parameters** tab.

In the Public Render Parameters panel, the **Available** list shows all the public render parameters that have been defined for the application.

6. Select the public render parameter that you want to use in the portlet and click the **Add** icon to move it to the **Selected** list.

This adds the public render parameter to the portlet definition in `portlet.xml`, as shown below.

```
<supported-public-render-parameter>Zip</supported-public-render-parameter>
```

7. Save the `portlet.xml` file.

8. You can now use this public render parameter in the code for your portlet.

13.3.7.4 Using Public Render Parameters: An Example

The following example shows how to use public parameters to link the behavior of two portlets using parameters with different names. In the example, we take a generic Parameter Form portlet that allows us to update public render parameters, and show how that can be linked to a Stock Price portlet that renders stock prices, taking the stock symbol from a public render parameter. The generic name of the parameter from the Parameter Form portlet (`parameter1`) does not match the name of the parameter in the Stock Price portlet (`stocksymbol`), so the developer of the Stock Price portlet uses an alias to provide a link between the two parameters.

The example below shows an excerpt from the `portlet.xml` file for the portlet producer application that contains the Parameter Form portlet. The portlet producer application defines a public render parameter (`parameter1`) that is supported by the Parameter Form portlet.

```
<portlet-app ...>
 <portlet id="1234567890">
 ...
 <supported-public-render-parameter>
 parameter1
 </supported-public-render-parameter>
 ...
 </portlet>
 ...
 <public-render-parameter>
 <description xml:lang="en">First parameter set by portlet</description>
 <identifier>parameter1</identifier>
 <qname xmlns:op="http://xmlns.oracle.com/portlet/oracle-portlet-app">
 op:parameter1
 </qname>
 </public-render-parameter>
 ...
</portlet-app>
```

The code for the Parameter Form portlet sets the value of the `parameter1` portlet when a value is entered in the portlet:

```
public void processAction(ActionRequest request,
 ActionResponse actionResponse)
 throws PortletException, IOException
{
 //
//Read the new parameter value posted in the portlet form
 //
String param1 =actionResponse.getParameter("form_param1");

 //
//Set the new parameter values. These will be intepreted by the
 //container as public render parameters as the names match the names
 //of the declared parameters.
 //
actionResponse.setRenderParameter("parameter1", param1);
}
```

The following example shows an excerpt from the `portlet.xml` file for the portlet producer application that contains the Stock Price portlet. This portlet producer

application also defines a public render parameter (`stocksymbol`) that is supported by the Stock Price portlet. For the `stocksymbol` parameter to link to the Parameter Form portlet's parameter, an alias for `parameter1` is added.

```
<portlet-app ...>
 <portlet>
 ...
 <supported-public-render-parameter>
 stocksymbol
 </supported-public-render-parameter>
 ...
 </portlet>
 ...
 <public-render-parameter>
 <description xml:lang="en">The stock symbol</description>
 <identifier>stocksymbol</identifier>
 <qname xmlns:s="http://www.oracle.com/stocks">s:stocksymbol</qname>
 <!-- Alias matches the Parameter Form portlet's first parameter name -->
 <alias xmlns:op="http://xmlns.oracle.com/portlet/oracle-portlet-app">
 op:parameter1
 </alias>
 </public-render-parameter>
 ...
</portlet-app>
```

The code for the Stock Price portlet reads the value of the public render parameter posted by the Parameter Form portlet:

```
public void doView(RenderRequest request, RenderResponse response)
 throws PortletException, IOException
{
 response.setContentType("text/html; charset=UTF-8");
 PrintWriter out =response.getWriter();

//Retrieve any values for the public render parameter.
 //The parameter is looked up in the request using its identifier in portlet.xml
 //(not it's name or alias).
 String symbol =request.getParameter("stocksymbol");
 ...
}
```

When these two portlets are dropped onto the same page, they are automatically linked together. Entering a value in the first parameter field of the Parameter Form portlet causes the Stock Price portlet to be updated with the new value.

13.3.8 How to Use Portlet Events in JSR 286 Portlets

This section includes the following topics:

- About Portlet Events

- Declaring a Portlet Event at the Application Level

- Defining a Processing Event for a Portlet

- Defining a Publishing Event for a Portlet

- Using Portlet Events: An Example

13.3.8.1 About Portlet Events

Portlet events are a JSR 286 feature that enables interportlet communication by providing portlets with the ability to respond to actions that occur outside of the portlet itself, for example an action performed on the page that contains the portlet or on another portlet on the same page. Portlet events can be cascaded so that a portlet may respond to an event by triggering an event of its own, which in turn affects other portlets on the page.

Any portlet events supported by a portlet must be declared in the application section of the portlet deployment descriptor (`portlet.xml`). Portlet events defined at the application level in this way are available to all the portlets in the application.

Individual portlets within the application can then specify which of these portlet events they want to use. Portlets can declare events that they are interested in receiving, called *processing events*, and events that they trigger, called *publishing events*.

If you declare and define portlet events as described below, interportlet communication happens automatically, without any further coding required on your part.

For more information about portlet events and how to implement them, see the JSR 286 specification at:

http://jcp.org/en/jsr/detail?id=286

13.3.8.2 Declaring a Portlet Event at the Application Level

For a portlet event to be available to a portlet, it must first be declared in the application section of the portlet deployment descriptor file (`portlet.xml`).

To define a portlet event at the application level:

1. Open the Overview Editor for the `portlet.xml` file.

    For more information, see How to Edit the Portlet Deployment Descriptor File.

2. Click the **Events** tab.

3. Click the **Add** icon to create a new portlet event.

4. From the drop-down list, select whether you are specifying a fully qualified name (QName) for the portlet event, or just the local part of the name.

    *   **Qualified Name**—A *QName* uniquely identifies the portlet event across applications by specifying a namespace for the event as well as a local name. A page typically contains multiple portlets which may come from different applications. Using QNames ensures that portlet events from one portlet do not unintentionally interfere with the other portlets on a page regardless of where those portlets come from.

    *   **Unqualified Name**—If the namespace for the portlet event is the same as the application default namespace, you can omit the namespace when defining the event by specifying an unqualified name. If the application default namespace has not been defined, a portlet event with an unqualified name uses the XML default namespace.

Tip:

To check whether an application default namespace has been defined, click the **Application** tab and see if a value has been entered in the **Default Namespace** field.

5. Enter a name for the portlet event.

   - If you selected **Qualified Name**—In the **Namespace** field, enter the namespace for the portlet event and in the **Name** field, enter the local part of the event name.

   - If you selected **Unqualified Name**—In the **Name** field, enter the local part of the portlet event name. The event uses the application default namespace as the namespace for the event (or the XML default namespace if no default namespace is defined for the application).

6. In the **Payload Type** field, enter or browse for the data type of the payload provided by the portlet event.

7. In the **Description** field, enter a description for the portlet event.

8. (Optional) In the Aliases panel, you can provide a list of aliases so that a portlet event can be recognized by the portlet even if it has a different QName from the one defined by the portlet.

   a. Click the **Create** icon to create a new alias for the portlet event.

   b. In the **Namespace** field, enter the namespace of the portlet event to use for the alias.

   c. In the **Name** field, enter the local part of the portlet event to use for the alias.

9. The portlet event is added to the application definition section of the `portlet.xml` file, as shown below.

```
<event-definition id="latLong">
 <qname xmlns:x="http://xmlns.oracle.com/portlet/EventSample">
 x:latLong
 </qname>
 <value-type>portlet.LatLong</value-type>
</event-definition>
```

10. Save the `portlet.xml` file.

13.3.8.3 Defining a Processing Event for a Portlet

If you want a portlet to listen for a particular portlet event, define it as a processing event.

To add a processing event to a portlet:

1. Open the Overview Editor for the `portlet.xml` file.

   For more information, see How to Edit the Portlet Deployment Descriptor File.

2. If the portlet event has not yet been defined within the Portlet Producer application, you must do this first.

   For more information, see Declaring a Portlet Event at the Application Level.

3. Click the **Portlets** tab and select the portlet in which you want to use the portlet event.

4. Click the **Events** tab.

In the Processing Events panel, the **Available** list shows all the portlet events that have been defined for the application.

5. Select the portlet event that you want to use in the portlet and click the **Add** icon to move it to the **Selected** list.

This adds the portlet event as a processing event to the portlet definition in `portlet.xml`, as shown below.

```
<supported-processing-event id="latLong">
 <qname xmlns:x="http://xmlns.oracle.com/portlet/EventSample">
 x:latLong
 </qname>
</supported-processing-event>
```

6. Save the `portlet.xml` file.

You can now use this portlet event in the code for your portlet.

13.3.8.4 Defining a Publishing Event for a Portlet

If you want a portlet to trigger a particular portlet event, define it as a publishing event.

To add a publishing event to a portlet:

1. Open the Overview Editor for the `portlet.xml` file.

For more information, see How to Edit the Portlet Deployment Descriptor File.

2. If the portlet event has not yet been defined within the Portlet Producer application, you must do this first.

For more information, see Defining a Publishing Event for a Portlet.

3. Click the **Portlets** tab and select the portlet in which you want to use the portlet event.

4. Click the **Events** tab.

In the Publishing Events panel, the **Available** list shows all the portlet events that have been defined for the application.

5. Select the portlet event that you want to use in the portlet and click the **Add** icon to move it to the **Selected** list.

This adds the portlet event as a publishing event to the portlet definition in `portlet.xml`, as shown below.

```
<supported-publishing-event>
 <qname xmlns:x="http://xmlns.oracle.com/portlet/EventSample">
 x:latLong
 <qname>
</supported-publishing-event>
```

6. Save the `portlet.xml` file.

You can now use this portlet event in the code for your portlet.

13.3.8.5 Using Portlet Events: An Example

The following example shows how to use portlet events to use the actions in one portlet to affect another portlet on the same page. In the example, we have a portlet, the Department Locations portlet, that lists the geographical locations of a company's various offices. The example shows how that portlet can be linked to another portlet, the Map portlet, which displays a Google map of the location selected in the Department Locations portlet.

The example below shows an excerpt from the `portlet.xml` file for the portlet producer application that contains the two portlets. The portlet producer application defines a portlet event (`latLong`) that is supported by the two portlets. The payload of the event is an object that encapsulates the latitude and longitude of a location.

```
<event-definition id="latLong">
 <qname xmlns:x="http://xmlns.oracle.com/portlet/EventSample">x:latLong</qname>
 <value-type>portlet.LatLong</value-type>
</event-definition>
```

The next example shows that the Department Locations portlet uses the `latLong` event as a publishing event. That is, it raises the event under particular circumstances.

```
<portlet id="locations">
 ...
 <supported-publishing-event>
 <qname xmlns:x="http://xmlns.oracle.com/portlet/EventSample">x:latLong<qname>
 </supported-publishing-event>
 ...
</portlet>
```

The Department Locations portlet raises the `latLong` event in its `processAction` method if a location is selected in the portlet (see example below). It also sets the payload of the event to the latitude and longitude of the selected location.

```
public void processAction(ActionRequest request, ActionResponse response)
 throws PortletException
 {
String locationId =request.getParameter("locationId");
 Location location =getLocation(Integer.parseInt(locationId));

 if (location != null)
 {
//QName matches the event declared as a supported-publishing-event in
 //portlet.xml for this portlet.

 //LatLong event used by the Map portlet.
 response.setEvent(new QName("http://xmlns.oracle.com/portlet/EventSample",
"latLong"),
 new LatLong(location.getLatitude(), location.getLongitude()));

 }
}
```

The example below shows another portlet, the Map portlet, that also uses the `latLong` event, but this time as a processing event. That is, it listens out for the event and takes a particular action if the event is raised elsewhere on the page.

```
<portlet id="map">
 ...
```

```
<supported-processing-event id="latLong">
<qname xmlns:x="http://xmlns.oracle.com/portlet/EventSample">x:latLong</qname>
</supported-processing-event>
...
</portlet>
```

The `processEvent` method for the Map portlet receives the `latLong` event and uses the payload from that event to set the `LAT_LONG_PARAMETER` render parameter, which the portlet uses during rendering to determine the location to display in the Google map:

```
@Override
public void processEvent(EventRequest request,
EventResponse response)
throws PortletException, IOException
{
response.setRenderParameters(request);

Event event =request.getEvent();
if (event.getName().equals("latLong"))
{
response.setRenderParameter(LAT_LONG_PARAMETER, event.getValue().toString());
}
}
```

Tip:

In this case, the events used by the Department Locations and Map portlets have the same name (`latLong`). If the Map portlet listens for an event using a different name (for example, `geolocation`), the two portlets could still be automatically linked by defining an alias for the `geolocation` event in the `portlet.xml` file as follows:

```
<event-definition id="geolocation"> <qname xmlns:x="http://xmlns.oracle.com/
portlet/Map">
x:geolocation
</qname> <alias xmlns:x="http://xmlns.oracle.com/portlet/EventSample">
x:latLong
</alias>
<value-type>portlet.LatLong</value-type>
</event-definition>
```

Note, however, that the payload type of the two events must be the same for automatic linking to work.

> **Note:**
>
> For an event payload to be passed between portlets it must be possible to form an XML payload from event payload. For this to be possible one of the following must be true:
>
> - The payload must have a JAXB binding, for example by adding the `XmlRootElement` annotation to the portlet class:
>
>   ```
>   package portlet;
>
>   import java.io.Serializable;
>   import javax.xml.bind.annotation.XmlRootElement;
>
>   @XmlRootElement(namespace="http://xmlns.oracle.com/portlet/EventSample")
>   public class LatLong implements Serializable
>   {
>    private final double _latitude;
>    private final double _longitude;
>   ...
>   ```
>
> - The `allowEventPayloadsWithoutJAXBBindings` container runtime option must be set to `true` in `portlet.xml`.
>
>   For more information, see How to Customize the Runtime Environment for JSR 286 Portlets.

13.3.9 How to Add Portlet Preferences to JSR 286 Portlets

This section includes the following topics:

- About Portlet Preferences
- Adding a Portlet Preference to a JSR 286 Portlet
- Adding Simple Portlet Personalization: An Example

13.3.9.1 About Portlet Preferences

Portlet preferences enable end users to personalize or customize portlets at runtime. Personalizations are visible only to the user that performs them; not to other users. Customizations are visible to all users who have not personalized the portlet.

By default, when you create a JSR 286 portlet using the JDeveloper wizard, a portlet preference is created to enable users to change the title of the portlet at runtime. You can create additional portlet preferences, either during portlet creation or by editing the `portlet.xml` file after portlet creation, to enable end users to make other modifications to the portlet at runtime.

For information about how to add a portlet preference to the `portlet.xml` file after portlet creation, see Adding a Portlet Preference to a JSR 286 Portlet. For information about how to add a portlet preference during portlet creation, see Creating a JSR 286 Java Portlet.

13.3.9.2 Adding a Portlet Preference to a JSR 286 Portlet

To enable users to modify portlet behavior or content at runtime, you must create portlet preferences for those attributes of the portlet that you want users to be able to modify.

To add a portlet preference to a portlet:

1. Open the Overview Editor for the `portlet.xml` file.

    For more information, see How to Edit the Portlet Deployment Descriptor File.

2. Click the **Portlets** tab and select the portlet for which you want to add the portlet preference.

3. Click the **Advanced** tab.

    The Preferences panel lists any existing portlet preferences that exist for the portlet, for example, the `portletTitle` preference.

4. Click the **Add** icon to create a new portlet preference.

5. In the **Name** field, enter a name for the preference.

    The name must be unique within the portlet.

6. In the **Value** field, enter one or more default values for the preference.

    Separate multiple entries with commas.

7. Select **Read-only** if you do not want users to be able to update the value of the preference.

    The portlet preference is added to the application definition section of the `portlet.xml` file, as shown below.

    ```
    <portlet-preference>
     <name>portletTitle</name>
    </portlet-preference>
    <portlet-preference>
     <name>myPreference</name>
     <value>7</value>
    </portlet-preference>
    ```

8. Save the `portlet.xml` file.

    You can now use this preference in your portlet code using the `getPreference` method.

    > **Tip:**
    >
    > If you want the preference to be translatable, you must add the appropriate key-value pairs to the resource bundle class.

13.3.9.3 Adding Simple Portlet Personalization: An Example

The following example provides a simple illustration of how you can enable personalization in portlets.

To add simple personalization to a portlet:

1. In JDeveloper, create a Portlet Producer application, accepting the default values.

2. Create a JSR 286 portlet, accepting the default values.

3. In the Application Navigator, expand the **Portlets** project.

4. Right-click `portlet.xml` and choose **Open**.

5. Click the **Portlets** tab and select **portlet1**.

6. Click the **Advanced** tab.

7. Click the **Add** icon to create a new portlet preference.

8. In the **Name** field, enter `portletContent`.

9. In the Application Navigator, right-click `view.jsp` and choose **Open.**

10. In the visual editor, click the **Source** tab and add the code indicated in bold in the example below to display the `portletContent` portlet preference:

```
<%@ 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]%><%}%>
```

11. 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.

12. In the visual editor, click the **Source** tab and add the code indicated in bold in the example below to implement a form field for users to enter a value for the `portletContent` customization preference:

```
<%@ 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>
```

13. In the Application Navigator, right-click `Portlet1.java` and choose **Open.**

14. In the visual editor, click the **Source** tab and add the code in bold in the example below 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();
```

When this portlet is available on a page, users can personalize it. The Edit mode of the portlet provides a **Portlet Content** field, where users can enter text. The text entered is then displayed as the content of the portlet.

13.3.10 How to Use Portlet Filters in JSR 286 Portlets

This section includes the following topics:

- About Portlet Filters

- Adding a Portlet Filter to an Application

- Applying a Portlet Filter to a Portlet

13.3.10.1 About Portlet Filters

Portlet filters are a JSR 286 feature that enables you to alter the content of a portlet at runtime. A *portlet filter* is a reusable Java component that can transform the content of portlet requests and portlet responses. Filters do not generally create a response or respond to a request as portlets do, rather they modify or adapt the requests and responses.

For more information about portlet filters and how to implement them, see the JSR 286 specification at:

http://jcp.org/en/jsr/detail?id=286

13.3.10.2 Adding a Portlet Filter to an Application

For a portlet to be able to use a portlet filter, the filter must first be defined within the application.

To add a portlet filter to an application:

1. Open the Overview Editor for the `portlet.xml` file.

   For more information, see How to Edit the Portlet Deployment Descriptor File.

2. Click the **Filters** tab.

3. Click the **Add** icon to create a new portlet filter.

4. In the Create Portlet Filter dialog you can either create a new portlet filter, or select a previously created portlet filter:

   - To create a new portlet filter:

     a. Select **New Portlet Filter**.

     b. In the **Name** field, enter a name for the portlet filter.

        The name must be unique within the application and use only letters, numbers, and the underscore character.

     c. In the **Package** field, enter or browse for the package to contain the new filter class.

     d.   Select one or more of the **Lifecycle Phase** check boxes to specify which of the `javax.portlet.filter` interfaces the filter class implements:

     Action—The filter class implements the `ActionFilter` interface

     Event—The filter class implements the `EventFilter` interface.

     Render—The filter class implements the `RenderFilter` interface.

     Resource—The filter class implements the `ResourceFilter` interface.

Tip:

When you create a new filter class using the Create Portlet Filter dialog, you can specify which of the filter interfaces the filter class implements. After the filter class has been created, you cannot add or remove filter interfaces through the Overview Editor. Instead, you must edit the source of the filter class directly to manually add or remove the interfaces and the `doFilter` methods defined for those interfaces.

- To select an existing portlet filter:

  a.   Select **Choose from Existing Class**.

  b.   Enter or browse for the filter class.

5.   Click **OK**.

6.   In the **Display Name** field, enter a more user-friendly name for the portlet filter.

7.   In the **Description** field, enter a description for the portlet filter.

8.   In the Initialization Parameters panel, you can specify initialization parameters that pass values to the `init()` method of the filter class.

   a.   Click the **Add** icon to specify an initialization parameter for the portlet filter.

   b.   In the **Name** field, enter the name of the initialization parameter.

   c.   In the **Value** field, enter the value to pass to the initialization parameter.

   d.   In the **Description** field, enter a description of the initialization parameter.

The portlet filter is added to the application definition section of the `portlet.xml` file, as shown below.

```
<filter>
 <display-name>Test Filter</display-name>
 <filter-name>filter_1</filter-name>
 <filter-class>javaportlets.MyFilter</filter-class>
 <lifecycle>ACTION_PHASE</lifecycle>
 <lifecycle>RENDER_PHASE</lifecycle>
</filter>
```

9.   Save the `portlet.xml` file.

13.3.10.3 Applying a Portlet Filter to a Portlet

After the portlet filter has been defined in the application, you can then apply it to one or more portlets within the application. You can also specify the order in which portlet filters are applied to portlets.

To apply a portlet filter to a portlet:

1. Open the Overview Editor for the `portlet.xml` file.

   For more information, see How to Edit the Portlet Deployment Descriptor File.

2. Click the **Filter Mappings** tab.

3. Click the **Add** icon to add a row to the filter mappings table.

4. From the **Filter Name** drop-down list, select the portlet filter that you want to apply to the portlet.

   The drop-down list is populated with all the portlet filters that are defined in the `portlet.xml` file.

5. From the **Portlet Name** drop-down list, select the portlet to which you want to apply the portlet filter.

   The drop-down list is populated with all the portlets that are defined in the `portlet.xml` file.

   Alternatively, you can use wildcards to apply the portlet filter to more than one portlet. For example, you can enter `*`to apply the portlet filter to all the portlets in the application.

   The filter mapping is added to the application definition section of the `portlet.xml` file. The example below shows the `filter_1` portlet filter applied to the portlet named `Portlet1`.

   ```
   <filter-mapping>
    <filter-name>filter_1</filter-name>
    <portlet-name>Portlet1</portlet-name>
   </filter-mapping>
   ```

6. Use the **Top**, **Up**, **Down**, and **Bottom** icons to order the portlet filters in the `portlet.xml` file. The order of the portlet filters determines the order in which the filters are applied to the portlets.

   > **Note:**
   >
   > Portlet filters can be mapped to multiple portlets. If you have multiple portlets mapped to (sharing) a filter, arbitrarily changing the filter order can produce undesired side effects. That is, if you change the order in which filters are applied in one portlet, the reordering will apply to all other portlets that share the filter.

7. Save the `portlet.xml` file.

13.3.11 How to Implement Interportlet Communication Across Different Pages

Generally, interportlet communication occurs between portlets that are displayed on the same page. However, it is possible to communicate between portlets on different pages, as shown in the following example.

To implement interportlet communication across different pages:

1. Create a custom data control and expose a method to capture the event that initiates the interportlet communication.

```
public void handleEvent(Object payload) {
    if(!(payload instanceof HashMap)){
            throw new IllegalArgumentException("Payload not a
HashMap<String,String>.");
    }
        String p1 = "",p2 = "",p3 = "";
        HashMap map = (HashMap) payload;
        String[] pa1 = (String[])map.get("parameter1");
        String[] pa2 = (String[])map.get("parameter2");
        String[] pa3 = (String[])map.get("parameter3");
        if(pa1 != null)
            p1 = pa1[0];
        if(pa2 != null)
            p2 = pa2[0];
        if(pa3 != null)
            p3 = pa3[0];
        //Only forward when all 3 parameters have values
        if(!p1.equals("") && !p2.equals("") && !p3.equals("")){
            HashMap<String,String> paramMap = new HashMap<String,String>();
            paramMap.put("parameter1", p1);
            paramMap.put("parameter2", p2);
            paramMap.put("parameter3", p3);
            Map<String, Object> pageFlowScope =
ADFContext.getCurrent().getPageFlowScope();
            pageFlowScope.put("paramMap", paramMap);
            //now the navigation
            FacesContext fctx = FacesContext.getCurrentInstance();
            Application application = fctx.getApplication();
            NavigationHandler navHandler = application.getNavigationHandler();
            navHandler.handleNavigation(fctx, null, "target");
        }
    }
```

2. Add this method to the page bindings and create an event from it.

```
<methodAction id="handleEvent" InstanceName="EventDataControl.dataProvider"
              DataControl="EventDataControl" RequiresUpdateModel="true"
              Action="invokeMethod" MethodName="handleEvent"
              IsViewObjectMethod="false">
    <NamedData NDName="payload" NDType="java.lang.Object"/>
    <events xmlns="http://xmlns.oracle.com/adfm/contextualEvent">
      <event name="handleEvent"/>
    </events>
</methodAction>
```

3. Wire the event to the portlet event. This ensures that the payload of the event triggered by the portlet is passed to the custom method.

```
<eventMap xmlns="http://xmlns.oracle.com/adfm/contextualEvent">
    <event name="ParameterFormPortlet1_1_Event">
      <producer region="*">
        <consumer region="" handler="handleEvent" handleCondition="">
          <parameters>
            <parameter name="payload" value="#{payLoad}"/>
          </parameters>
        </consumer>
      </producer>
    </event>
  </eventMap>
```

4. Create a `HashMap` to pass the payload to the `pageFlowScope` of the page that contains the second portlet.

```
HashMap<String,String> paramMap = new HashMap<String,String>();
paramMap.put("parameter1", p1);
paramMap.put("parameter2", p2);
paramMap.put("parameter3", p3);
Map<String, Object> pageFlowScope = ADFContext.getCurrent().getPageFlowScope();
pageFlowScope.put("paramMap", paramMap);
```

5. Define a navigation rule in the `faces-config.xml` file to navigate to the target page.

```
FacesContext fctx = FacesContext.getCurrentInstance();
Application application = fctx.getApplication();
NavigationHandler navHandler = application.getNavigationHandler();
navHandler.handleNavigation(fctx, null, "target");
```

6. Pass the parameters in the `pageFlowScope` to the target portlet using the `parameterMap` attribute.

```
<portlet id="ParameterDisplayPortlet1_1"
         portletInstance="/oracle/adf/portlet/PortletExample/ap/
Ei2default_354ce3c1_013d_1000_8002_c0a83801a490"
         class="oracle.adf.model.portlet.binding.PortletBinding"
         retainPortletHeader="false"
         listenForAutoDeliveredPortletEvents="true"
         listenForAutoDeliveredParameterChanges="true"
         xmlns="http://xmlns.oracle.com/portlet/bindings"
         parameterMap="#{pageFlowScope.paramMap}">
    <events>
      <event eventType="ParametersChange"
             name="ParameterDisplayPortlet1_1_Event"/>
    </events>
</portlet>
```

Tip:

You can also create a single generic `eventHandler` and add it to the page template. By adding a method binding in the page template, this is available on all pages. By using a wildcard for the publisher, you can also make sure that it listens to all events with a specific name.

For more information and to download a sample application, see the blog entry at:

http://www.ateam-oracle.com/inter-portlet-communication-between-pages/

13.3.12 How to Enhance JSR 286 Portlet Performance with Caching

This section includes the following topics:

- About Portlet Caching

- Implementing Expiry-Based Caching in JSR 286 Portlets

- Implementing Validation-Based Caching in JSR 286 Portlets

13.3.12.1 About Portlet Caching

When you have completed 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. JSR 286 portlets support expiry-based and validation-based caching.

For more information about caching, see Portlet Performance.

13.3.12.2 Implementing Expiry-Based Caching in JSR 286 Portlets

You can choose to implement expiry-based caching when you first create a portlet using the Create JSR 286 Portlet Wizard. However, during the initial development of a portlet, you may prefer to turn portlet caching off and implement it later in the development cycle when the portlet content becomes more stable. You may also want to edit the expiration period, or change the cache scope.

To implement expiry-based caching:

1. Open the Overview Editor for the `portlet.xml` file.

    For more information, see How to Edit the Portlet Deployment Descriptor File.

2. Click the **Portlets** tab and select the portlet for which you want to implement expiry-based caching.

3. Click the **Advanced** tab.

4. In the Cache Management panel, select **Cache Portlet**.

    > **Note:**
    >
    > To disable portlet caching, deselect **Cache Portlet**. This sets the cache expiration period to `0`, meaning that the cached content is always expired.

5. Select the **Cache Scope**:

    *   Select **Public** to share the cached content across users

    *   Select **Private** if the cached content should not be shared across users.

6. Specify the cache expiration period:

    *   Select **Cache Content Never Expires** to set the cache expiration period to -1. This means that the cached content never expires and the content is always retrieved from the cache. Use this option if you are confident that the portlet contains static content that is unlikely to change. Setting this option results in the following code being added to the `portlet.xml` file:

        ```
        <expiration-cache>-1</expiration-cache>
        ```

    *   Select **Cache Content Expires After** to expire the cached portlet content after a specified number of seconds. Enter the expiration period (n), in seconds, in the adjacent field. Setting this option results in the following code being added to the `portlet.xml` file:

        ```
        <expiration-cache>n</expiration-cache>
        ```

7. Save the `portlet.xml` file.

13.3.12.3 Implementing Validation-Based Caching in JSR 286 Portlets

Implementation of validation-based caching takes place after the initial portlet creation and requires hand coding.

The example below shows how a `GenericPortlet` would typically implement its `doView()` method such that the consumer caches the markup using validation-based caching. The example also shows how expiry-based caching can be defined programmatically (using the `CacheControl.setExpirationTime()` method) and used in conjunction with validation-based caching to further reduce the load on the producer. This would work equally well for `serveResource()`.

```
protected void doView (RenderRequest request, RenderResponse response)
 throws PortletException, IOException
{
CacheControl cacheControl =response.getCacheControl();
 String eTag =request.getETag();
 if (isMarkupStillValid(eTag))
{
//Wait 30 seconds before checking ETag again
 cacheControl.setExpirationTime(30);
//Tell consumer to use its cached content
 cacheControl.setUseCachedContent(true);
 return;
 }
//ETag not valid so set a new one ...
 //Define a new ETag
 cacheControl.setETag(generateETag(...));
 //Wait 60 seconds before checking ETag again
 cacheControl.setExpirationTime(60);
 //... and generate fresh portlet markup
 createMarkup(request, response);
}
private boolean isMarkupStillValid(String eTag)
{
 if (eTag ==null)
 {
return false; //No ETag was supplied
 }
//Perform portlet specific checks for the consumer's cached
//copy of the markup still being valid based on the given ETag
 ...
}
private String generateETag(...)
{
 //Portlet specific code to generate an ETag, for example, a hash
//of the data on which the portlet's markup is based
 ...
 return eTag;
}
```

For more information about validation-based caching in JSR 286 portlets, see the JSR 286 specification at:

http://jcp.org/en/jsr/detail?id=286

13.3.13 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 you 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 the code below to use resource proxying for a URL that you are encoding. Encapsulate this code within a method to avoid repeating it for every URL individually.

```
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`.

You can use the `defaultProxiedResourceRequiresWsrpRewrite` container runtime option to specify the default WSRP `requiresRewrite` flag to use. The option specified by this container runtime option (which is set to `true` by default) is used unless overridden by the request attribute. For more information, see "com.oracle.portlet.defaultProxiedResourceRequiresWsrpRewrite."

13.3.14 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 require 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 is required, set the `PortletRequest` attribute `oracle.portlet.server.useStatelessProxying` to `true`. For example, you might include code similar to the excerpt in the code below to use stateless proxying for a URL that you are encoding. Encapsulate this code within a method to avoid repeating it for every URL individually.

```
request.setAttribute("oracle.portlet.server.useStatelessProxying", Boolean.TRUE);
String url =response.encodeURL(pathToResource);
request.removeAttribute("oracle.portlet.server.useStatelessProxying");
```

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`.

13.3.15 How to Manage the Persistence Store for JSR 286 Portlets

This section includes the following topics:

- About the Persistence Store

- Setting Up a Persistence Store for a WSRP Producer

- Migrating a WSRP Producer Persistence Store

13.3.15.1 About the Persistence Store

The portlet persistence store is used for persisting consumer registration handles and portlet preference data. Portlet producers can use one of three types of persistence store:

- **Consumer**—Ties the producer metadata to the consumer application. This is the recommended method.

- **Database**—Persists data using a relational database. This is mainly provided for backward compatibility but may be useful if there is likely to be a large number of customizations.

- **File**—Persists data using the file system. This is provided for backward compatibility. You should not use a file based persistence store in your production environment as it does not support multi-tier or high availability environments. You may, however, want to use a file based persistence store while testing your application using Integrated WLS.

For more information, see Portlet Personalization and Customization.

13.3.15.2 JNDI Variables for WSRP Producer Persistence Store

Table 13-2 lists and describes the JNDI variables used to specify the persistence store for WSRP producers.

Table 13-2   WSRP Producer Database Preference Store-Related JNDI Variable

Variable Name	Variable Value	Description

`oracle/portal/wsrp/server/`
`persistentStore`
	`Database`
	`File`
	`Consumer`
		Determines which data store (File, Database, or Consumer) is used for persisting a portlet producer application's consumer registration handles and portlet preferences.

`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 `WC_ORACLE_HOME/` `portal` directory. Note that all producers running within the same WebLogic Server must use the same path for this variable. Otherwise, you get a `Portlet unavailable` error for some portlets.

13.3.15.3 Setting Up a Persistence Store for a WSRP Producer

WSRP portlet producers use a JNDI variable (`persistentStore`) to determine which type of persistence store to use. When you create a portlet for the first time in an

application, by default this variable is set to `Consumer`. You can change the value of this variable in the `web.xml` file of the portlet producer application.

> **Note:**
>
> If you create a portlet in an application that already contains other portlets, the existing persistence store setting is maintained.

If you use a File persistence store, you must also use the `fileStoreRoot` JNDI variable to specify the path to the root directory to be used by the persistence store. See JNDI Variables for WSRP Producer Persistence Store.

To set up the persistence store for a WSRP portlet producer:

1. In JDeveloper, open the portlet producer application for which you want to set up the persistence store.

2. Expand the portlet project (for example, **Portlets**).

3. Expand the **Web Content** node and then the **WEB-INF** node.

4. Right-click **web.xml** and choose **Open**.

5. Click the **Source** tab.

6. To specify a Database persistence store, add the following code:

```
<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>
```

To specify a File-based persistence store, add the following code:

```
<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>
```

To specify a Consumer persistence store, add the following code:

```
<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>Consumer</env-entry-value>
</env-entry>
```

7. Save the `web.xml` file.

8. Restart your WebLogic Server.

13.3.15.4 Migrating a WSRP Producer Persistence Store

If you want to change the type of persistence store used by your portlet producer, for example when moving from a testing to production environment, you can migrate the existing data from the old to the new persistence store.

This section includes the following topics:

- Migrating a Database or File-Based Persistence Store

- Migrating to or from a Consumer Persistence Store

- Moving a WSRP Portlet Producer

13.3.15.4.1 Migrating a Database or File-Based Persistence Store

The WSRP portlet producer persistence store migration utility, `PersistenceMigrationTool`, enables you to migrate existing data between Database and File-based persistence stores (for example, from a File-based persistence store used for testing to a production Database persistence 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, enabling data to be moved from one database store to another.

> **Note:**
>
> There are several libraries that must be referenced in the classpath when running the `PersistenceMigrationTool` utility:
>
> - `wcs-producer-spi.jar`
>
> - `portlet-utils.jar`
>
> - `portlet-producer-container-common.jar`
>
> - `portlet-producer-container-persistence.jar`
>
> - `oracle-portlet-api.jar`
>
> - `wsrp-container.jar`
>
> - `oracle-portlet-tags.jar`
>
> - `ojdbc6.jar`
>
> The `ojdbc6.jar` library referenced in the classpath must be the same as the one used by your database.

To migrate a database or file-based persistence store:

1. Ensure that the appropriate libraries are referenced in the classpath:

   ```
   ./java -classpath
   ORACLE_COMMON_HOME/modules/oracle.wccore/portlet-producer-container-
   persistence.jar:
   ORACLE_COMMON_HOME/modules/oracle.wccore/portlet-producer-container-common.jar:
   ORACLE_COMMON_HOME/modules/oracle.adf.share.ca/adf-share-base.jar:
   ORACLE_COMMON_HOME/modules/oracle.wccore/portlet-utils.jar:
   ```

```
ORACLE_COMMON_HOME/modules/oracle.wccore/wcs-producer-spi.jar:
ORACLE_COMMON_HOME/modules/oracle.adf.share/adflogginghandler.jar:
DB_ORACLE_HOME/jdbc/lib/ojdbc6.jar
```

2. Run the `PersistenceMigrationTool` using the following syntax:

```
java oracle.portlet.server.containerimpl.PersistenceMigrationTool
-sourceType [file |db]
-destType [file |db]
{-sourcePath [dir |
-sourceUsername username -sourcePassword password -sourceDatabase db
 -sourceDriver srcDriver]}
{-destPath [dir |destUsername username -destPassword password -destDatabase db
 -destDriver dstDriver]}
[-debug]
```

where:

- `sourceType` indicates whether the source store is in a file (`file`) or database (`db`). 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 (`file`) or database (`db`). 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 persistence store. This argument is required when `sourceType` is `file`.

- `sourceUsername` is the database user name for a persistence store database. This argument is required when `sourceType` is `db`.

- `sourcePassword` is the database password for a persistence store database. This argument is required when `sourceType` is `db`.

- `sourceDatabase` is the name of a persistence store database. This argument is required when `sourceType` is `db`.

- `sourceDriver` is the name of a database driver to use. For example, for a SQL Server database, the database driver is `com.microsoft.sqlserver.jdbc.SQLServerDriver`. If you do not specify a value for this argument, the migration tool attempts to determine the correct driver to use from the `sourceDatabase` argument.

- `destPath` is the location of a file-based persistence store. This argument is required when `destType` is `file`.

- `destUsername` is the database user name for a persistence store database. This argument is required when `destType` is `db`.

- `destPassword` is the database password for a persistence store database. This argument is required when `destType` is `db`.

- `destDatabase` is the name of a persistence store database. This argument is required when `destType` is `db`.

- `destDriver` is the name of a database driver to use. For example, for a SQL Server database, the database driver is `com.microsoft.sqlserver.jdbc.SQLServerDriver`. If you do not

specify a value for this argument, the migration tool attempts to determine the correct driver to use from the `destDatabase` argument.

- `debug` turns on full logging through standard output to enable users to diagnose issues that arise when the tool runs.

The example below demonstrates running the `PersistenceMigrationTool` utility. In this example, preferences from a File store are copied to a Database store.

```
./java -classpath

ORACLE_COMMON_HOME/modules/oracle.wccore/portlet-producer-container-persistence.jar:
ORACLE_COMMON_HOME/modules/oracle.wccore/portlet-producer-container-common.jar:
ORACLE_COMMON_HOME/modules/oracle.adf.share.ca/adf-share-base.jar:
ORACLE_COMMON_HOME/modules/oracle.wccore/portlet-utils.jar:
ORACLE_COMMON_HOME/modules/oracle.wccore/wcs-producer-spi.jar:
ORACLE_COMMON_HOME/modules/oracle.adf.share/adflogginghandler.jar:
DB_ORACLE_HOME/jdbc/lib/ojdbc6.jar
oracle.portlet.server.containerimpl.PersistenceMigrationTool
-sourceType file \
-sourcePath /data/prefs
-destType db \
-destUsername scott \
-destPassword tiger \
-destDatabase abc.mycompany.com:1521:yourdatabase \
```

where:

- *ORACLE_COMMON_HOME* is your Oracle Common home

- *DB_ORACLE_HOME* is your database Oracle home if it is on the same machine as the WebCenter Portal Oracle home. If the database Oracle home is on a separate machine, then you must copy the *DB_ORACLE_HOME/jdbc/lib* directory into a temporary directory on the WebCenter Portal Oracle home and reference the `ojdbc6.jar` library from this temporary directory in the classpath.

13.3.15.4.2 Migrating to or from a Consumer Persistence Store

You cannot use the persistence store migration utility to migrate to or from a consumer persistence store. To migrate to or from a consumer persistence store, you must export the data from one producer from the consumer and import into another.

To migrate a consumer persistence store:

1. Export the producer metadata from your consumer application to an EAR file.

   This contacts the remote producer to get customization data, and so on. You can use WLST to do this, see `exportPortletClientMetadata` in *WebCenter WLST Command Reference*.

2. Either remap the producer connection to another producer, which can be using any persistence store type, or change the preference store configuration of the current producer, and restart it.

3. Import from the exported EAR file back to your consumer application. This pushes the relevant metadata back to the producers, which then store it in their configured persistence store.

13.3.15.4.3 Moving a WSRP Portlet Producer

If you move a portlet producer to a different server, you may also have to move the persistence store for the producer.

After installing the new producer, move the persistence store according to the following:

- Consumer persistence store—Because the data is stored with the consumer, there is no requirement to move the persistence store.

- Database persistence store—Perform one of the following:

  - Configure the new producer to point the WebLogic Server data source to the same database as the original producer.

  - Migrate the persistence store using the Persistence Store Migration Utility (see Migrating a WSRP Producer Persistence Store for more information).

- File persistence store—Perform one of the following:

  - Configure the new producer to point the same file system location as the original producer.

  - Migrate the persistence store using the Persistence Store Migration Utility (see Migrating a WSRP Producer Persistence Store for more information).

Finally, update the URL of the producer registration by using Enterprise Manager Fusion Middleware Control or the WLST command `setWSRPProducerRegistration`.

13.4 Testing JSR 286 Portlets

Before making your portlets available in a production environment, it is highly advisable to test them first to make sure that they behave as expected.

This section includes the following topics:

- How to Run a WSRP Portlet Producer on Integrated WebLogic Server

- How to Deploy a WSRP Portlet Producer to the Integrated WebLogic Server

- How to Hide or Remove the WSRP Test Page

13.4.1 How to Run a WSRP Portlet Producer on Integrated WebLogic Server

The Integrated WebLogic Server (Integrated WLS) provides a quick and easy way of testing your portlets because it is preconfigured so that you can run applications within JDeveloper without needing to create deployment profiles.

To test a JSR 286 portlet on Integrated WLS:

1. In JDeveloper, open the portlet producer application that owns the portlet that you want to test.

2. Expand the portlet project (for example, **Portlets**).

3. Expand the **Web Content** node and then the **WEB-INF** node.

4. Right-click **portlet.xml** and choose **Run**.

Running `portlet.xml` triggers the packaging and deployment of your portlet producer application on an Integrated WLS instance named after the application.

5. Check the IntegratedWebLogicServer - Log window to monitor the deployment progress. The 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 13-4 displays in a browser window.

Figure 13-4   WSRP Producer Test Page

6. While the application is running, you can switch back and forth between JDeveloper and your browser to make changes at design time in your application, save the changes, and then refresh the test page in your browser.

7. When the producer is successfully running, you should register it with an application and add one or more portlets to a page to check that it is working correctly. 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.

8. To stop an Integrated WLS instance, in the Run Manager tab, select **DefaultServer** and click the red **Stop** icon. When the instance stops, the application is undeployed, and therefore, becomes unavailable.

> **Tip:**
>
> You can also stop an Integrated WLS instance by clicking the red **Stop** icon in the IntegratedWebLogicServer - Log window and selecting **DefaultServer**.

What Happens When You Run a WSRP Portlet Producer on Integrated WebLogic Server

These configurations vary depending upon the portlet requirements.

- 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 286 portlet producer 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 286 portlets are added to the `weblogic.xml` file, for example, `oracle.portlet-producer.wsrp`.

When you run a WSRP portlet producer on Integrated WLS, the following happens:

13.4.2 How to Deploy a WSRP Portlet Producer to the Integrated WebLogic Server

When you run a WSRP portlet producer application on the Integrated WLS instance, when the instance stops, the application is undeployed and therefore becomes unavailable. For a more persistent testing scenario, you can deploy your portlet producer application to the Integrated WLS so that it is always available while the Default Server is running.

If you choose this method, then you must first create deployment profiles. If you deploy your application to the Integrated WLS, then the Deployment Configuration dialog displays to enable you to configure and customize deployment settings. The file system MDS repository precreated by JDeveloper displays in the **Repository Name** field.

13.4.3 How to Hide or Remove the WSRP Test Page

For security purposes, you may want to hide the WSRP test page so that it is visible only to administrators, or you may want to remove it entirely.

This section includes the following subsections:

- Hiding the WSRP Test Page

- Removing the WSRP Test Page

Note:

There is also a Webservice test page, which enables you to build up SOAP requests to the producer in a web browser. For WSRP portlet producers this test page is disabled by default. Although this test page does not provide any useful information for testing your portlet producers, you can enable it, if desired, by extracting the `oracle-webservices.xml` file from the deployed producer's WAR file, setting the `expose-testpage` flag to `true` for both WSRP v1 and WSRP v2 producers, and then repacking the WAR and EAR files and redeploying the producer.

13.4.3.1 Hiding the WSRP Test Page

If you do not want all users to be able to see the WSRP test page, you can protect it so that only administrators can see it.

To hide the WSRP test page:

1. In JDeveloper, open the portlet producer application for which you want to hide the test page.

2. Expand the portlet project (for example, **Portlets**).

3. Expand the **Web Content** node and then the **WEB-INF** node.

4. Right-click **web.xml** and choose **Open**.

5. Click the **Source** tab.

6. Add the following code:

```
<security-role>
 <description>AdministratorRole</description>
 <role-name>Admin</role-name>
</security-role>
<security-constraint>
 <display-name>TestPageInfo</display-name>
 <web-resource-collection>
 <web-resource-name>TestPageInfo</web-resource-name>
 <description>Protect the test page servlet.</description>
 <url-pattern>/info/*</url-pattern>
 </web-resource-collection>
 <auth-constraint>
 <description>Administrators</description>
 <role-name>Admin</role-name>
 </auth-constraint>
 <user-data-constraint>
 <transport-guarantee>NONE</transport-guarantee>
 </user-data-constraint>
</security-constraint>
```

7. Save the `web.xml` file.

13.4.3.2 Removing the WSRP Test Page

You can remove the WSRP test page completely.

To remove the WSRP test page, you must edit an element that is injected into the `web.xml` file at packaging time, so you must edit the `web.xml` file in the resulting EAR file.

To remove the WSRP test page:

1. Extract the `web.xml` file from the EAR file created at packaging time and open it in an editor of your choice.

2. Comment out the following code:

    ```
    <servlet-mapping>
     <servlet-name>WSRPTestPage</servlet-name>
     <url-pattern>/info</url-pattern>
    </servlet-mapping>
    ```

3. Save the `web.xml` file and add the edited file to the EAR file.

13.5 Migrating WebLogic Portal Portlets to WebCenter Portal

In general, JSR 286 standard portlets (Java portlets) that were developed with the WebLogic Portal's Eclipse IDE can be moved directly to a WebCenter Portal/ JDeveloper environment. Simply copy all of the portlet artifacts (`portlet.xml`, `.java` files, `.jsp` files, and so on) into a JDeveloper Portlet Producer application project.

Note:

Any WebLogic Portal specific APIs used by the portlet must be rewritten to use WebCenter Portal APIs. WebLogic Portal specific APIs will not work in a WebCenter environment.

Tip:

You can use the WebLogic Portal Export feature to export your Java portlets to an archive file and then import the archive file into your JDeveloper project. This technique may be simpler than manually copying all the portlet artifacts from one environment to another. See Exporting Java Portlets for Use on Other Systems in the *Portlet Development Guide for Oracle WebLogic Portal*.

Moving portlets from a WLP development environment to a WebCenter Portal development environment is not directly supported. In general, this process can involve substantial rewriting or refactoring of the migrated portlet code and related files.

> **Note:**
>
> If a portlet (regardless of type) was capable of running over WSRP in WebLogic Portal, the portlet can be consumed directly from the WebLogic Portal's portlet producer in a WebCenter Portal consumer. See WSRP Interoperability with Oracle WebCenter Portal and Oracle Portal in the *Federated Portals Guide for Oracle WebLogic Portal*.

Below are some general guidance on moving WLP portlets into WebCenter Portal's portlet producer environment.

- **JSP portlets** –Moving JSP portlets directly into a WebCenter Portal project is not supported. You can consider refactoring the JSP portlet into a JSR286 portlet and then migrate it.

- **JSR 168/286 portlets** –Most Java (JSR 168 /286) portlets can be directly imported to a WebCenter Portal portlet producer and run as JSR286 portlets. Some JSR168 portlets that take advantage of specific error conditions guaranteed by the JSR168 specification may need to be run in a JSR168 compatibility mode. See JSR-286/ JSR-168 Portlet Compatibility in the *Portlet Development Guide for Oracle WebLogic Portal*. JSR168 portlets using WLP's proprietary eventing (event subscriptions declared in a `.portlet` file) must be re-written to use JSR286 events.

- **Java Page Flow portlets** –JPF portlets are not supported by WebCenter Portal's portlet producer and must be either consumed from a WebLogic Portal WSRP producer or refactored to become JSR286 or JSF portlets.

- **JSF portlets** –If the portlet is written to the JSR329 JSF Portlet bridge in WLP, it should run on a WebCenter producer with no changes. For portlets using the WLP "native" JSF portlet bridge, the portlet must be consumed from a WebLogic Portal producer or upgraded to a JSF 1.2 portlet using the JSR329 bridge. See also Working With JSF-Java Portlets in the *Portlet Development Guide for Oracle WebLogic Portal*.

- **Clipper portlets** –Clipper portlets are not supported in WebCenter portal, but the web clipping features in WebCenter Portal's pagelet producer provide equivalent functionality.

- **Struts portlets** –Moving Struts portlets directly into a WebCenter Portal project is not supported. You can consider refactoring the JSP portlet into a JSR286 portlet and then migrate it.

- **Content Presenter portlets** –WLP Content Presenter portlets will not work over WSRP and will not work with WebCenter Portal. However, equivalent functionality is available in WebCenter Portal's Content Presenter. See Publishing Content Using Content Presenter in *Building Portals with Oracle WebCenter Portal*.

- **Remote (WSRP) portlets** –Remote (WSRP) portlets consumed in WLP can be consumed in a WebCenter Portal's consumer instead. Remote portlets taking advantage of WLP-specific WSRP features may need modification. For example the custom data transfer feature must be replaced by using events or shared parameters to convey data. See WSRP Interoperability with Oracle WebCenter Portal and Oracle Portal and Configuring WSRP Security Between WLP and a WebCenter Portal: Framework Application in the *Federated Portals Guide for Oracle WebLogic Portal*.

Problems inherent in moving WebLogic Portal portlets directly to a WebCenter Portal project in JDeveloper can include the following:

- **URL Generation** –Some URL types supported by WebLogic Portal do not work in WebCenter Portal, including DesktopURL, CustomEventURL, PageURL, WindowURL, StandalonePortletURL, and possibly others.

- **Events** –The WebCenter Portal consumer does not generate all of the events that the WebLogic Portal framework generates. These unsupported events include Init, LookAndFeelReinit, Notification, Refresh, WindowActivation, WindowDeactivation, and possibly others.

- **Render Dependencies** –WLP render dependencies do not work in WebCenter Portal.

- **WLP Framework APIs** –Many WLP APIs are not supported in WebCenter Portal.

13.6 Files Related to JSR 286 Portlets

This section describes the files that are created for you when you build a JSR 286 portlet. It includes the following topics:

- portlet.xml

- oracle-portlet-tags.jar

- portlet_mode.jsp

- portlet_name.java

- portlet_nameBundle.jar

- web.xml

13.6.1 portlet.xml

`portlet.xml` defines the characteristics of your JSR 286 portlet. For complete details on `portlet.xml`, you should see the Java Portlet Specification available at:

```
http://jcp.org/en/jsr/detail?id=168
```

The example below 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 **portlet.xml**

```
<portlet>
 <description xml:lang="en">JSR 286 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>
 <portlet-mode>help</portlet-mode>
 <portlet-mode>about</portlet-mode>
 </supports>
 <supported-locale>en</supported-locale>
 <resource-bundle>jsrportlet.resource.MapPortletBundle</resource-bundle>
```

```
<portlet-info>
<title>Map Portlet</title>
<short-title>Map</short-title>
<keywords/>
<portlet-preferences>
<preference>
<name>portletTitle</name>
</preference>
</portlet-preferences>
<security-role-ref>
<role-name>viewer</role-name>
</security-role-ref>
</portlet>
```

For JSR 286 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>` describes the portlet, providing details to the end user.

- `<portlet-name>` uniquely identifies the portlet within the Portlet Producer 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. For JSF portlets created using the Oracle JSF Portlet Bridge, this is `oracle.portlet.bridge.adf.application.ADFBridgePortlet`.

- `<expiration-cache>` the default duration (in seconds) of the expiration cache.

- `<init-param>` defines initialization parameters for configuring the behavior of the portlet. The Oracle JSF Portlet Bridge uses initialization parameters to identify the entry points for the different portlet modes supported by the portlet, for example, for View mode:

```
<init-param>
 <name>javax.portlet.faces.defaultViewId.view</name>
 <value>/myPage.jspx</value>
</init-param>
```

Initialization parameters for other WebCenter Portal-supported modes are:

- Edit mode: `javax.portlet.faces.defaultViewId.edit`

- Help mode: `javax.portlet.faces.defaultViewId.help`

- About mode: `javax.portlet.faces.defaultViewId.about`

- Config mode: `javax.portlet.faces.defaultViewId.config`

- Edit Defaults mode: `javax.portlet.faces.defaultViewId.edit_defaults`

- Preview mode: `javax.portlet.faces.defaultViewId.preview`

- Print mode: `javax.portlet.faces.defaultViewId.print`

> **Note:**
>
> The value for the `defaultViewId` is relative to the application context root and must always start with a `/`. In the example provided, the value for `defaultViewId.view` is `/myPage.jspx`.
>
> If you add a `defaultViewId` for other portlet modes, then you must also add the mode to the `<supports>` tag. For example, `<portlet-mode>HELP</portlet-mode>`.

- `<supports>` provides information about the portlet modes supported for each content type. Portlet modes supported by WebCenter Portal include: `edit`, `help`, `about`, `config`, `edit_defaults`, `preview`, and `print`.

- `<supported-locale>` lists the locales supported by the portlet at runtime.

- `<resource-bundle>` is the fully qualified class name of the resource bundle used to provide language specific portlet information, such as title and keywords.

- `<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.

- `<portlet-preferences>` are preference attribute definitions.

- `<supported-processing-event>` are events that the portlet can receive.

- `<supported-publishing-event>` are events that the portlet raises.

- `<supported-public-render-parameter>` is a public render parameter supported by the portlet.

- `<container-runtime-option>` an option for defining additional runtime behavior.

- `<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>
```

13.6.2 oracle-portlet-tags.jar

`oracle-portlet-tags.jar` is the Oracle implementation of the JSP tag library defined by the Java Portlet Specification.

13.6.3 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 need `view.jsp` and `edit.jsp` in your *portlet_name*\html directory. For JSR 286 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 Portlet Modes.

13.6.4 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.

13.6.5 portlet_nameBundle.jar

portlet_name`Bundle.jar` is a resource bundle class, containing translation of the strings used by the portlet.

13.6.6 web.xml

`web.xml` is a Java EE standard descriptor that contains details about Web applications. For more information about `web.xml`, see *Developing Fusion Web Applications with Oracle Application Development Framework*.

Part IV

Working with Pagelets

This part contains the following chapters:

- Introduction to Pagelets

- Creating Pagelet Producer Objects

- Working with Pagelets and Gadgets

- Creating Custom Pagelets

- Using Pagelets in Web Applications

- Creating Pagelets: Examples and Advanced Topics

14

Introduction to Pagelets

This chapter provides an introduction to creating pagelets with Oracle WebCenter Portal's Pagelet Producer.

This chapter includes the following sections:

- About Creating Pagelets with Pagelet Producer
- Configuring Pagelet Producer Settings

14.1 About Creating Pagelets with Pagelet Producer

This section provides an introduction to Pagelet Producer concepts and features. It describes how you can use Pagelet Producer, its underlying architecture and components, and introduces key concepts.

This section includes the following topics:

- Overview
- Pagelet Producer Architecture
- Requirements

14.1.1 Overview

A pagelet is a reusable user interface component. Any HTML fragment can be a pagelet, but pagelet developers can also write pagelets that are parameterized and configurable to dynamically interact with other pagelets, and respond to user input. Pagelets can be run on any web page, including within a portal or other web application. Pagelets can be used to expose platform-specific portlets in other Web environments. Oracle WebCenter Portal's Pagelet Producer (previously known as Oracle WebCenter Ensemble) provides a collection of useful tools and features that facilitate dynamic pagelet development.

Pagelet Producer lets you leverage investments in existing web infrastructure by making it usable in new ways. Pagelet Producer can be used to consume:

- Web UI from applications that do not expose integration APIs or portlets out of the box
- Standard portlets such as WSRP (JSR-168 or JSR-286), Oracle PDK-Java, CSP, and WebParts

It can also deliver web UI to WebCenter Portal, Sites, or pretty much any HTML page with the added benefits of role-based access control to Pagelet Producer resources, and reverse proxying of web resources with HTML markup parsing, clipping, injection and event handling (inter-pagelet communication).

Pagelet Producer provides:

- Tools for consuming existing web UI and re-using it as portable components called *pagelets*

- Alternatives for integrating non-standard/non-compliant containers and components that do not support portlet standards. For example, you can:

  - Capture functionality from web applications that do not expose portlets

  - Make existing WSRP, Oracle PDK-Java, or Content Syndication Protocol (CSP) portlets available for use in a non-portal setting

- Injectors and parsers to modify HTML markup

- A JavaScript framework for inter-pagelet communication and event handling

14.1.2 Pagelet Producer Architecture

This section describes the Pagelet Producer architecture and its key components.

This section includes the following topics:

- Pagelet Producer Architecture and Components

- Pagelet Producer Key Concepts

- Pagelet Producer Console

- Using HTTP and Content Syndication Protocol

- Pagelet Producer Proxy

14.1.2.1 Pagelet Producer Architecture and Components

Figure 14-1 shows a high-level view of the Pagelet Producer components and their interactions. Figure 14-1 also reflects the way in which the pagelet request is being handled in Pagelet Producer. The request is received by the Proxy Servlet that maps incoming requests to the resource. Next, the request is authenticated. If the user can access the requested resource, pagelet processing starts by requesting HTML markup from the back-end system. After the markup is received it is modified using any defined clippers, parsers, and so forth.

Figure 14-1   Pagelet Producer - High-Level Architecture
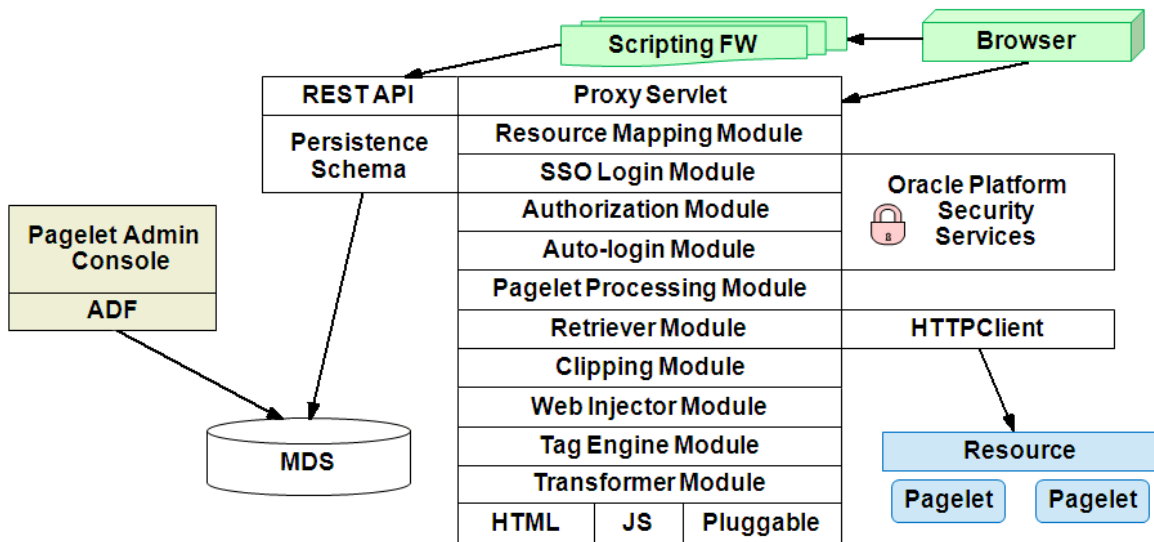
Figure 14-2 provides a different view of component interaction:

Figure 14-2   Pagelet Producer - Interaction Flow
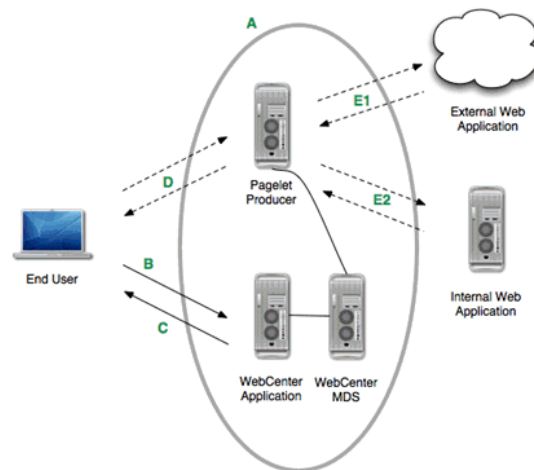
A: A Pagelet Producer application runs on a J2EE application server. Note that Pagelet Producer does not require the WebCenter schema, but it does require an MDS Instance (typically shared with WebCenter).

B: User calls a WebCenter Portal page that contains a pagelet.

C: WebCenter Portal returns the page, plus the JavaScript that will call Pagelet Producer to render the pagelet.

D: The JavaScript makes a render call to Pagelet Producer.

E: Pagelet Producer can call external (**E1**) or internal (**E2**) web applications as sources of HTML content. Pagelet Producer transforms the content (i.e., rewrites the URL) and returns it to the browser for insertion into the page. Pagelet Producer itself can be externally accessible (i.e., in the DMZ).

14.1.2.2 Pagelet Producer Key Concepts

The following key concepts are useful when working with Pagelet Producer:

- *Resources* are core objects used to register applications within Pagelet Producer, including stand-alone web applications, Pagelet Producers and OpenSocial containers. Creating a resource allows the proxy to map internal applications to external URLs, manage authentication, and transform applications. a web application as a Pagelet Producer resource allows you to do the following:

  - Proxy internal web applications to external addresses

  - Manage authentication, both at the proxy level and at the resource level

  - Transform proxied web applications, including URL rewriting

  For more information about creating resources, see Creating Resources

- *Pagelets* are sub-components of a web page accessed through Pagelet Producer that can be injected into any proxied application. Any application on a Pagelet Producer resource that returns markup can be registered as a pagelet, which can then be displayed in WebCenter Portal, or any other web application.

  Any HTML fragment can be a pagelet. Pagelet developers can create pagelets that are parameterized and configurable, dynamically interact with other pagelets, and respond to user input using Asynchronous Javascript and XML (AJAX) patterns.

  For more information about creating pagelets, see Creating Pagelets .

Pagelet Producer registration is dynamic. Additions and updates to existing producers are immediately available. In most cases, it is not necessary to restart WebCenter Portal or the managed server.

14.1.2.3 Pagelet Producer Console

The **Pagelet Producer Console** is a browser-based administration tool used to create and manage the various objects in your Pagelet Producer deployment. From the Console you can register web applications as resources, create pagelets, manage proxy and transformation settings, and more.

The Pagelet Producer Console is accessible from any web browser at the following URL:

```
http://<host>:<port>/pagelets/admin
```

The Pagelet Producer Console can also be launched in accessibility mode at:

```
http://<host>:<port>/pagelets/admin/accessible
```

During runtime, the Pagelet Producer Console is also accessible from the WebCenter Portal Shared Assets tab.

Note:

Pagelet Producer Console supports the standard administration languages and Dutch only. If you configure the browser language to something other than one of these languages, it will revert to the language defined for the current server.

For more information about using the Pagelet Producer Console to configure Pagelet Producer and create objects, see Configuring Pagelet Producer Settings and Creating Pagelet Producer Objects. For information about using the Pagelet Producer Console to register producers and migrate pagelet data, see Managing the Pagelet Producer in *Administering Oracle WebCenter Portal*.

14.1.2.4 Using HTTP and Content Syndication Protocol

HTTP is a protocol used mostly for transferring Web page content and XML between a server and a client. Content Syndication Protocol (CSP) is a platform-independent protocol based on the open standard of HTTP 1.1 that extends HTTP defining the syntax of communication between Pagelet Producer and external CSP resources.

Services on external resources communicate with Pagelet Producer using CSP. For example, when a browser requests a page, Pagelet Producer makes simultaneous requests to each external resource to retrieve the pagelet content for the page. The external resource reads the current user's preferences from the CSP headers sent by Pagelet Producer and sends back the appropriate HTML. Pagelet Producer inserts the HTML into the page markup. Any images stored in the Image Service are retrieved and displayed by the browser.

This section contains the following subsections:

- HTTP

- Content Syndication Protocol

14.1.2.4.1 HTTP

HTTP communication is made up of requests and responses. Requests and responses are essentially lists of name-value pairs of metadata in headers, along with an optional body. The body is the data that is being transferred (an HTML page or XML file). The metadata in the headers is information about the request or response itself (for example, what language the content is in, or how long the browser should cache it). The request and response each contain specific information as described below. For more detailed information on HTTP, see RFC 2616 (`http://www.faqs.org/rfcs/rfc2616.html`).

The client sends the server an **HTTP Request**, asking for content. The request body is used only for requests that transfer data to the server, such as POST and PUT.

HTTP Request Format:

```
[METHOD] [REQUEST-URI] HTTP/[VERSION]
[fieldname1]: [field-value1]
[fieldname2]: [field-value2]
[request body, if any]
```

HTTP Request Example:

```
GET /index.html HTTP/1.1
Host: www.xmlns.oracle.com
User-Agent: Mozilla/3.0 (compatible; Opera/3.0; Windows 95/NT4)
Accept: */*
Cookie: username=JoeSmith
```

The server sends back an **HTTP Response** that contains page content and important details, such as the content type, when the document was last modified, and the server type. The response contains an error message if the requested content is not found.

HTTP Response Format:

```
HTTP/[VERSION] [CODE] [TEXT]
[fieldname1]: [field-value1]
[fieldname2]: [field-value2]
[response body, if any (document content here)]
```

HTTP Response Example:

```
HTTP/1.0 200 Found
Last-modified: Thursday, 20-Nov-97 10:44:53
Content-length: 6372
Content-type: text/html
<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 3.2 Final// EN'><HTML>
...followed by document content...
```

Custom HTTP headers can be configured to include specialized information.

> **Note:**
>
> Header size limits are controlled by the server that hosts the code. The standard limit for IIS/ASP is 60K. Java Application Servers range from 2K to 10K. These limits are generally configurable; see your server documentation for more information.

Services can also access standard HTTP headers, such as the Set-Cookie header or HTTP 1.1 basic authentication header. If you want to investigate HTTP further, you can view all the headers being passed back and forth between your browser and Web server using logging (as described in Debugging Pagelets). HTTP is used in conjunction with SSL to serve up secure content. Single sign-on (SSO) also uses HTTP headers for basic authentication.

14.1.2.4.2 Content Syndication Protocol

Content Syndication Protocol (CSP) extends HTTP and defines proprietary headers to pass settings between Pagelet Producer and external CSP resources (specifically, Oracle WebCenter Interaction portlets). CSP outlines how these services use HTTP to communicate and modify settings. The latest version of the CSP protocol is 1.4, and is available from:

http://docs.oracle.com/cd/E13158_01/alui/wci/docs103/devguide/references/CSP1.4.pdf

The custom CSP headers used to communicate system and user configuration variables (see Table 14-1) can also be used by pagelets.

Table 14-1   Pagelet Producer Headers

Header Name	Description
User ID	The User ID of the currently logged in user. This value can be used to determine if the session has expired. If UserID=2, the default 'Guest' user is logged in; any other user's session has ended.
User Name	The name of the logged in user. The user's name can be used to personalize display or pre-fill form fields.
Image Service URL	The URL to the root virtual directory of the Image Service in the user's implementation of Pagelet Producer. This location should be used for all static images used in services.
Stylesheet URL	The URL to the current user's style sheet. In each implementation of Pagelet Producer, the UI is customized. In some portals, users can choose between a selection of stylesheets. Using these styles ensures that pagelets appear in the style of the current user's implementation of Pagelet Producer.

Table 14-1   (Cont.) Pagelet Producer Headers

Header Name	Description
Pagelet ID	The ID for the current resource (pagelet), and the instance ID for the current pagelet. This value is useful for appending to the names of HTML forms and client-side JavaScript functions to ensure unique form and function names on the page to avoid name conflicts.
Host Page URL	The URL to the page that hosts the pagelet. Preference pages need this URL to return the user to the correct page after settings are configured. This value can also be used to allow a single pagelet to display different content on different pages.

14.1.2.5 Pagelet Producer Proxy

Pagelet Producer acts as a proxy server, brokering transactions between client computers and external resources. This configuration is typically used to serve content to clients that would otherwise be unable to access the external resource, but it can also be used to impose additional security restrictions on the client, including the use of policies. The proxy hides the external resource so that the content appears to come directly from the proxy server.

This architecture makes Pagelet Producer the single point of access for content, and allows external resources to reside on a private network or behind a firewall. As long as Pagelet Producer can connect to the external resource, users can view the content, even if they cannot access it directly. To the browser, Pagelet Producer appears to be the source of content on the external resource.

When a user interacts with a service, any request made to a URL in the proxy is automatically rerouted through Pagelet Producer. To the user, the content appears to come from Pagelet Producer; the external resource is an unknown back-end system.

There are many benefits to this configuration. The most useful to services are:

- **Dynamic functionality and personalization**: Pagelet Producer intercepts request from pagelets, which allows the response to be modified dynamically, using information from MDS, the request, or the response"

- **Security**: Services can allow users to access content that is not publicly available. Files stored on a secure server can be made available by including specific URLs in the configuration of the proxy.

> **Warning:**
>
> The proxy is a powerful feature and can compromise security if incorrectly configured. Allowing direct access to a external resource that hosts unprotected private content could create a dangerous security gap.

- **Performance**: Pagelet Producer caches proxied content, decreasing response time for end users and improving performance on the external resource. While proxying works efficiently for content like HTML, it is generally not appropriate for binary data like static images. Images do not need to be transformed, and proxying large images can adversely affect performance. This is one reason the Image Service should be used to prevent routing static images through the proxy.

Note that all URLs to other proxied CSP or Web resources will be proxied by default unless specified in the **Do not proxy hosts** field. For more information about configuring proxying, see How to Configure Proxy Settings.

Keep the following warnings and best practices in mind when implementing services that use the proxy:

- **URL transformation**: Pagelet Producer must transform code so that proxied URLs open correctly. Before Pagelet Producer sends a response, it parses the HTML and looks for URLs containing the **HTTP Proxy Server URL** field prefix (see How to Configure Proxy Settings). Pagelet Producer transforms any URLs that should be proxied before returning the response to the client. Relative URLs are transformed to point to the correct location.

- **Scripting limitations**: JavaScript constructs that dynamically create URLs can cause problems, because they are run after content is already transformed. VBScript is not transformed by the proxy; you can continue to use dynamic scripts and VBScript as long as your code is proxy-aware. Parsers and Web injectors can be used to implement targeted control over URL rewriting. For details on parsers and Web injectors, see Modifying Pagelet Functionality at Runtime.

- **URL encoding**: It is a best practice to encode all headers that are URLs to prevent unexpected transformation. In JSP, encode all URLs that are written. If the code writes URLs in the body of a page (for example, a link to a preferences page) it should be encoded. The standard Java servlet command `response.encodeURL()` is the preferred method, but you can also use `URLEncoder.encode(url)`. In the .NET framework, the `HttpUtility.URLEncode` class provides the necessary functionality. Note that for .NET, there is no need to encode the redirect URL; this is handled automatically by the back end.

14.1.3 Requirements

Install WebCenter Portal following the instructions in the *Oracle Fusion Middleware Installation Guide for Oracle WebCenter*. Make sure that all dependent components are also installed and configured, including WebCenter Content Server, the discussions server, and Pagelet Producer.

14.2 Configuring Pagelet Producer Settings

This section explains important configuration settings that affect all Pagelet Producer resources and how to set them using the Settings pages of the Pagelet Producer Console. For an introduction to the Pagelet Producer Console, see Pagelet Producer Console.

The Settings pages are described in the following topics:

- How to Configure a WCI Data Source

- How to Configure Logging Settings

- How to Configure Proxy Settings

- How to Configure Transform Settings

- How to Configure CSP Settings

- How to Configure Kerberos Settings

- How to Configure OpenSocial Settings

14.2.1 How to Configure a WCI Data Source

A CSP login token is required to consume WCI resources, and a data source is required for that login token to be propagated from Pagelet Producer to CSP portlets. Pagelet Producer generates a login token that is sent to all remote portlets to propagate user identity and validate incoming requests. Follow the steps in Configuring JDBC Data Sources in *Administering JDBC Data Sources for Oracle WebLogic Server* to configure the data source. Note that the data source must be named `wcidb`.

14.2.2 How to Configure Logging Settings

The Logging page lets you set logging levels for Pagelet Producer components.

Figure 14-3   Pagelet Producer Console: Settings - Logging
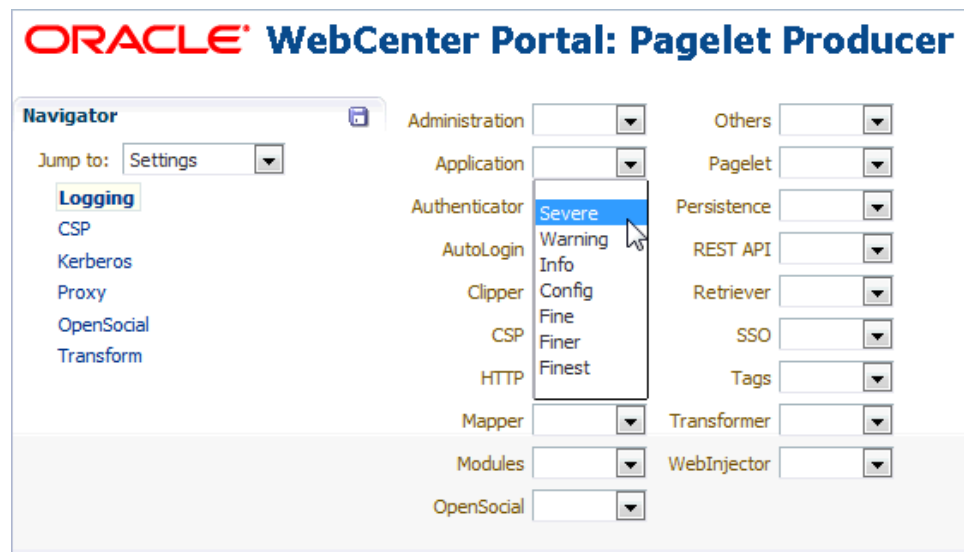
To enable logging for any component, choose one of the following logging levels:

- Severe

- Warning

- Info

- Config

- Fine

- Finer

- Finest

Logging messages can be viewed in the managed server log files. In Oracle WebLogic Server, this would be *domain_home*/servers/*managed server name*/*managed server name*.log and *managed server name*-diagnostic.log.

14.2.3 How to Configure Proxy Settings

The Proxy page lets you define a list of URLs that will not be proxied, and set the URL, user name, and password for the HTTP proxy if required.

You must restart Pagelet Producer after changing the proxy settings.

> **Note:**
>
> If an HTTP proxy is required for external connections in the deployment environment, initial server startup will report connection errors. These errors are not an indication of problems with the environment; they indicate that some external OpenSocial libraries cannot be loaded remotely. The errors can be resolved by configuring the proxy settings as described above and restarting the server.

> **Note:**
>
> The HTTP Proxy Server URL entered on the Proxy page is applied to all applications running on the server that hosts Pagelet Producer (on Oracle WebLogic Server, this setting is applied to all applications running on the `WC_Portlet` managed server). Oracle WebLogic Server users should pay particular attention to this setting and make sure that the WLS Administrative Server host and all clustered managed servers are included in the "Do not proxy hosts" list.

Figure 14-4   Pagelet Producer Console: Settings - Proxy

The Proxy page contains the following settings:

- The **Do not proxy hosts** field lets you define a list of semicolon-separated list of URLs that will not be proxied (wildcards are allowed). For Oracle WebLogic Server deployments, be sure to include the WLS Administrative Server host and all clustered managed servers in this list.

- The **HTTP Proxy Server Password** field lets you set the password for the proxy server.

- The **HTTP Proxy Server URL** field lets you define the proxy server URL.

- The **HTTP Proxy Server Username** field lets you define the username with which to access the proxy server.

14.2.4 How to Configure Transform Settings

The Transform page lets you enter the path to the credential vault provider and configure secure and insecure ports for Pagelet Producer. This page also lets you choose whether or not to log pre- and post-transformation content.

Figure 14-5   Pagelet Producer Console: Settings - Transform

The Transform page contains the following fields:

- The **Credential Vault Plugin Path** field lets you define a path to the credential vault, where plug-ins are stored.

- The **Nonsecure Port** field lets you define a non-secure port for Pagelet Producer.

> **Note:**
>
> The Nonsecure Port defaults to 8889. If Pagelet Producer is deployed on a different port, change this entry and restart the server.

- The **Secure Port** field is where you define a secure port for Pagelet Producer.

- The **Trace Transformation** option, when checked, turns on logging for pre- and post-transformation content. Note that this option, although useful for debugging, should not be turned on in production environments.

14.2.5 How to Configure CSP Settings

CSP is a platform-independent protocol based on the open standard of HTTP 1.1. CSP defines the syntax of communication between Pagelet Producer and external resources. It also defines custom headers as well as outlines how services use HTTP to communicate and modify settings.

The CSP page lets you configure the Oracle WebCenter Interaction Credential Mapper, SOAP API service, and image service.

> **Note:**
>
> This page may be ignored if Oracle WebCenter Interaction is not present in your deployment. These settings are used for backwards compatibility with CSP portlets written for Oracle WebCenter Interaction.

14.2.6 How to Configure Kerberos Settings

The Kerberos page is used to define the paths to the Kerberos configuration files (`java.security.krb5.conf` and `java.security.auth.login.conf`). These two configuration files are needed to configure the Kerberos realm and KDC to instruct HTTPClient where to retrieve the Kerberos Service ticket.

14.2.7 How to Configure OpenSocial Settings

The OpenSocial page lets you configure Pagelet Producer for use with OpenSocial gadgets.

Figure 14-6   Pagelet Producer Console: Settings - OpenSocial
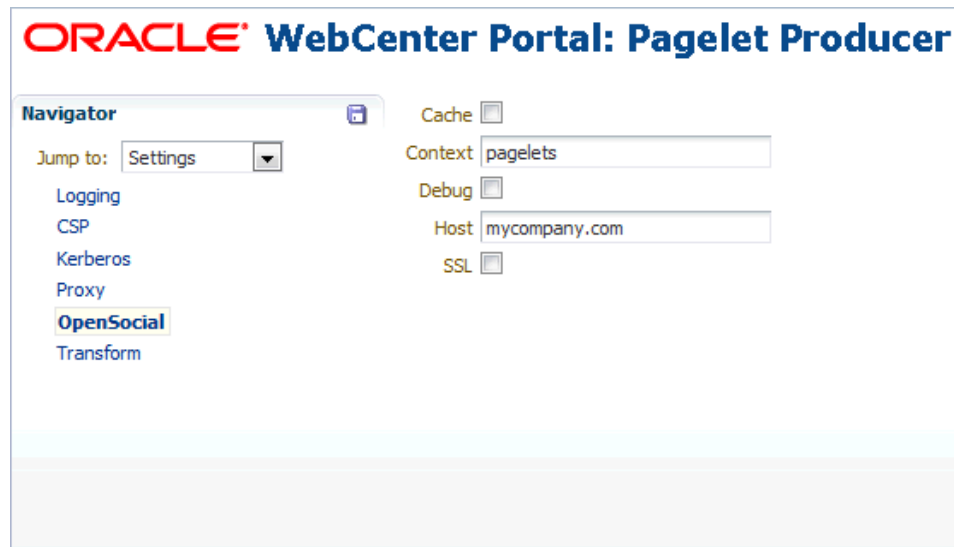
The OpenSocial page includes the following settings:

- The **Cache** option enables Pagelet Producer internal caching for OpenSocial gadgets.

- The **Context** field defines the context to which Pagelet Producer is deployed. This value should be left at the default setting ("pagelets") unless a change is required by the OpenSocial container. For details on changing the context, see Redeploying Pagelet Producer to a Different Context in *Administering Oracle WebCenter Portal*.

- The **Debug** option enables debugging for OpenSocial gadgets (disables JavaScript obfuscation).

- The **Host** field should contain the fully-qualified domain name of Pagelet Producer host. This value should be retrieved automatically from the environment; if it is not, restart the server to pick up the correct configuration settings. As with the Context setting, this value should be left at the default unless a change is required by the OpenSocial container.

- The **SSL** option enables SSL for OpenSocial gadgets.

Note:

You must also configure the secure (HTTPS) and nonsecure (HTTP) ports before importing OpenSocial gadgets. For details on these settings, see How to Configure Transform Settings.

15

Creating Pagelet Producer Objects

This chapter describes how to create and configure the various objects in a Pagelet Producer deployment, including resources, pagelets, Web injectors, parsers and hosted files.

As explained in Pagelet Producer Console, the Pagelet Producer Console is a browser-based administration tool used to create and manage the various objects in a Pagelet Producer deployment. This chapter describes the settings required for each type of object.

This chapter includes the following sections:

- Creating Resources

- Creating Pagelets

- Creating Web Injectors

- Creating Custom Parsers

- Creating Hosted Files

15.1 Creating Resources

Resources are applications registered with Pagelet Producer. An application as a resource allows the proxy to map internal applications to external URLs, manage authentication, and transform application functionality.

To create a new resource, use the steps below:

1. Select the **Resources** option from the dropdown list in the Pagelet Producer Console navigator.

2. Click on any existing resource and click the **Create** icon in the navigator toolbar. (This button is only enabled when you have selected an object type that can be created.)

Figure 15-1   Pagelet Producer Console - Create Resource

3. In the **Select Producer Type** dialog box, choose the type of producer that the resource will be configured to proxy from the dropdown list:

- **Web**: Any standard web application

- **CSP**: CSP portlet provider application (for use with Oracle WebCenter Interaction)

- **WSRP/JPDK**: WSRP or Oracle PDK-Java portlet producers (before you create a resource of this type, register the associated producer as described in Managing the Pagelet Producer in *Administering Oracle WebCenter Portal*).

- **OpenSocial**: OpenSocial Container

Figure 15-2   Select Producer Type Dialog
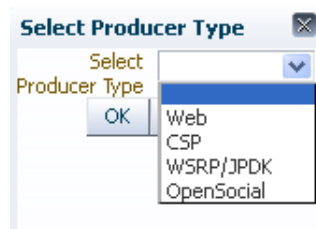
4. Click **OK**. An entry called "<new>" will be added to the list of resources. This new resource will include the necessary configuration pages for the producer type chosen.

5. Configure the resource in the Pagelet Producer Console. The required configuration pages differ based on the type of producer. To save your changes at any time, click the **Save** icon in the navigator toolbar.

- How to Configure Web and CSP Resources

- How to Configure WSRP and Oracle PDK-Java Resources

- How to Configure OpenSocial Resources (OpenSocial Gadget Producers)

Once the resource is defined, create the pagelets and other objects within the resource. For more information, refer to the following topics:

- Creating Pagelets

- Creating Web Injectors

- Creating Custom Parsers

- Creating Hosted Files

15.1.1 How to Configure Web and CSP Resources

This section describes how to configure web and CSP resources.

This section includes the following topics:

- How to Configure General Settings

- How to Configure CSP settings

- How to Configure Policy Settings

- How to Configure Autologin

- How to Configure Autologin: Form Login

- How to Configure Autologin: Basic Login and NTLM Login

- How to Configure Autologin: Kerberos Login

- How to Configure Autologin: Authentication Sources

- How to Configure Headers

This section uses the *welcome_resource* created when Pagelet Producer is installed as an example.

15.1.1.1 How to Configure General Settings

Use the **General** page (see Figure 15-3) to enter basic information about the resource.

Figure 15-3   Pagelet Producer Console: Resource - General Page
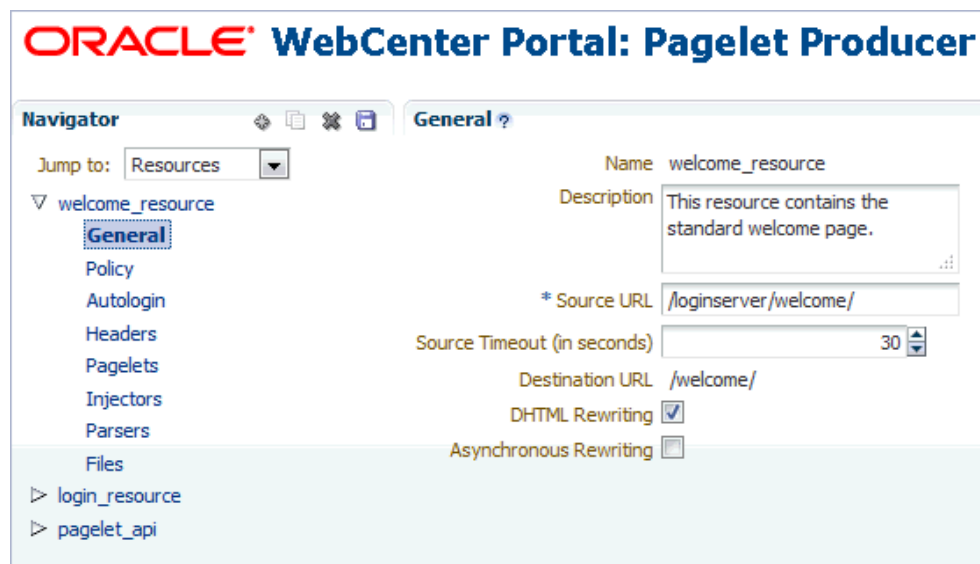
- Enter a **Name** for the resource.

- Enter a **Description** for the resource (optional).

- In the **Source URL** field, type the URL to the location of the web application resource to be proxied. For example, `http://internalServer/foo/`.

Note:

If you are configuring an ADF web application as a resource, the Source URL cannot be any more specific than `http://hostname:portnumber/context-root/`.

- By default, Pagelet Producer attempts to connect to the resource for 30 seconds before returning an error message. To change this value, enter a new **Source Timeout** period in seconds.

- In the **Destination URL** field, type the relative path to be used to access the resource. This path will be used to create an URL to the resource on the server that hosts Pagelet Producer.

- To enable Dynamic HTML, choose **DHTML Rewriting**. This option supports URLs that are not in the original HTML returned from the server, but are added by DHTML. In most cases, this option should be enabled.

- To enable rewriting of DHTML through asynchronous Ajax calls select **Asynchronous Rewriting**.

15.1.1.2 How to Configure CSP settings

By default, the CSP login token is not passed to the proxied resource. To enable this feature, choose **Send CSP Login Token**. Note that the CSP page is only available for CSP resources.

15.1.1.3 How to Configure Policy Settings

The Policy page (see Figure 15-4) lets you limit access to a resource to specific roles within WebCenter Portal.

Figure 15-4   Pagelet Producer Console: Resource - Policy Page

The J2EE container hosting Pagelet Producer (such as Oracle WebLogic Server) is responsible for establishing the role memberships associated with the current user. A resource can specify multiple roles on the Policy page, and users will be allowed access if they are a member of any of the specified roles; otherwise they will be directed to a suitable J2EE container delegated authentication page to establish the required credentials. If no roles are entered in the list, anonymous access is allowed, and the resource is termed as an "anonymous resource".

Note:

The role name(s) entered on this page must match those created in the J2EE container (such as Oracle WebLogic Server).

15.1.1.4 How to Configure Autologin

The Autologin feature allows Pagelet Producer to supply credentials to applications automatically. The Autologin page (see Figure 15-5) lets you configure authentication information for all users who access the resource.

Figure 15-5   Pagelet Producer Console: Resource - Autologin Page
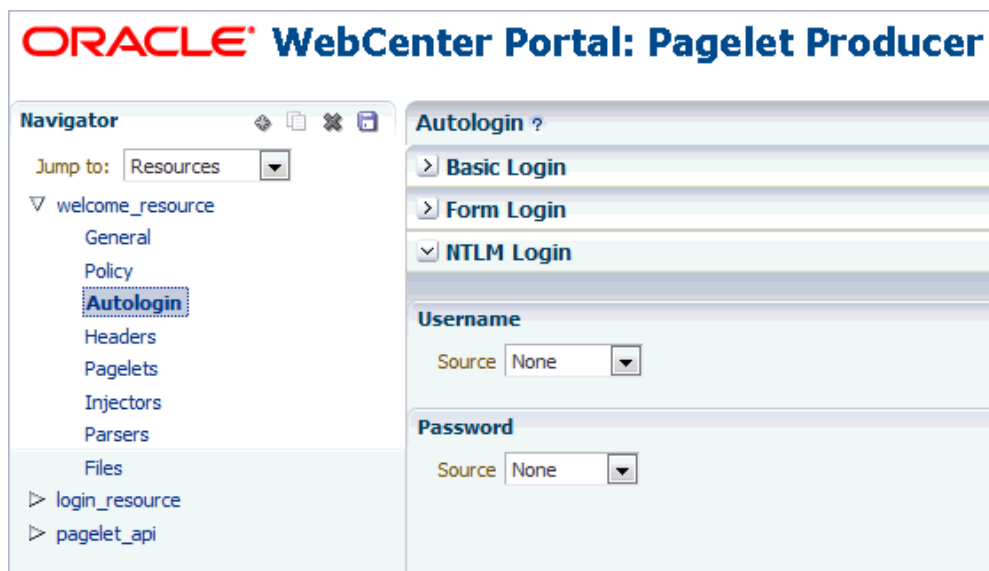
The following sections describe how to configure credential mapping for authentication:

- How to Configure Autologin: Form Login describes how to configure autologin for a resource that prompts for authentication with an HTML form.

- How to Configure Autologin: Basic Login and NTLM Login describes how to configure autologin for a resource that prompts for authentication with basic authentication or NTLM.

- How to Configure Autologin: Kerberos Login describes how to configure autologin for a resource that prompts for authentication with Kerberos.

- How to Configure Autologin: Authentication Sources describes the static, user profile, and credential vault authentication field sources.

15.1.1.4.1 How to Configure Autologin: Form Login

This section describes how to configure Autologin for a resource that prompts for authentication with an HTML form.

1. On the **Autologin** page for the resource, expand the **Form Login** section (Figure 15-6).

Figure 15-6   Autologin Page - Form Login

2. The login page can be identified by an URL or a regular expression. In the **Login Form Identification** section, choose one of the following options:

   - If the login form is located at a static URL, select **URL** and enter the URL in the **Value** field. You can choose to **Automatically Detect Form Fields** on the page or enter them manually as described in step 4 below.

   - If the login form is dynamic, select **RegEx** and type the regular expression pattern into the box.

3. Set the login form action. In the **Form Submit Location** section, choose one of the following options:

   - If the login form action is a static URL, select **URL** and type the URL into the box. Choose the action for the form submission: POST or GET.

   - If the login form is dynamic, select **RegEx** and type the regular expression pattern into the box.

4. To map fields from the form to authentication field sources, either click **Automatically Detect Form Fields** in the Login Form Identification section as described above or enter them manually using the process below:

   a. Click **Create** to add a new row to the **Form Fields** list.

   b. Enter the name of the HTML form input in the **Field Name** box.

   c. For details on how to configure the **Source** and **Value** properties, see How to Configure Autologin: Authentication Sources.

   Note:

   Sensitive fields should be stored securely using the credential vault (User Vault or Secured).

   ---

   d. To delete field mappings, click **Delete**.

5. The logout page and login error pages can also be identified by an URL or a regular expression. In the **Logout Page Identification** and **Login Error Page Identification** sections, choose one of the following options:

- If the page is located at a static URL, select **URL** and type the URL into the field provided.

- If the page is dynamic, select **RegEx** and type the regular expression pattern into the field provided.

15.1.1.4.2 How to Configure Autologin: Basic Login and NTLM Login

This section describes how to configure autologin for a resource that prompts for authentication with basic authentication or NTLM.

1. On the **Autologin** page for the resource, expand the **Basic Login** or **NTLM Login** section.

Note:

Basic authentication transmits passwords as plain text, and therefore, it must not be used in production systems. Further, it is strongly recommended that the underlying transport is HTTPS.

---

2. In the **Username** and **Password** sections, choose the appropriate authentication source and enter a value as necessary. For details on how to configure these properties, see How to Configure Autologin: Authentication Sources.

15.1.1.4.3 How to Configure Autologin: Kerberos Login

This section describes how to configure autologin for a resource that prompts for authentication using Kerberos. For information on defining basic Kerberos settings, see Configuring Pagelet Producer Settings.

1. On the **Autologin** page for the resource, expand the **Kerberos Login** section.

2. In the **Username** and **Password** sections, choose the appropriate authentication source and enter a value as necessary. For details on how to configure these properties, see the next section, How to Configure Autologin: Authentication Sources.

3. In the **SPN** field, enter the Service Principal Name (SPN) for the Kerberos account, in the format `http://hostname_with_kerberos`. (Before the Kerberos authentication service can use an SPN to authenticate a service, the SPN must be registered on the account object that the service instance uses to log on.)

15.1.1.4.4 How to Configure Autologin: Authentication Sources

Authentication sources define the source for login fields. Table 15-1describes each of the authentication field source values:

Table 15-1   Pagelet Producer Authentication Sources

Field	Description
Unsecured	Uses the provided authentication information for all users accessing the resource. Type the static value in the field provided.

Table 15-1   (Cont.) Pagelet Producer Authentication Sources

Field	Description
Secured	Uses the field value is entered by an administrator in the Pagelet Producer Administration Console. The value is stored in the Credential Vault and is shared among all users, including non-authenticated (anonymous) users. The field key is auto-generated and displayed as a read-only field in the Administration Console.
User Vault	Prompts the user for credentials the first time the resource is accessed. The supplied credentials are encrypted and stored in the credential vault, and each subsequent access to the resource is authenticated with the stored credentials. The field key is auto-generated and displayed as a read-only field in the Administration Console.In the second field, enter the name of the credential vault to use, or leave the entry as "default" to use the server vault.
	Note: When you choose `User Vault`, the user will be presented with an error page that includes a link: "Click here to access pagelet directly." This link opens the authentication dialog. This is a known issue and occurs only the first time the resource is accessed by a user.
Generated	(Form Login only) The field value is taken from the backend server response markup.

15.1.1.5 How to Configure Headers

The Headers page (see Figure 15-7) lets you choose request and response headers that should be dropped from the HTTP that is provided by Pagelet Producer.

Figure 15-7   Pagelet Producer Console: Resource - Headers Page
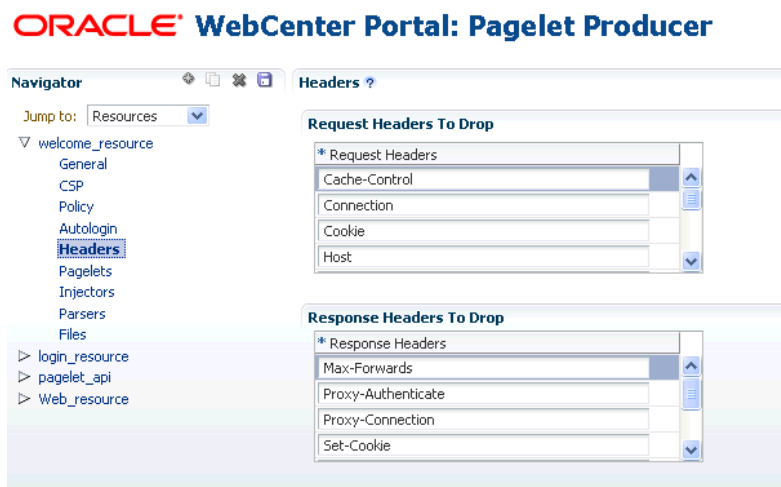
Some header elements should be blocked from being passed to back-end applications. For example, when using delegated (third-party SSO) authentication, the SSO system might insert some headers that need not be passed to the back-end applications. When passed, these headers might interfere with the back-end application functionality.

The following headers are dropped by default:

Request Headers	Response Headers
Cache-Control	Max-Forwards
Connection	Proxy-Authenticate
Cookie	Proxy-Connection
Host	Set-Cookie
Max-Forwards	Trailer
Pragma	Transfer-Encoding
Proxy-Connection	Upgrade
Proxy-Authorization	
TE	
Trailer	
Transfer-Encoding	
Upgrade	

The Content-Length header is always implicitly dropped, because manipulating content during the proxying operation renders the content length invalid in almost all cases.

To add a header to the list, click **Create** and enter the header name in the field provided.

Once the web or CSP resource is defined, you can create pagelets and other objects. For more information, see the following topics:

- Creating Pagelets

- Creating Web Injectors

- Creating Custom Parsers

- Creating Hosted Files

15.1.2 How to Configure WSRP and Oracle PDK-Java Resources

This section describes how to configure WSRP/JPDK resources based on WSRP and Oracle PDK-Java portlet producers.

Note:

Before you can create a resource based on a WSRP or Oracle PDK-Java portlet producer, you must register the producer as described in Managing the Pagelet Producer in *Administering Oracle WebCenter Portal*.

This section includes the following subsections:

- How to Configure General Settings

- How to Configure Policy Settings

15.1.2.1 How to Configure General Settings

Use the **General** page to enter basic information about a WSRP or JPDK resource.

- Choose the portlet producer type from the **Portlet Producer** dropdown list. This list is populated with the producers that have been registered as described in Managing the Pagelet Producer in *Administering Oracle WebCenter Portal*.

- Enter a **Name** for the resource.

- Enter a **Description** for the resource (optional).

15.1.2.2 How to Configure Policy Settings

The Policy page lets you limit access to a resource to specific roles within WebCenter Portal.

The J2EE container hosting Pagelet Producer (such as Oracle WebLogic Server) is responsible for establishing the role memberships associated with the current user. A resource can specify multiple roles on the Policy page, and users will be allowed access if they are a member of any of the specified roles; otherwise they will be directed to a suitable J2EE container delegated authentication page to establish the required credentials. If no roles are entered in the list, anonymous access is allowed, and the resource is termed as an "anonymous resource".

> **Note:**
>
> The role name(s) entered on this page must match those created in the J2EE container (such as Oracle WebLogic Server).

Once the WSRP or JPDK resource is defined, you can create pagelets and Web injectors. See the following topics for more information:

- Creating Pagelets

- Creating Web Injectors

15.1.3 How to Configure OpenSocial Resources (OpenSocial Gadget Producers)

This section describes how to configure OpenSocial resources based on OpenSocial gadget producers.

- How to Configure General Settings

- How to Configure Policy Settings

> **Note:**
>
> Configure Pagelet Producer for use with OpenSocial before creating an OpenSocial resource. For more information, see How to Configure OpenSocial Settings.

15.1.3.1 How to Configure General Settings

Use the **General** page to enter basic information about the resource.

- Enter a **Name** for the resource.

- Enter a **Description** for the resource (optional).

15.1.3.2 How to Configure Policy Settings

The Policy page lets you limit access to a resource to specific roles within WebCenter Portal.

The J2EE container hosting Pagelet Producer (such as Oracle WebLogic Server) is responsible for establishing the role memberships associated with the current user. A resource can specify multiple roles on the Policy page, and users will be allowed access if they are a member of any of the specified roles; otherwise they will be directed to a suitable J2EE container delegated authentication page to establish the required credentials. If no roles are entered in the list, anonymous access is allowed, and the resource is termed as an "anonymous resource".

> **Note:**
>
> The role name(s) entered on this page must match those created in the J2EE container (such as Oracle WebLogic Server).

Once the OpenSocial resource is defined, you can create pagelets and files. Refer to the following topics for more information:

- Creating Pagelets
- Creating Hosted Files

15.2 Creating Pagelets

The pagelets collection lists the pagelets associated with the resource. To create a new pagelet, select the **Pagelets** section under the resource you want to use in the Pagelet Producer Console and click the **Create** icon in the toolbar. A pagelet called "<new>" will be added to the list. To modify an existing pagelet, click the pagelet name.

> **Note:**
>
> Do not make updates to the seeded resources (`welcome_resource`, `login_resource`, and `pagelet_api`). These were not intended to be modified.

This section contains the following topics:

- How to Configure General Settings
- How to Configure Preferences
- How to Configure Parameters
- How to Configure the Clipper
- How to Access the Pagelet and Preference Editor

15.2.1 How to Configure General Settings

Enter a **Name** for the pagelet and the **Library** name with which to associate the pagelet. (A pagelet library is a user-defined way to group related pagelets.) Enter a **Description** for the pagelet (optional).

- **For Web and CSP pagelets**, type the relative path to the pagelet in the **URL Suffix** field (do not include the Source URL prefix you entered for the resource). If you leave the URL Suffix blank, then the entire resource will be considered the pagelet.

- **For pagelets based on WSRP or Oracle PDK-Java portlets**, choose the portlet on which to base the pagelet from the **Portlet** dropdown list. This list is populated with the portlets on the WSRP or Oracle PDK-Java portlet producer associated with the parent resource. Any public parameters associated with the portlet will automatically be imported as pagelet parameters. For more information on support for WSRP or Oracle PDK-Java portlets, see Working with Pagelet Chrome for WSRP and Oracle PDK-Java Portlets.

- **For pagelets based on OpenSocial gadgets**, enter the location of the gadget XML schema in the **Gadget URL** field. Click the **Import Gadget Metadata** button to import the following information from the XML schema:

  - **Gadget name**: This value will be imported into the Description field on the General page.

  - **Gadget user preferences**: The pagelet parameters on the How to Configure Parameters page will be populated with the gadget's user preferences.

  For more information on support for OpenSocial gadgets, see Working with OpenSocial Gadgets.

To block access to the pagelet, choose **Disabled**. If the pagelet is included on any pages, it will display a simple error message.

15.2.2 How to Configure Preferences

On the Preferences page (see Figure 15-8), enter the relative URLs to any preference pages required by the pagelet: Global, Customize, or Personalize. Do not include the Source URL prefix you entered for the resource. (As noted above, for OpenSocial gadgets with user preferences, a default entry will be created; this entry should not be modified.)

The Preferences page is not used by WSRP or Oracle PDK-Java based pagelets.

Figure 15-8   Pagelet Producer Console: Pagelets - Preferences Page
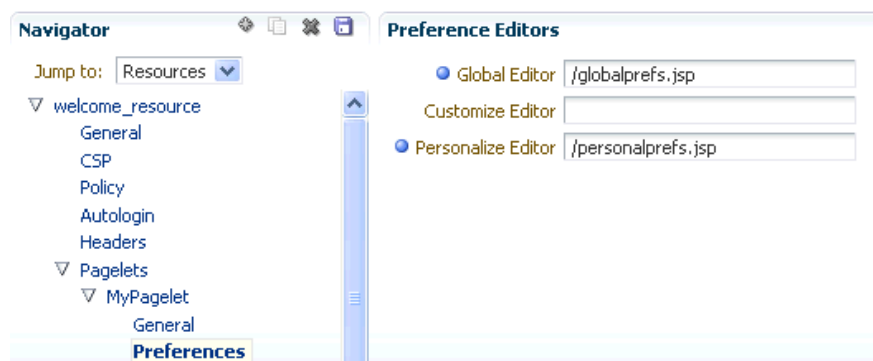
15.2.3 How to Configure Parameters

Data can be passed to pagelets using pagelet parameters. Parameters pass name-value pairs to the pagelet application as described below.

On the Parameters page (see Figure 15-9), enter the **Parameters** that should be passed to the pagelet.

To add a parameter, click **Create**.

- Enter the **Name** of the parameter.

- If the parameter is required by the pagelet, select the **Required** checkbox.

- Choose the appropriate data **Type**: string or numeric.

- Enter a **Description** (optional).

Figure 15-9   Pagelet Producer Console: Pagelets - Parameters Page
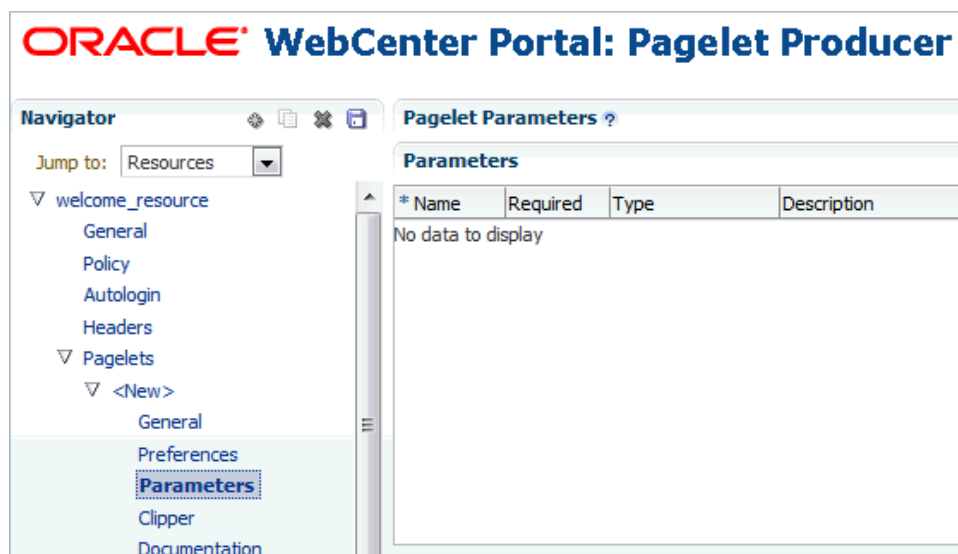
15.2.4 How to Configure the Clipper

Clipping lets you create a pagelet by clipping a portion of a larger Web page in a proxied application. A news Web page, for example, may contain a box listing the latest headlines. By identifying the containing HTML for that box, you can clip only the headlines and serve that subset of the news Web page as a pagelet.

To create a clipper, select the **Clipper** section under the pagelet and click the **Create** icon in the toolbar. A new clipper will be created with two configuration pages:

- On the **General** page, enter a name for the clipper.

- On the **Content** page (see Figure 15-10), define the clipper content.

Figure 15-10   Pagelet Producer Console: Clipper - Content Page
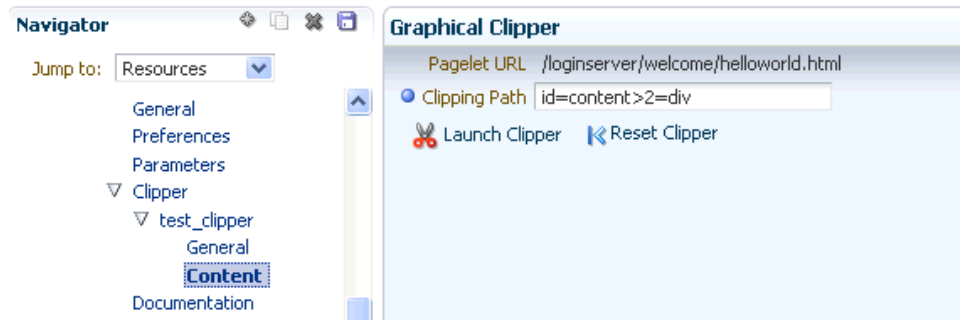
- To use a graphical tool to select page content, click **Launch Clipper**.

- To specify HTML tag attributes that describe the section to be clipped, expand the **Advanced Clipper** section (Figure 15-11) and enter the **Clipping Tag Name** and associated attribute(s).

Figure 15-11   Advanced Clipper
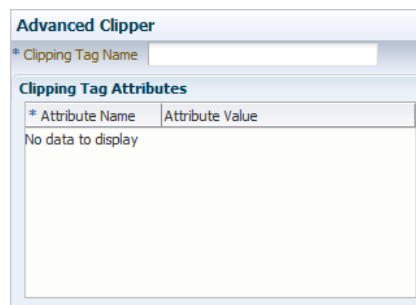
Keep the following in mind when using the clipper:

- If the back-end resource is accessed over HTTPS, make sure the Pagelet Producer Console is also accessed over a secure port.

- If the clip source is protected by a login form or other form of authentication, make sure to configure Autologin for the parent resource as described in How to Configure Autologin. If you are using the vault to store credential values (see How to Configure Autologin: Authentication Sources), make sure to capture the credentials prior to using the clipper.

- If you are having problems with the clipper, make sure the configured pagelet URL can be loaded by the browser without redirects. If necessary, change the pagelet suffix to reflect the final URL loaded by the browser after following all the redirects.

- Images and code overwritten using hosted files cannot be clipped. (For more information about hosted files, see Creating Hosted Files.)

- If the graphical clipper cannot be used (for example, if the page ID is not available), use the Advanced Clipper to define the page region to clip using a regular expression.

15.2.5 How to Access the Pagelet and Preference Editor

Use the Documentation page (see Figure 15-12) to display sample code with which to access the pagelet and preference editor using either JavaScript or the REST API.

Figure 15-12   Pagelet Producer Console: Pagelets - Documentation Page
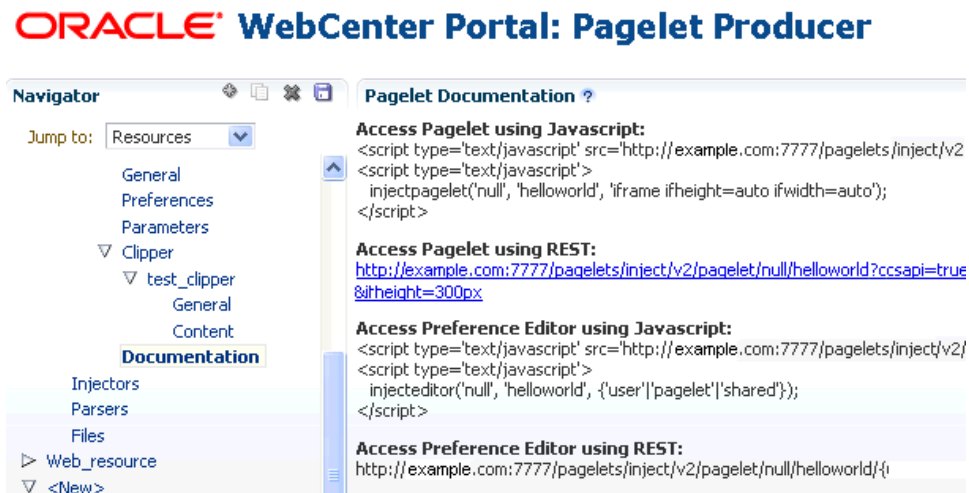
15.3 Creating Web Injectors

A Web injector inserts content into a specified location in the proxied resource page. The content may be any text, including HTML, CSS, JavaScript, and pagelet declarations. An empty injector may also be used to remove unwanted content from a page. Injectors cannot be created for OpenSocial resources.

To create a Web injector, select the **Injectors** section under the resource you want to use and click the **Create** icon in the toolbar. A new injector called "<new>" will be added to the list. Injectors can then be configured using the General and Content configuration pages as described in the following topics:

- How to Configure General Settings

- How to Inject Content

Note:

Do not make updates to the seeded resources (`welcome_resource`, `login_resource`, and `pagelet_api`). These were not intended to be modified.

15.3.1 How to Configure General Settings

Use the General page (see Figure 15-13) to configure basic settings for the injector.

Enter a **Name** for the Web injector.

The injector can be applied to a subset of the resource by typing a URL pattern into the **URL Filter** box. The injector will be applied only to those URLs within the resource that begin with the text in the URL Filter box. If the box is empty or contains only a '/', the injector will be applied to the entire resource.

To restrict the injector to specific kinds of content, type a comma separated list of MIME types in the **MIME Filter** box. For example, *text/html* restricts the injector to HTML content, while *text/css* only restricts the injector to CSS content.

In the **Inject Location** section, define where in the resource's output the injection will be made in relation to a unique string. Enter the unique string in the field provided and choose **Before**, **After** or **Replace** to define where to put the content relative to the string. If you choose to replace the string, you can use the **Enclose tag** to replace both the string and the enclosing tag. You can choose to ignore the case of the string by selecting **Ignore case**.

Figure 15-13   Pagelet Producer Console: Injectors - General Page
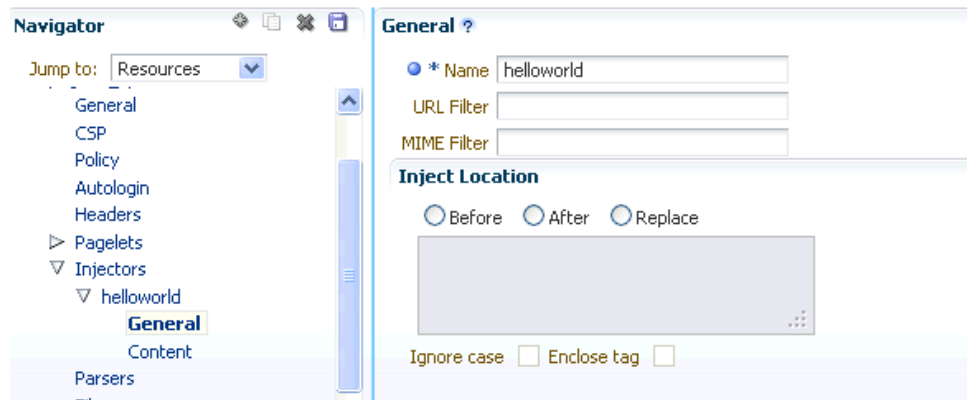
15.3.2 How to Inject Content

Use the Content page to define content to be injected using the injector. Content to be injected may be any text, including HTML, CSS, JavaScript, and pagelet declarations.

For example, the following code could be injected at the beginning of a page. The example registers a handler function with the page load event and then uses the handler to modify the page markup (by finding and hiding the header and footer).

```
<script type="text/javascript">
if (window.addEventListener) {
window.addEventListener('load', hideHeaderFooter, false);
} else if (document.attachEvent) {
window.attachEvent('onload', hideHeaderFooter);
}
function hideHeaderFooter() {
var header =null;
 //find the header table by class
if (document.getElementsByClassName) {
header =document.getElementsByClassName('page_header')[0];
 }else {
//for older versions of IE
 var tables =document.getElementsByTagName('table');
 for (var table in tables) {
if (table.class =='page_header') {
header =table;
 break;
 }
}
}
```

```
if (header != null) {
header.style.display ='none';
 }
//now find and hide the footer (easier to find, since it has an id)
 var footer =document.getElementById('t23PageFooter');
 if (footer != null) {
footer.style.display ='none';
 }
}
</script>
```

15.4 Creating Custom Parsers

Custom parsers allow you to supplement or change built-in logic for parsing content and finding URLs. When the built-in parsers fail to identify URLs or identify sections that must not be rewritten as URLs, custom parsers can be used to change the default behavior. Note that parsers cannot be created for WSRP or Oracle PDK-Java portlet producers or OpenSocial gadget producers.

Note:

Do not make updates to the seeded resources (`welcome_resource`, `login_resource`, and `pagelet_api`). These were not intended to be modified.

To create a custom parser, select the **Parsers** section under the resource you want to use and click the **Create** icon in the toolbar to display the Parser's General page (see Figure 15-14):

- Enter a **Name** for the parser.

- The parser can be applied to a subset of the resource by typing a URL pattern into the **URL Filter** box. The parser will be applied only to those URLs within the resource that begin with the text in the URL Filter box. If the box is empty or contains only a '/', the parser will be applied to the entire resource.

- To add a new parsing rule, click **Create** to add a new row to the **Fragment Locations** section.

- In the **Regular Expressions** column, enter the regular expression for identifying the URL fragment that should be transformed. The first grouping expression (in parentheses) identifies the fragment, and the rest of the expression provides the context for finding it.

- Choose the **Fragment Type** to define how the selected location should be parsed:

  - **Static URLs** are transformed on the server.

  - **Dynamic URLs** are transformed using JavaScript on the client.

  - **HTML Fragment** and **Javascript Fragment** types are used for content that is embedded in another content type, such as XML.

  - **No Rewriting** specifies a location that should not be searched for URLs. This option is used to prevent rewriting of markup mistakenly identified as URLs.

- Click the **Save** icon in the navigator toolbar.

Figure 15-14   Pagelet Producer Console: Parsers - General Page
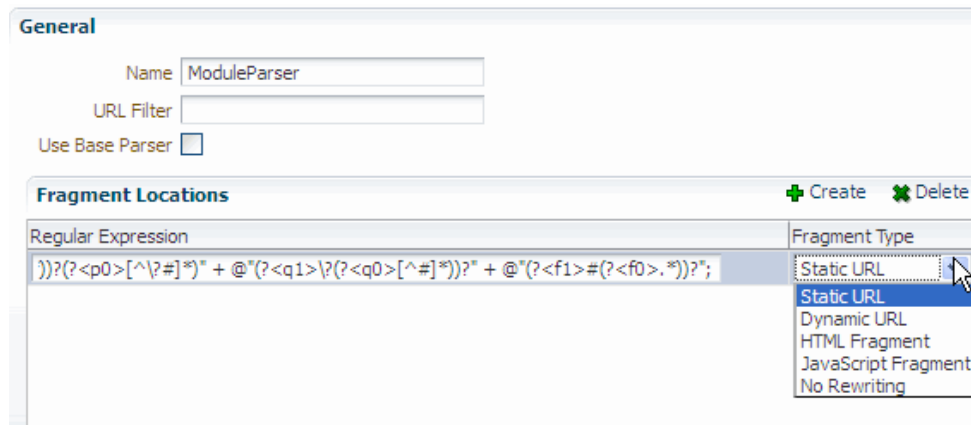
For example, the regular expression `XMLFile=(.*?)"` would identify URLs to XML files defined within a tag, as in `<embed src="/i/flashchart/anychart_5/swf/OracleAnyChart.swf?>`**`XMLFile=http://apex.oracle.com/pls/apex/apex_util.flash?`**
`p=53557:1:74175370346201:FLOW_FLASH_CHART5_R45192463162785599619`
`_en``"`.

15.5 Creating Hosted Files

Pagelet Producer can host all types of content (HTML, JavaScript, CSS, etc.) and present the file at a virtual URL location. Hosted files can be used for a range of purposes:

- Overwrite content and functionality in a proxied application, by uploading files and configuring them to use the same URL as the original file.

- Use hosted files in injectors to insert images or content into a proxied application.

- Host pagelet files on the Pagelet Producer server.

Note:

Do not make updates to the seeded resources (`welcome_resource`, `login_resource`, and `pagelet_api`). These were not intended to be modified.

To upload a file, select the **Files** section under the resource you want to use and click the **Create** icon in the toolbar.

- On the **General** page (see Figure 15-15), enter the relative path to the file in the **Name** field. Do not use a leading forward-slash ("/"). The directory structure of the Files collection in the navigator is updated to match the path to the file.

  Enter the **MIME type** of the file.

Figure 15-15   Pagelet Producer Console: Files - General Page
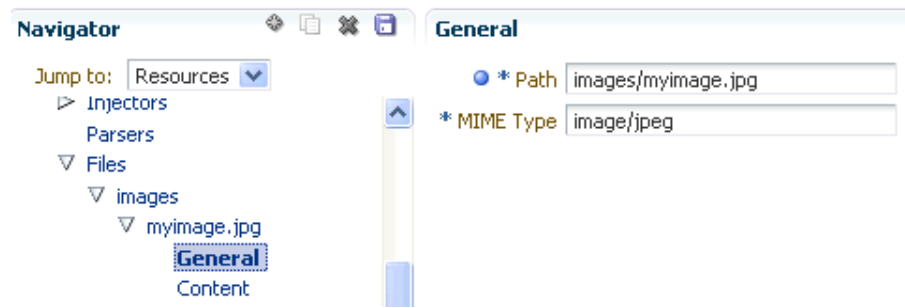
- On the **Content** page, enter the path to the file or click **Browse** to navigate to the file.

  Click **Upload** to upload the file to the Pagelet Producer server. If you entered text or html as the MIME type, you can also use the editor on the Content page to enter or edit file content. (The editor is only available for text and html files.) If you entered an image type as the MIME type, the uploaded image will be displayed on the Content page.

  > **Note:**
  >
  > Text files (text/plain) uploaded to the Pagelet Producer must be saved as UTF-8.

- Click the **Save** icon in the navigator toolbar.

After the file is uploaded, it will be available for use in injectors and pagelets at the following URL: `http://<host_name>:<port_number>/pagelets/<resourcename>/<filepath>`.

For example, if the file was uploaded under the "welcome_resource" resource and the name for the file was entered as "images/myimage.jpg" the path to the hosted file on the Pagelet Producer server would be: `http://<host_name>:<port_number>/pagelets/welcome_resource/images/myimage.jpg`

Keep the following in mind when working with hosted files:

- The hosted file feature should not be used for bulky files.

- If you choose to host OpenSocial gadget XML files, the files must be placed under an anonymous resource (with no security policy) or gadget functionality will not work correctly.

- Hosted files cannot be created for WSRP or Oracle PDK-Java portlet producers.

16

Working with Pagelets and Gadgets

This chapter describes how to create pagelets with Oracle WebCenter Portal's Pagelet Producer.

This chapter includes the following sections:

- Working with OpenSocial Gadgets
- Working with Pagelet Chrome for WSRP and Oracle PDK-Java Portlets

16.1 Working with OpenSocial Gadgets

Any OpenSocial gadget reachable by the Pagelet Producer server can be registered as a pagelet and used in any Web application, including a portal. To support OpenSocial gadgets, you must first define an OpenSocial container as described in How to Configure OpenSocial Settings.

Pagelet Producer supports most of the standard OpenSocial APIs excluding OAuth. The complete OpenSocial API reference documentation can be found here:

```
http://docs.opensocial.org/display/OSD/Specs
```

Pagelet Producer also allows gadgets to store preferences, retrieve WebCenter Portal profile and connection information, and access a user's activity stream using OpenSocial APIs.

This section contains the following topics:

- How to Configure Authentication
- How to Store User Preferences
- How to Access WebCenter Portal Profile Information
- How to Access a User's Activity Stream
- How to Use Gadget Eventing
- Example: How to Consume an External OpenSocial Gadget
- Example: How to Consume a Local OpenSocial Gadget

16.1.1 How to Configure Authentication

In order for gadgets to request user-level data (preferences or people connections), the end user's identity must be established. If any OpenSocial gadgets need to access user-level data from the server, you must configure a security policy for the parent OpenSocial resource in the Pagelet Producer Console (see How to Configure Policy Settings). The first time a user accesses an OpenSocial gadget, a login page will be

presented. After the initial login, subsequent requests for OpenSocial gadgets will use the established user identity.

16.1.2 How to Store User Preferences

OpenSocial gadgets may use user preferences to store data at the container. User preferences are scoped to a particular user and may optionally be scoped to an appId (the gadget appId is the pagelet context ID). If you choose to use the OpenSocial gadget.Prefs API, the user preferences will be scoped to the user and pagelet instance. Alternatively, you may use the opensocial.DataRequest API to manage preferences at the user level that can be shared with other pagelets.

When registered as a pagelet, a gadget's user preferences are treated as pagelet preferences. In WebCenter Portal, for example, non-hidden user preferences can be edited by the end user using the Personalize button. Additionally, to simulate preferences shared between users, you may pass in user preferences via pagelet parameters. Note that a pagelet preference, if set, will always override the corresponding pagelet parameter (in other words, personalization takes precedence over customizations).

Outside of WebCenter Portal, gadget-backed pagelets are provided with a simple chrome that displays the gadget title and buttons for accessing the preference editor and minimizing/maximizing the gadget. The chrome may be suppressed by passing in the value of 'none' to the chrome parameter in the Pagelet Inject API. The preference editor UI supports all four types of user preferences:

- string: rendered as a text field

- bool: rendered as a checkbox

- enum: rendered as a dropdown list

- list: rendered as a text field (values must be separated with a pipe "|" character)

16.1.3 How to Access WebCenter Portal Profile Information

OpenSocial gadgets can query the current user's profile data and people connections via the standard OpenSocial API. To use this feature, you must manually target the WebCenterDS data source to the `WC_Portlet` server as described in Managing the Pagelet Producer in *Administering Oracle WebCenter Portal*.

> **Note:**
>
> The OpenSocial API cannot be used to update profile or connection information.

The supported user profile fields are listed in Table 16-1.

Table 16-1   User Profile Fields

OpenSocial Field	Type	Description
aboutme	string	A general statement about the person.

Table 16-1   (Cont.) User Profile Fields

OpenSocial Field	Type	Description
addresses	Plural-Field <Address>	A physical mailing address for the person.
appData	Plural-Field <AppData>	A collection of AppData keys and values, used for preferences.
birthday	Date	The birthday of the person. The value must be a valid Date. The year may be set to 0000 when the age of the person is private or the year is not available.
emails	Plural-Field <string>	Email address for the person.
location	string	Physical address for the person.
name	Name	The broken-out components and formatted version of the person's real name.
organizations	Plural-Field <Organization>	Current or past organizational affiliation of the person.
phoneNumbers	Plural-Field <string>	Phone number for the person. In addition to the standard canonical values for type, this field defines the additional values mobile, fax, and pager.
photos	Plural-Field <string>	URL to a photo of the person. The value must point to an actual image file (e.g. a GIF, JPEG, or PNG image file) rather than to a Web page containing an image. Note that this field should only be used to download profile photos of the contact suitable for display when describing the contact.
preferredUserna me	string	The preferred username of this person on sites that ask for a username (e.g., jsmarr or daveman692).
status	string	The person's status, headline or shoutout.
thumbnailUrl	string	The person's photo thumbnail URL, specified as a string. This URL MUST be fully qualified.

16.1.4 How to Access a User's Activity Stream

OpenSocial gadgets can operate on a user's activity stream using OpenSocial APIs. The following operations are supported:

- get activities

- create activity

The following operations are not supported:

- update activity

- delete activity

The supported activity stream fields are listed in Table 16-2.

Table 16-2   Activity Stream Fields

OpenSocial Field	Type	Description
appId	Object-Id	The application with which the activity is associated.
body	string	An optional expanded version of the activity. Bodies may only have the following HTML tags: <b> <i>, <a>, <span>, but this formatting may be ignored.
externalId	Object-Id	An optional ID generated by the posting application.
id	Object-Id	An ID that is permanently associated with the activity.
postedTime	string	The time at which the activity took place in milliseconds since the epoch.
priority	number	A number between 0 and 1 representing the relative priority of the activity in relation to other activities from the same source.
title	string	The primary text of the activity. Titles may only have the following HTML tags: <b> <i>, <a>, <span>, but this formatting may be ignored.
userId	Object-Id	ID of the user for whom the activity is defined.

16.1.5 How to Use Gadget Eventing

Pagelet Producer supports the OpenSocial pubsub inter-gadget eventing model. A gadget may publish events over any number of arbitrary channels (defined by simple string names) in JavaScript. On the receiving end, a gadget may subscribe to receive events over any number of channels, again in JavaScript, and take appropriate actions based on the events.

```
function connSelected(element) {
var connId =element.id;
 var connName =element.lastChild.textContent;
 gadgets.pubsub.publish(channel, {id: connId, name: connName});
```

For a complete JavaScript reference to the supported eventing API, see:

http://docs.opensocial.org/display/OSD/Specs

16.1.6 Example: How to Consume an External OpenSocial Gadget

The example that follows is simplified for illustration. To support OpenSocial gadgets, you must first define an OpenSocial container as described in How to Configure OpenSocial Settings.

1. Using the Pagelet Producer Console, create a new Resource and choose **OpenSocial** as the type. Define URLs and policies as necessary. Save the new Resource.

2. Using the Pagelet Producer Console, create a new pagelet and enter the location of the gadget XML schema in the **Gadget URL** field.

   - To import the gadget name and any user preferences defined for the gadget, click the **Import Gadget Metadata** button. If any of the preferences are

editable, Pagelet Producer will create a preference editor using the imported preferences.

- To configure or customize the generated preference editor, go to the **Pagelet Parameters** page.

3. Go to the **Documentation** page to view sample code to access the pagelet and preference editor using either the JavaScript or REST API.

4. Save the new pagelet.

16.1.7 Example: How to Consume a Local OpenSocial Gadget

The example that follows is simplified for illustration. To support OpenSocial gadgets, you must first define an OpenSocial container as described in How to Configure OpenSocial Settings.

1. Using the Pagelet Producer Console, create a new Resource and choose **OpenSocial** as the type. Define URLs and policies as necessary. Save the new Resource.

2. Create or upload the gadget file.

> **Note:**
>
> If you choose to host OpenSocial gadget XML files, the files must be placed under an anonymous resource (with no security policy) or gadget functionality will not work correctly.

a. Select the **Files** section under the resource and click the **Create** icon in the toolbar.

b. On the **General** page, enter the relative path to the file in the **Name** field. Do not use a leading forward-slash ("/"). For example, "gadgets/activities.xml". Any path and name can be used; the path is a 'virtual' URL that Pagelet Producer will use to serve the file. The visual directory structure of the Files collection in the navigator is updated to match the path to the file.

c. Go to the **Content** page to upload or create the gadget file.

To upload a gadget file, enter the path to the file or click the Browse button to navigate to the file, then click the **Upload** button to upload the file to the Pagelet Producer server.

To create a gadget file, use the editor on the **Content** page to enter and edit content.

d. Click the **Save** icon in the navigator toolbar.

3. Create or upload any other files required by the gadget (JavaScript, images, etc.) in the **Files** section of the resource. Define paths for the files that match the paths in the gadget code. For example, if the gadget uses JavaScript files in a subfolder called "js", include that directory in the Name field when you upload the files, (for example, "gadgets/js/activities.js".)

4. Using the Pagelet Producer Console, create a new pagelet and enter the relative path to the gadget XML schema (created or uploaded in step 2) in the **Gadget**

URL field. For local files, this is the same path defined in the Name field for the file object (for example, "gadgets/activities.xml").

- To import the gadget name and any user preferences defined for the gadget, click the **Import Gadget Metadata** button. If any of the preferences are editable, Pagelet Producer will create a preference editor using the imported preferences.

5. Go to the **Documentation** page to view sample code to access the pagelet and preference editor using either the JavaScript or REST API.

6. Save the new pagelet.

16.2 Working with Pagelet Chrome for WSRP and Oracle PDK-Java Portlets

Pagelet Producer can be used to present WSRP and Oracle PDK-Java portlets for use in any web application. This section explains how to use pagelet chrome to modify markup at runtime in WSRP or Oracle PDK-Java portlets. For details on configuring the Pagelet Producer to connect to a WSRP or Oracle PDK-Java portlet producer, see Managing Pagelet Producer in *Administering Oracle WebCenter Portal*.

A pagelet chrome template is an HTML file that specifies:

- How portlet markup should be rendered stylistically

- How to display the portlet title

- How to render mode switching options

The default chrome template displays the portlet name and a dropdown menu that allows the user to switch into different modes. The dropdown menu is dynamically populated with all the standard modes supported by the underlying portlet. Templates use the following reserved tokens (see Table 16-3) to identify key portlet elements that Pagelet Producer then substitutes at runtime.

Table 16-3   Pagelet Chrome Template Tokens

Token	Description
`$$PORTLET TITLE$$`	Portlet title
`$$REPEAT MENU ITEM$$`	Used to indicate the beginning of a repeating section for navigational items
`$$END REPEAT MENU ITEM$$`	Used to indicate the end of a repeating section for navigational items
`$$MENU ITEM URL$$`	Navigation URL (to switch portlet modes or window states)
`$$MENU ITEM$$`	Display name of navigational item (for example, Customize)

Table 16-3   (Cont.) Pagelet Chrome Template Tokens

Token	Description
`$$TOKEN$$`	Unique identifier for pagelet instance on the page
`$$PORTLET CONTENT$$`	Portlet content
`pt://images`	Tag used to indicate the imageserver URL

The example below is a very simple pagelet chrome template:

```
<script type="text/javascript">
function goto(url)
{
 document.location =url;
 return false;
}
</script>
<div style="border: 1px solid">
<span><b><!-- $$PORTLET TITLE$$ --></b></span>
<span style="align: right">
 Switch Mode:
 <select size="1" name="mode">
 <!-- $$REPEAT MENU ITEM$$ -->
 <option onclick="goto('$$MENU ITEM URL$$')"><!-- $$MENU ITEM$$ --></option>
 <!-- $$END REPEAT MENU ITEM$$ -->
 </select>

</span>
</div>
<!-- $$PORTLET CONTENT$$ -->
```

> **Note:**
>
> The pagelet chrome template file must be hosted on the classpath of the Pagelet Producer web application.
>
> If you configured Pagelet Producer to use an external image server, copy the files from `ensemblestatic.war/imageserver/yahoo` to your image server to properly render the default chrome template.

To implement the chrome template, add it as a parameter to the pagelet inject URL (REST or JavaScript). For details on pagelet inject URLs, see Adding a Pagelet to a Web Page. For example:

- REST: `/inject/v2/pagelet/pagelet_lib/pagelet_name?chrome=mychrome.html`

- JavaScript: `injectpagelet(library, name, iframe_options, payload, params, context_id, element_id, is_in_community, chrome)`

The value of the chrome parameter can be the name of the file containing the chrome template or the special reserved value "none", which suppresses all chrome and sends back portlet markup only. If the chrome parameter is omitted, the default chrome is returned with the portlet markup. The default chrome template uses YUI menu control to display a gradient title bar and a DHTML dropdown menu for switching modes. (When ADF content is detected, a different chrome template is used by default. This template can be overridden with a custom template or with the standard default template by setting `chrome=chrometemplate.html`.)

17

Creating Custom Pagelets

This chapter provides detailed information about building pagelets using Pagelet Producer, including how to use the Adaptive Pagelet Scripting Framework, configure security settings for pagelets and resources to manage authentication for users, use custom web injectors and parsers to modify pagelet functionality at runtime, and access advanced logging traces for debugging.

This chapter includes the following sections:

- Using the Adaptive Pagelet Scripting Framework
- Modifying Pagelet Functionality at Runtime
- Debugging Pagelets

17.1 Using the Adaptive Pagelet Scripting Framework

The *Adaptive Pagelet Scripting Framework* is a client-side JavaScript library that provides services to CSP pagelets and proxied pages. This section explains how to use the scripting framework to implement dynamic functionality in pagelets.

> **Note:**
>
> CSP (and the Adaptive Pagelet Scripting Framework) only applies to WCI and its predecessor, Plumtree Portal. WebCenter Portal does not support CSP.

This section includes the following topics:

- How to Handle Structured HTTP Responses
- How to Use Event Notification
- How to Use In-Place Refresh
- How to Use Session Preferences

For a full list of classes and methods, see the JSPortlet API documentation. For additional information about adaptive pagelets, see What You May Need to Know About Adaptive Pagelet Development.

17.1.1 How to Handle Structured HTTP Responses

This section describes how the adaptive pagelet scripting framework can be used as a client-side response handler for structured HTTP, typically encoded as XML.

In many cases it can be expensive and inefficient to send large amounts of HTML back in response to some HTTP request, if only a small part of the user interface needs to be

changed. This is especially true with more complex user interfaces. In these cases, the response can be encoded in XML. The client-side response handler can then parse the XML, and update the user interface (or perform some other action) based on that response. Use the Structured Response design pattern to redraw a small area of the user interface after making an HTTP request, or to access a simple HTTP/URI type web service from a pagelet. The example code below (structuredresponse_portlet.html) accesses an RSS feed from a selection of news sites.

```html
<!-- jsxml includes -->
<a id="imgServerHref" href="/images/plumtree" style="display:none"></a>
<script type="text/javascript"
src="/images/plumtree/common/private/js/PTLoader.js"></script>
<script type="text/javascript">
var oImgServer =new Object();
oImgServer.location =document.getElementById('imgServerHref').href;
var imageServerURL =document.getElementById('imgServerHref').href;
var imageServerConnectionURL =oImgServer.location;
new PTLoader(imageServerURL, imageServerConnectionURL).include('jsxml','en');
</script>

<!-- jscontrols includes -->
<link rel="stylesheet" type="text/css" href="/portal-remote-server/js/jscontrols/
styles/css/PTMenu.css"/>
<link rel="stylesheet" type="text/css" href="/portal-remote-server/js/jscontrols/
styles/css/PTRichTextEditor.css"/>
<script type="text/javascript" src="/portal-remote-server/js/jscontrols/strings/
PTControls-en.js"></script>
<script type="text/javascript" src="/portal-remote-server/js/jscontrols/
PTControls.js"></script>

<!-- Inline JS helper functions -->
-->

<script defer type="text/javascript" id="structured-response-portlet-A-script">
// Function that gets the RSS XML feed found at the specified url
getRSSFeed =function(url)
 {
//First clear out any existing rows in the table
 channelTable.clearRows();

 //Force the transformer to fix up the url
 var oURL =new Object();
 oURL.location =url;

 //Do the http get
 var get =new PTHTTPGETRequest(oURL.location, handleRSSResponse);
 get.invoke();
 }

//Function that handles the RSS XML response and updates the table based on the RSS
items
handleRSSResponse =function(response)
 {
//Get the rss xml
 var xml =response.responseText;
 if (!xml ||xml.indexOf('<?xml') ==-1) {return; }

//Parse into a dom, and get the channel node
 var xmlDOM =new PTXMLParser(xml);
 var rssNode =xmlDOM.selectSingleNode('rss');
 var channelNode =rssNode.selectSingleNode('channel');
```

```
 //Get the channel title and set the status bar text in the table
 var channelTitle =channelNode.selectSingleNode('title').getNodeValue();
 channelTable.statusBarText ='<b>Loaded Channel</b>: ' +channelTitle;

 //Get channel item nodes
 var itemNodes =channelNode.selectNodes('item');

//Build table rows
 channelTable.rows =new Array();
 for (var i=0; i<itemNodes.length; i++)
 {
var itemNode =itemNodes[i];

 //Get channel item properties
 var itemTitle =itemNode.selectSingleNode('title').getNodeValue();
 var itemLink =itemNode.selectSingleNode('link').getNodeValue();
 var itemDescription =itemNode.selectSingleNode('description').getNodeValue();
 if (itemNode.selectSingleNode('author'))
 var itemAuthor =itemNode.selectSingleNode('author').getNodeValue();
 if (itemNode.selectSingleNode('category'))
 var itemCategory =itemNode.selectSingleNode('category').getNodeValue();
 if (itemNode.selectSingleNode('pubDate'))
 var itemPubDate =itemNode.selectSingleNode('pubDate').getNodeValue();

 //Create a row and add it to the table
 var row =new PTRow();
 row.parent =channelTable;
 row.id =i;
 row.uid =i;
 row.previewText =itemDescription;
 row.link =itemLink;
 row.columnValues[0] =new PTTextColumnValue(itemTitle);
 row.columnValues[1] =new PTTextColumnValue(itemCategory);
 row.columnValues[2] =new PTTextColumnValue(itemAuthor);
 row.columnValues[3] =new PTTextColumnValue(itemPubDate);
 channelTable.rows[channelTable.rows.length] =row;
 }

//Redraw the table
 channelTable.draw();
 }
</script>

<b>Select RSS Feed:</b>
<a href="#" onclick="getRSSFeed('http://www.wired.com/news/feeds/
rss2/0,2610,,00.xml'); return false;">Wired News</a>
<a href="#" onclick="getRSSFeed('http://news.com.com/2547-1_3-0-5.xml'); return
false;">CNET News.com</a>
<a href="#" onclick="getRSSFeed('http://partners.userland.com/nytRss/
nytHomepage.xml'); return false;">NY Times</a>
<br><br>

<!-- Set up a table control to display channel items -->
<div id="channelTableContainer"></div>
<script defer type="text/javascript">
 var channelTable =new PTTableControl();
 channelTable.locale ='en_US';
 channelTable.objName ='channelTable';
 channelTable.container ='channelTableContainer';
 channelTable.baseURL ='/imageserver/plumtree/common/private/portal-remote-server/js/
```

```
                    jscontrols/1/';
                     channelTable.statusBarText ='No RSS Feed Selected';
                     channelTable.rowDetailAction =new PTJavaScriptAction('window.open(\'${ROW.link}
                    \');');
                     channelTable.columns[0] =new PTColumn();
                     channelTable.columns[0].name ='Title';
                     channelTable.columns[0].width ='40%';
                     channelTable.columns[1] =new PTColumn();
                     channelTable.columns[1].name ='Category';
                     channelTable.columns[1].width ='20%';
                     channelTable.columns[2] =new PTColumn();
                     channelTable.columns[2].name ='Author';
                     channelTable.columns[2].width ='20%';
                     channelTable.columns[3] =new PTColumn();
                     channelTable.columns[3].name ='Publication Date';
                     channelTable.columns[3].width ='20%';
                     channelTable.areColumnsResizable =true;
                     channelTable.clientSortEnabled =true;
                     channelTable.scrollHeight =250;
                     channelTable.init();
                     channelTable.draw();
                    </script>
                    </div>
```

17.1.2 How to Use Event Notification

This section describes how the adaptive pagelet scripting framework allows pagelets to respond to both page-level events and custom events raised by other pagelets.

The `registerForWindowEvent` and `registerOnceForWindowEvent` methods in the scripting framework provide pagelets with access to page-level events. For a complete list, see How to Use Page-Level Events with the Scripting Framework. To register for notification of these events, pass in the name of the event and the name of the method that should be called when it occurs. When a page-level event is raised, the JavaScript event object is passed to the event handler as an argument. The scripting framework also allows pagelets to raise and respond to custom events using `raiseEvent` and `registerForEvent`. The Broadcast-Listener design pattern illustrates an important example of using notification services with session preferences. Users can select an item or perform some other action in a "broadcast" pagelet, which causes the content in other related "listener" pagelets to be redrawn. In the following example, the broadcast pagelet displays a form that allows you to enter a number in a text box.

Figure 17-1   Broadcast Portlet

When the user enters a number in the text box, the values in the listener pagelets change. The first listener pagelet displays the square root of the number entered in the broadcast pagelet.

Figure 17-2   Listener -1 Portlet
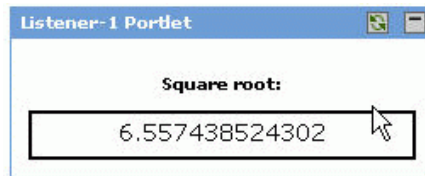
The second listener pagelet displays the cube root of the number entered in the broadcast pagelet.

Figure 17-3   Listener -2 Portlet
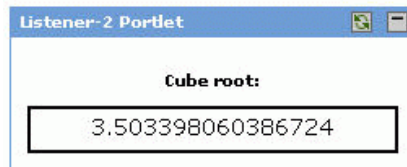
The following steps summarize how the pagelets work:

- On load, each listener pagelet calls its own instance method (`registerForEvent`) to register for events of type 'onBroadcastUpdate'.

- On each onkeyup event that occurs in the "Enter number" text box, the broadcast pagelet sets a session preference to the value entered in the text box, and calls its own instance method (`raiseEvent`) to raise an event of type 'onBroadcastUpdate'.

- When the onBroadcastUpdate event is raised or the page is reloaded, each listener pagelet retrieves the session preference set by the broadcast pagelet and computes a new value to display based on the value of the preference.

Broadcast Pagelet

```
<div style="padding:10px;" align="center">
<p><b>Enter number:</b>
 <input type="text" style="font-size:22px;font-weight:bold;text-align:center;"
id="broadcast_prefName" value="4" size="7"
onkeyup="broadcast_setPrefs(this.value)"></p>
<br>
</div>

<script type="text/javascript">
function broadcast_setPrefs(val)
{
 var prefName ='broadcastNumber';
 var prefValue =val;
 PTPortlet.setSessionPref(prefName,prefValue);
 var broadcastPortlet =PTPortlet.getPortletByGUID('{D9DFF3F4-
EAE7-5478-0F4C-2DBD94444000}');

 if (!broadcastPortlet)
 {
broadcast_debug('Could not locate PTPortlet object which corresponds to <b>Broadcast
Portlet</b> on page.');
 return;
 }
broadcast_debug('<b>Broadcast Portlet</b> raising onBroadcastUpdate event.');
 broadcastPortlet.raiseEvent('onBroadcastUpdate',false);
}
```

```
function broadcast_debug(str)
{
 if (window.PTDebugUtil)
{
PTDebugUtil.debug(str);
}
}
</script>
```

Listener Pagelet #1

```
<div style="padding:10px;" align="center">
<p><b>Square root:</b>
<div style="height:21px;border:2px solid black;padding:2px;overflow:visible;font-
size:14px;"id="listener1-swatch">
</div>
</div>

<script>
function listener1_update()
{
 var broadcastNumber =parseFloat(PTPortlet.getSessionPref('broadcastNumber'));
 if (isNaN(broadcastNumber))
 {
listener1_error('<b>Listener-1 Portlet</b> cannot parse number from session pref
broadcastNumber');
 return;
 }

listener1_debug('<b>Listener-1 Portlet</b> computing square root of '
+broadcastNumber);
 var swatch =document.getElementById('listener1-swatch');
 swatch.innerHTML =Math.sqrt(broadcastNumber);
}

function listener1_debug(str)
{
 if (window.PTDebugUtil)
{
PTDebugUtil.debug(str);
}
}

function listener1_error(str)
{
 if (window.PTDebugUtil)
{
PTDebugUtil.error(str);
}
}

function listener1_getPortlet()
{
 var portletGUID ='{D9DFF3F4-EAE7-5478-0F4C-2DBDB4F4A000}';
 var listener1Portlet =PTPortlet.getPortletByGUID(portletGUID);
 return listener1Portlet;
}

var listener1Portlet =listener1_getPortlet();
if (listener1Portlet)
{
```

```
 listener1Portlet.registerForEvent('onBroadcastUpdate','listener1_update');
 listener1_debug('<b>Listener-1 Portlet</b> registered refreshOnEvent for event
onBroadcastUpdate');
 listener1Portlet.registerForEvent('onload','listener1_update');
}

</script>
```

Listener Pagelet #2

```
<div style="padding:10px;" align="center">
<p><b>Cube root:</b>
<div style="height:21px;border:2px solid black;padding:2px;overflow:visible;font-
size:14px;"id="listener2-swatch">
</div>
</div>

<script>
var listener2_oneThird =(1/3);
function listener2_update()
{
 var broadcastNumber =parseFloat(PTPortlet.getSessionPref('broadcastNumber'));
 if (isNaN(broadcastNumber))
 {
listener2_error('<b>Listener-2 Portlet</b> cannot parse number from session pref
broadcastNumber');
 return;
 }

listener2_debug('<b>Listener-2 Portlet</b> computing square root of '
+broadcastNumber);
 var swatch =document.getElementById('listener2-swatch');
 swatch.innerHTML =Math.pow(broadcastNumber,listener2_oneThird);
}

function listener2_debug(str)
{
 if (window.PTDebugUtil)
{
PTDebugUtil.debug(str);
}
}

function listener2_error(str)
{
 if (window.PTDebugUtil)
{
PTDebugUtil.error(str);
}
}

function listener2_getPortlet()
{
 var portletGUID ='{D9DFF3F4-EAE7-5478-0F4C-2DBDCA1C7000}';
 var listener2Portlet =PTPortlet.getPortletByGUID(portletGUID);
 return listener2Portlet;
}

var listener2Portlet =listener2_getPortlet();
if (listener2Portlet)
{
```

```
listener2Portlet.registerForEvent('onBroadcastUpdate','listener2_update');
listener2_debug('<b>Listener-2 Portlet</b> registered refreshOnEvent for event
onBroadcastUpdate');
listener2Portlet.registerForEvent('onload','listener2_update');
}

</script>
```

17.1.2.1 How to Use Page-Level Events with the Scripting Framework

The scripting framework automatically has access to the following page-level events.

Table 17-1   Page-Level Events

Event	Triggered:
onload	immediately after the browser loads the page
onbeforeunload	prior to a page being unloaded (browser window closes or navigates to different location)
onunload	immediately before the page is unloaded (browser window closes or navigates to different location)
onactivate	the page is set as the active element (receives focus)
onbeforeactivate	immediately before the page is set as the active element (receives focus)
ondeactivate	when the active element is changed from the current page to another page in the parent document
onfocus	when the page receives focus
onblur	when the page loses focus
oncontrolselect	when the user is about to make a control selection of the page
onresize	when the size of the page is about to change
onresizestart	when the user begins to change the dimensions of the page in a control selection
onresizeend	when the user finishes changing the dimensions of the page in a control selection
onhelp	when the user presses the F1 key while the browser is the active window
onerror	when an error occurs during page loading
onafterprint	immediately after an associated document prints or previews for printing

17.1.3 How to Use In-Place Refresh

This section describes how pagelets can reload their internal content without refreshing the page using the scripting framework to implement in-place refresh.

Many pagelets display data that is time sensitive. In some cases, users should be able to navigate across links within a pagelet without changing or refreshing the rest of the

page. You can refresh pagelet content on command, associate the refresh action with an event (refreshOnEvent), or program the pagelet to refresh at a set interval (setRefreshInterval). The scripting framework also contains methods for expanding and collapsing pagelets. In the simplified example below, the refresh pagelet displays a "Refresh Portlet" button. Clicking the button updates the date and time displayed in the pagelet.

Figure 17-4   Refresh Portlet

The in-place refresh is executed by calling the refresh() method on the pagelet object instance. You can also set a new URL to be displayed within the pagelet upon refresh. (The title bar cannot be altered on refresh.)

```
<div style="padding:10px;" align="center">
<p><button onclick="refresh_portlet()">Refresh Portlet</button></p>
<p><b>Current time is:</b><br> <span id="refreshTimeSpan"></span></p>
</div>

<script type="text/javascript">
function refresh_portlet()
{
var refreshPortlet =PTPortlet.getPortletByID($PORTLET_ID$);
if (!refreshPortlet)
 {
refresh_debug('Could not locate PTPortlet object which corresponds to <b>Refresh
Portlet</b> on page.');
 return;
 }
refresh_debug('<b>Refresh Portlet</b> calling refresh() method.');
refreshPortlet.refresh();
}

function refresh_debug(str)
{
if (window.PTDebugUtil)
{
PTDebugUtil.debug(str);
}
}

var t =new Date();
document.getElementById('refreshTimeSpan').innerHTML =t;
</script>
```

17.1.4 How to Use Session Preferences

This section describes how browser-level variables can be stored and shared among pagelets, even if they are not on the same page. For example, a value entered by the user in one pagelet can be retrieved by another. The scripting framework acts as an intermediary, allowing all pagelets access to all values stored in a common session using session preferences.

Pagelets can use preferences to communicate with each other, but accessing preferences usually requires a round trip to a database. Session preferences provide a way to store and share settings in the user's session within the client browser. The Master-Detail design pattern illustrates the most basic usage of session preferences. This design pattern splits control and display between two pagelets. For example, the "master" pagelet could summarize data in list form, and the "detail" pagelet could display details on each data item in response to user selection. In the example below, the master pagelet displays a form that allows you to enter a color code in a text box.

Figure 17-5   Color Code

When the user enters a color code in the text box, the color in the detail pagelet changes.

Figure 17-6   Color Swatch
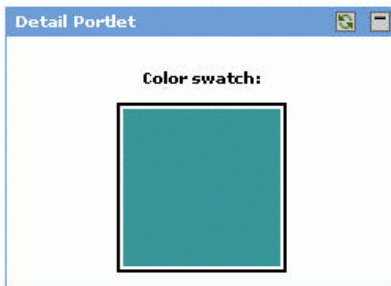
For each `onkeyup` event that occurs in the "Enter color" text box in the master pagelet, the following steps are executed:

1. The master pagelet sets the session preference using the current value of the text box.

2. The master pagelet calls an update method on the detail pagelet.

3. The detail pagelet retrieves the session preference to get the color value.

4. The detail pagelet redraws its color swatch area to reflect the new color value.

Note:

Shared session preferences must be specified by name on the Preferences page for the pagelet in the Pagelet Producer Console or they will not be sent to the pagelet.

The adaptive pagelet scripting framework provides an easy way to detach the relationship between pagelets and use a common event interface for communication.

Note:

The example below is oversimplified; the master pagelet makes a direct call to a JavaScript method of the detail pagelet. Unless the master pagelet takes extra measures to ensure that the detail pagelet is actually present on the same page, calls from master to detail could generate errors.

Master Pagelet

```
<div style="padding:10px;" align="center">
<p><b>Enter color:</b>  
<input type="text" style="font-size:22px;font-weight:bold;text-align:center;"
id="master_prefName"
value="#FFFFFF" size="8" onkeyup="master_setPrefs(this.value)"></p><br>
</div>

<script type="text/javascript">
function master_setPrefs(val)
{
var prefName ='masterColor';
var prefValue =val;
PTPortlet.setSessionPref(prefName,prefValue);

master_debug('<b>Master Portlet</b> called PTPortlet.setSessionPref(\'masterColor\',
\'' +prefValue +'\').');

if (window.detail_update)
 {
master_debug('<b>Master Portlet</b> calling detail_update().');
 detail_update();
 }
else
 {
master_debug('Could not locate portlet <b>Detail Portlet</b> on page.');
 }
}
function master_debug(str)
{
if (window.PTDebugUtil)
{
PTDebugUtil.debug(str);
}
}
</script>
```

Detail Pagelet

```
<div style="padding:10px;" align="center">
<p><b>Color swatch</b>  
<div style="width:100px;height:100px;border:2px solid black;padding:2px;"id="detail-
swatch"></div>
<script>
function detail_update()
{
var color =PTPortlet.getSessionPref('masterColor');
detail_debug('<b>Detail Portlet</b> received value="' +color +'" for
PTPortlet.getSessionPref(\'masterColor\')');

var swatch =document.getElementById('detail-swatch');
if (swatch)
```

```
 {
swatch.innerHTML ='<div style="background-color:' +color +';width:100%;height:
100%;"></div>';
 }
else
 {
detail_debug('<b>Detail Portlet</b> cannot find \'detail-swatch\' DIV element.');
 }
}

function detail_debug(str)
{
if (window.PTDebugUtil)
{
PTDebugUtil.debug(str);
}
}
</script>
```

17.1.5 What You May Need to Know About Adaptive Pagelet Development

These tips apply to most pagelets that use the adaptive pagelet scripting framework.

- **Use unique names for all forms and functions.** Use the GUID of a pagelet to form unique names and values to avoid name collisions with other code on the page.

- **Proxy all URLs.** You cannot make a request to a URL with a host/port that is different from that of the calling page. All URLs requested through JavaScript must be proxied.

- **If you are using the adaptive pagelet scripting framework, check for support.** It is good practice to include code that determines whether or not the component is present. Ideally, your pagelet should be able to handle either situation. The simplest solution is to precede your code with an If statement that alerts the user if the scripting framework is not supported.

```
<script>
if (PTPortlet ==null)
{
if (document.PCC ==null)
{
alert("This pagelet only works in portals that support the JSPortlet API .
The pagelet will be displayed with severely reduced
 functionality. Contact your Administrator.");
 }
}
else
{
[scripting code here]
 }
</script>
```

- **Close all popup windows opened by a pagelet when the window closes.** The scripting framework can be used to close popup windows using the onunload event.

- **Add all required JavaScript to the page in advance.** Browsers might not process script blocks/includes added to the page through the innerHTML property.

  - Microsoft Internet Explorer: Add the defer attribute to the script tag.

– Netscape: Use RegExp to parse the response and look for the script, then eval it.

- **JavaScript HTTP and proxied HTTP must use the same authentication credentials.** JavaScript brokered HTTP requests send the same authentication token (cookie) as when you make a normal gatewayed HTTP request.

17.2 Modifying Pagelet Functionality at Runtime

This section describes how to modify pagelet functionality at runtime using custom injectors and parsers.

This section includes the following topics:

- How to Use Web Injectors
- How to Use Custom Parsers

17.2.1 How to Use Web Injectors

A web injector inserts content into a specified location in the proxied resource page. The content may be any text, including HTML, CSS, JavaScript, and pagelet declarations. An empty injector may also be used to remove unwanted content from a page. Injectors cannot be created for OpenSocial resources. While injecting simple HTML content has a limited use case, you can also inject JavaScript that directly modifies the pagelet's HTML markup. For details on creating a web injector, see Creating Web Injectors.

17.2.2 How to Use Custom Parsers

Custom parsers allow you to supplement or change built-in logic for parsing content and finding URLs. When the built-in parsers fail to identify URLs or identify sections that must not be rewritten as URLs, custom parsers can be used to change the default behavior. Parsers cannot be created for WSRP or Oracle PDK-Java portlet producers or for OpenSocial gadget producers. For details on creating a custom parser, see Creating Custom Parsers.

17.3 Debugging Pagelets

This section describes how to use advanced logging traces for debugging pagelets. Logging is configured in the Settings section of the Pagelet Producer Console, where you can define different levels of logging for each Pagelet Producer component. For more information, see Configuring Pagelet Producer Settings.

This section includes the following topics:

- How to View HTTP Requests and Responses
- How to View Transformation Content

17.3.1 How to View HTTP Requests and Responses

To view the HTTP requests and responses received and sent by Pagelet Producer, set the **HTTP** component on the Logging Settings page to **Finest**. The traces in "HTTP Request Received By Pagelet Producer," "HTTP Response Sent By Pagelet Producer," "HTTP Request Sent By Pagelet Producer," and "HTTP Response Received By Pagelet Producer" were captured from a test environment.

HTTP Request Received By Pagelet Producer

```
URL: http://example.com:7001/pagelets/bidwiki/includes/js/ajax.js
METHOD: GET
SESSION ID: GdYGNJzMhxy1CJBMVTX8xTNq32GmLXYNY9VqFBcdprFnhcyQtzdp!1377086614!
1305769149498
HEADERS:
Host: example.com
 Connection: keep-alive
 Referer: http://example.com:7001/pagelets/bidwiki/dashboard.action
 User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/534.16
(KHTML, like Gecko) Chrome/10.0.648.205 Safari/534.16
 Accept: */*
Accept-Encoding: gzip,deflate,sdch
 Accept-Language: en-US,en;q=0.8,ru;q=0.6,it;q=0.4
 Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
 If-None-Match: W/"736-1124953206000"
 If-Modified-Since: Thu, 25 Aug 2012 07:00:06 GMT
COOKIES:
JSESSIONID: GdYGNJzMhxy1CJBMVTX8xTNq32GmLXYNY9VqFBcdprFnhcyQtzdp!1377086614
```

HTTP Response Sent By Pagelet Producer

```
URL: http://example.com:7001/pagelets/bidwiki/styles/main-action.css
Response Code: 200
Reason:
HEADERS:
Date: Thu, 19 May 2011 01:39:14 GMT
 Content-Type: text/css;charset=UTF-8
 Server: Apache-Coyote/1.1
BODY:
.sidebar {
/*background-image: url(http://example.com:7001/pagelets/bidwiki/download/ resources/
leftnav_bg.jpg);*/
 /*background-repeat: repeat-y;*/
 background-color: #F0F0F0;
 /*border-bottom:1px solid #F0F0F0;*/
} ...
```

HTTP Request Sent By Pagelet Producer

```
URL: http://xmlns.oracle.com/includes/js/ajax.js
METHOD: GET
HEADERS:
CSP-Ensemble-REST-API: http://example.com:7001/pagelets
 X-Client-IP: example.com
 User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/534.16
(KHTML, like Gecko) Chrome/10.0.648.205 Safari/534.16
 CSP-Session-Username: weblogic
 Accept-Language: en-US,en;q=0.8,ru;q=0.6,it;q=0.4
 CSP-Gateway-Type: Proxy
 PT-Proxy-Passes: 1
 If-Modified-Since: Thu, 25 Aug 2012 07:00:06 GMT
 CSP-Protocol-Version: 1.4
 Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
 Accept-Encoding: gzip,deflate,sdch
 Referer: http://example.com:7001/pagelets/bidwiki/dashboard.action
 If-None-Match: W/"736-1124953206000"
 Accept: */*
CSP-Aggregation-Mode: Single
 PT-Proxy-instance0: {DECBB085-D891-72CF-2B75-005E7FE20000}
 CSP-Gateway-Specific-Config: PT-User-Name=weblogic,PT-Guest-User=0,...
```

HTTP Response Received By Pagelet Producer

```
Original URI: http://xmlns.oracle.com/styles/main-action.css
Effective URI: http://xmlns.oracle.com/styles/main-action.css
Status Code: 200
Reason: OK
Version: HTTP/1.1
HEADERS:
Content-Type: text/css;charset=UTF-8
 Content-Length: 29178
 Server: Apache-Coyote/1.1
 Date: Thu, 19 May 2011 01:39:14 GMT
TRAILERS:
BODY:
body, p, td, table, tr, .bodytext, .stepfield {
font-family: Verdana, arial, sans-serif;
 font-size: 11px;
 line-height: 16px;
 color: #000000;
 font-weight: normal;
}
```

17.3.2 How to View Transformation Content

To view the content proxied by Pagelet Producer before and after transformation, set the **Transform** component on the Logging Settings page to **Finest**. The purpose of these traces is to log response content at different stages of transformation, allowing you to compare them and view the result of different transformers. The example traces in the following examples were captured from a test environment.

Example: Untransformed Markup

```
Original request: URL: http://example.com:7001/pagelets/bidwiki/styles/main-
action.css
METHOD: GET
SESSION ID: GdYGNJzMhxy1CJBMVTX8xTNq32GmLXYNY9VqFBcdprFnhcyQtzdp!1377086614!
1305769149498
HEADERS:
Host: example.com:7001
 Connection: keep-alive
 Referer: http://example.com:7001/pagelets/bidwiki/dashboard.action
 User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/534.16
(KHTML, like Gecko) Chrome/10.0.648.205 Safari/534.16
 Accept: text/css,*/*;q=0.1
 Accept-Encoding: gzip,deflate,sdch
 Accept-Language: en-US,en;q=0.8,ru;q=0.6,it;q=0.4
 Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
COOKIES:
JSESSIONID: GdYGNJzMhxy1CJBMVTX8xTNq32GmLXYNY9VqFBcdprFnhcyQtzdp!1377086614

Untransformed content:
body, p, td, table, tr, .bodytext, .stepfield {
font-family: Verdana, arial, sans-serif;
 font-size: 11px;
 line-height: 16px;
 color: #000000;
 font-weight: normal;
}
```

Example: Transformed Markup (Transformed by Transformer Class)

```
Transformed by: class com.plumtree.server.impl.portlet.transformers.CSSTurboParser
 Original request: URL: http://example.com:7001/pagelets/bidwiki/styles/main-
action.css
METHOD: GET
SESSION ID: GdYGNJzMhxy1CJBMVTX8xTNq32GmLXYNY9VqFBcdprFnhcyQtzdp!1377086614!
1305769149498
HEADERS:
Host: 10.148.118.211:7001
 Connection: keep-alive
 Referer: http://example.com:7001/pagelets/bidwiki/dashboard.action
 User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/534.16
(KHTML, like Gecko) Chrome/10.0.648.205 Safari/534.16
 Accept: text/css,*/*;q=0.1
 Accept-Encoding: gzip,deflate,sdch
 Accept-Language: en-US,en;q=0.8,ru;q=0.6,it;q=0.4
 Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
COOKIES:
JSESSIONID: GdYGNJzMhxy1CJBMVTX8xTNq32GmLXYNY9VqFBcdprFnhcyQtzdp!1377086614

Transformed content:
body, p, td, table, tr, .bodytext, .stepfield {
font-family: Verdana, arial, sans-serif;
 font-size: 11px;
 line-height: 16px;
 color: #000000;
 font-weight: normal;
}
```

18

Using Pagelets in Web Applications

This chapter describes how to add pagelets to a web or WebCenter Portal page.

Note:

Before you can add a pagelet to any web application, you must deploy and configure the resource and pagelet in Pagelet Producer as described in Deploying Portals, Templates, Assets, and Extensions in *Administering Oracle WebCenter Portal*.

This chapter includes the following sections:

- Adding a Pagelet to a Web Page
- Adding a Pagelet to a Portal Page

18.1 Adding a Pagelet to a Web Page

Once you have deployed a pagelet, you can:

- Insert pagelets into non-proxied pages using a simple JavaScript function.
- Retrieve information about resources and pagelets from Pagelet Producer, and inject pagelets into proxied and non-proxied pages using Pagelet Producer's REST APIs.
- Automatically resize the IFrame that encapsulates pagelet content using the pagelet inject function.

This section includes the following topics:

- How to Insert Pagelets Using Javascript
- How to Access Pagelets Using REST
- How to Use Automatic Resizing with IFrames

18.1.1 How to Insert Pagelets Using Javascript

You can insert pagelets into non-proxied pages using a simple JavaScript function.

To activate this feature, add the following HTML snippet in the <HEAD> section of the page.

```
<script type="text/javascript" src="http://proxy:port/pagelets/inject/v2/csapi">
</script>
```

This script injects all CSAPI and pagelet inject functions into the page to display the pagelet. One of the sections injected is the following function:

```
function injectpagelet(library, name, iframe_options, params, context_id,
element_id, is_in_community, chrome, forward_params)
{
 ...
}
```

This function injects a pagelet as a widget into the parent page. The method interface uses the following parameters:

- **library**: Required. A string representing the library name of the pagelet to inject. Accepts Unicode letters and numbers only; no spaces.

- **name**: Required. A string representing the name of the pagelet to inject. Accepts Unicode letters and numbers only; no spaces.

- **iframe_options**: Specifies whether to use IFRAME around the pagelet content. Sample IFRAME options: `iframe width=100% height=auto frameborder=0`. If omitted or left blank, the pagelet content is rendered inline.

- **params**: The pagelet parameters in query string format. For example, `'param1=value1&param2=value2&param3=value3'`.

- **context_id**: The external identifier of the pagelet instance, used to scope preferences on the Pagelet Producer server. Must be an integer.

- **element_id**: The HTML element ID in which the pagelet content is injected. If omitted or left blank, the pagelet content is injected using `document.write()` when the `injectpagelet` call is evaluated.

- **is_in_community**: Specifies whether this pagelet is on a community or group page. If the value is set to `true`, it sends the `context_id` in the community ID header to the pagelet. Defaults to `false`.

- **chrome**: Specifies the name of the chrome template to use for WSRP/JPDK pagelets. To suppress chrome, use a value of `none`.

- **forward_params**: Specifies whether to forward query string arguments from the consuming page to the back-end server. To suppress this functionality, use a value of `false`.

Note:

These arguments are positional; they must be provided in the given order. If you do not want to specify a particular argument, but do want to specify an argument that follows it, you must pass in an empty value (") for the former. All arguments are optional except for library and name.

The script also creates a new `<div>` with a unique name that includes a reference to the `injectpagelet` function. Several examples are shown below:

```
<div>
 <script type="text/javascript">
 injectpagelet('library', 'name');
 </script>
</div>
```

```
<div>
 <script type="text/javascript">
 injectpagelet('library', 'name', 'iframe', 'payload',
'param1=value1&param2=value2&param3=value3');
 </script>
</div>

<div>
 <script type="text/javascript">
 injectpagelet('library', 'name', 'iframe width=100% height=200', 'payload');
 </script>
</div>
```

18.1.1.1 How to Add a Preference Editor Using Javascript

The `injecteditor` function lets you add preference editors that enable users to set personal and shared preferences for pagelets that support this capability. This functionality is analogous to personalization and customization functionality in WebCenter Portal.

```
injecteditor(library, name, type, iframe_options, context_id, element_id,
is_in_community, chrome)
```

where:

- **library**: Required. The library name of the pagelet to inject. Accepts Unicode letters and numbers only; no spaces.

- **name**: Required. The name of the pagelet to inject. Accepts Unicode letters and numbers only; no spaces.

- **type**: The editor type. This argument supports these values: 'admin', 'pagelet', 'community'. In case of the 'community' argument, `context_id` is sent to pagelet in the community ID CSP header.

- **iframe_options**: Specifies whether to use IFRAME around the pagelet editor content. Sample IFRAME options: `iframe width=100% height=auto frameborder=0`. If omitted or left blank, the editor content is rendered inline.

- **context_id**: The external identifier of the pagelet instance, used to scope preferences on the Pagelet Producer server. Must be an integer.

- **element_id**: The HTML element ID in which the pagelet content is injected. If omitted or left blank, the pagelet content is injected using `document.write()` when the `injecteditor` call is evaluated.

- **is_in_community**: Specifies whether this pagelet is on a community or group page. If the value is set to `true`, it sends the `context_id` in the community ID header to the pagelet. Defaults to `false`.

- **chrome**: Specifies the name of the chrome template to use for WSRP/JPDK pagelets. To suppress chrome, use a value of `none`.

> **Note:**
>
> These arguments are positional; they must be provided in the given order. If you do not want to specify a particular argument, but do want to specify an argument that follows it, you must pass in an empty value (") for the former. All arguments are optional except for library and name.

18.1.2 How to Access Pagelets Using REST

REST stands for Representational State Transfer and is a simple way of providing APIs over HTTP. The basic principles of REST are:

- API URLs point to the resource rather than a generic method endpoint

- Requests use standard HTTP verbs for simplified CRUD methods. This is a read-only API and allows GET requests only.

- Every request returns a full representation of the object retrieved (pagelet or resource)

Pagelet Producer REST APIs let you:

- Inject pagelets into non-proxied pages, allowing Pagelet Producer to act as a portlet provider for Oracle WebCenter Interaction, Oracle WebLogic Portal, or other third-party portals

- Allow remote Web services to retrieve information about resources and pagelets from Pagelet Producer

This section contains the following topics:

- What You May Need to Know About the Pagelet Inject API

- What You May Need to Know About Data Retrieval APIs

18.1.2.1 What You May Need to Know About the Pagelet Inject API

The pagelet inject URL can be used in portals to specify the location of a remote portlet (this is how a pagelet can be used as a portlet). The inject URL can also be used as the source attribute of an IFrame tag in any HTML page.

The URL must use the following format:

```
http://host:port/pagelets/inject/v2/pagelet/libraryname/
pageletname?content-type=html
```

where `libraryname` and `pageletname` refer to the library and pagelet configured in Pagelet Producer.

> **Note:**
>
> When using the pagelet inject API as the URL for a Portlet Web Service in Oracle WebCenter Interaction, you must switch "pagelet" to "portlet" in the URL. For example, the above URL would become:
>
> ```
> http://host:port/pagelets/inject/v2/portlet/libraryname/
> pageletname?content-type=html
> ```

The query string arguments to the above call define how the pagelet is to be returned. The following parameters are defined:

- **instanceid:** Optional. The instance ID of the pagelet, used to uniquely identify the pagelet on the page to facilitate inter-pagelet communication. Must be unique to the page.

- **context**: Optional. The external identifier of the pagelet instance, used for scoping preferences on the Pagelet Producer server. Must be an integer.

- **content-type**: The return type. Three types are supported:

    - **javascript**: Returns injectable code.

    - **html**: Returns the pagelet markup with its associated PTPortlet object.

    - **iframe**: Returns an IFrame that points back to the inject api, filling the IFrame with the pagelet content, instead of directly inline with the page. The IFrame can be styled by providing a set of query string parameters.

Parameter	Description	Default
ifwidth	Sets the width of the IFrame; can be specified in percent '%' or pixels 'px', for example: ifwidth=500px. Can be set to 'auto' to automatically resize the IFrame to fit the content within. For details, see How to Use Automatic Resizing with IFrames .	100%
ifheight	Sets the height of the IFrame; can be specified in percent '%' or pixels 'px', for example: ifheight=500px. Can be set to 'auto' to automatically resize the IFrame to fit the content within. For details, see How to Use Automatic Resizing with IFrames .	No default
ifborder	Sets the border of the IFrame.	'none'
ifalign	Sets the align rule within the IFrame, for example: ifalign=center.	No default
ifdesc	Sets the description of the IFrame.	No default
ifmarginheight	Sets the margin height; can be specified in percent '%' or pixels 'px', for example: ifmarginheight=500px.	No default
ifmarginwidth	Sets the margin width; can be specified in percent '%' or pixels 'px', for example: ifmarginwidth=500px.	No default
ifscrolling	Sets the scrollbars of the IFrame. Accepted values: yes/no/auto.	auto
ifstyle	Sets the CSS style of the IFrame	No default
ifclass	Sets the CSS class of the IFrame.	No default

- **csapi:** Optional. Sets whether the CSAPI will be included with the pagelet response (true or false). Including the CSAPI is optional, but the pagelet included in the response relies on the CSAPI libraries being present on the page where the pagelet is to be rendered. If csapi=false, then the CSAPI libraries must be included with the parent page (usually in the HEAD section).

- **onhtttperror:** Optional. When a pagelet request results in a 403, 404 or any other error code, Pagelet Producer can forward the error code and the error page itself to the browser for display to the user. The onhttperror parameter accepts the following values:

  - **comment** (default): Pagelet Producer will create an HTML comment in place of the failing pagelet (the failing pagelet will simply not be displayed).

  - **inline**: The pagelet error along with the server error page will be displayed inline where the pagelet would normally be shown on the page.

  - **fullpage**: The http error will consume the whole page. This mode is only available if Pagelet Producer controls the parent page.

- **inline-refresh:** Optional. Valid values are `true` or `false (default)`. If the `inline-refresh` option is set to `true`, Pagelet Producer does not use an iFrame to embed pagelet markup onto the page. When the user interacts with the pagelet, markup updates are done inline, hence the term 'inline-refresh'. To achieve this, Pagelet Producer re-writes the URLs in the pagelet markup to use a JavaScript method call instead. That is, when the user clicks on a link or submits a form request to the backend, it's done by JavaScript on the browser and then the response is dynamically injected.

  The `inline-refresh` option works well for relatively simple HTML markup. If the pagelet exposes a very rich UI, such as an ADF task flow, `inline-refresh` may not work well or may simply break. Consequently, the `inline-refresh` option is set to `false` by default (that is, inject pagelet markup in an iFrame).

The following example URL points to the linkspagelet in the samples library:

```
http://host:port/pagelets/inject/v2/pagelet/samples/
linkspagelet?content-
type=iframe&csapi=true&ifheight=123px&ifclass=myclass
```

This URL should result in markup similar to the code below.

> **Note:**
>
> The IFrame source points back to the inject API, but this time the content-type parameter is set to `html`. This feature adds an additional step in the pagelet retrieval. The `csapi` parameter is set to `true` on the subsequent call to get the IFrame contents so that the required CSAPI content is included in the IFrame (if this was not the case, JavaScript resolve errors would be returned because the pagelet code cannot access any CSAPI script included outside the IFrame).

```
<html>
 <head>
 </head>
 <body>
 <iframe frameborder="none" class="myclass" width="100%" height="123px"
scrolling="auto" src="http://proxy:port/inject/v2/pagelet/samples/linkspagelet?
asdg=asdfgas&param=true&content-type=html&jswrap=false&csapi=true">
```

```
<html>
<head>
<script src="http://proxy:loginserverport/loginserver/ensemblestatic/imageserver/
plumtree/common/private/js/jsutil/LATEST/PTUtil.js" type="text/javascript"> </script>
 <script src="http://proxy:loginserverport/loginserver/ensemblestatic/imageserver/
plumtree/common/private/js/jsutil/LATEST/PTDateFormats.js" type="text/javascript"></
script>
 <script src="http://proxy:loginserverport/loginserver/ensemblestatic/imageserver/
plumtree/common/private/js/jsxml/LATEST/PTXML.js" type="text/javascript"></script>
 <script src="http://proxy:loginserverport/loginserver/ensemblestatic/imageserver/
plumtree/common/private/js/jsportlet/LATEST/PTPortletServices.js" type="text/
javascript"></script>
 </head>

<body>
 <div id="pt-pagelet-content-1" class="pagelet-container" style="display: inline;">
 <span xmlns:pt="http://www.xmlns.oracle.com/xmlschemas/ptui/">
 Pagelet links:
 <br/>
 <a href="http://proxy:port/inject/RP_PID393219_I1/headpagelet1.html">The first
pagelet</a>
 <br/>
 <a href="http://proxy:port/inject/RP_PID393219_I1/headpagelet2.html">The second
pagelet</a>
 <br/>
 <a href="http://proxy:port/inject/RP_PID393219_I1/csapipagelet.html">The csapi
pagelet</a>
 <br/>
 <a href="http://proxy:port/inject/RP_PID393219_I1/linkspagelet.html">This
pagelet</a>
 <br/>
 </span>
 </div>
 </body>
 </html>
 </iframe>
 </body>
</html>
```

18.1.2.2 What You May Need to Know About Data Retrieval APIs

Two REST APIs are available to retrieve data from Pagelet Producer:

- **Pagelet API**: Allows remote applications to retrieve pagelet data from Pagelet Producer. (See the example for "All Pagelets" and "Pagelets by Library Name.")

- **Resource API**: Allows remote applications to retrieve resource data from Pagelet Producer. (See the example for "All Resources" and the example for "Resource By Name.")

The base URL for all requests is `http://host:port/pagelets/restservice/pageletproducer/.`

Example: All Pagelets

```
http://host:port/pagelets/restservice/pageletproducer/pagelets/
http://host:port/pagelets/restservice/pageletproducer/pagelets/?format=xml
```

Example: Pagelets By Library and Name

```
http://host:port/pagelets/restservice/pageletproducer/pagelet/libraryname/
pageletname/
```

```
http://host:port/pagelets/restservice/pageletproducer/pagelet/libraryname/
pageletname/?format=xml
```

Example: All Resources

```
http://host:port/pagelets/restservice/pageletproducer/resources/
http://host:port/pagelets/restservice/pageletproducer/resources/
```

Example: Resource By Name

```
http://host:port/pagelets/restservice/pageletproducer/resource/name
http://host:port/pagelets/restservice/pageletproducer/resource/name/?format=xml
```

18.1.3 How to Use Automatic Resizing with IFrames

The Pagelet Producer pagelet inject API can automatically resize the IFrame that encapsulates pagelet content. The resizing is done so that the IFrame stretches to fit the content within. To use this feature, the `ifwidth` and `ifheight` parameters must be set to 'auto' as shown in the example below:

```
http://proxy:port/inject/v2/pagelet/samples/linkspagelet?content-
type=iframe&csapi=true&ifheight=auto&ifwidth=auto&ifclass=myclass
```

In addition, this feature relies on an external page on the same domain as the consumer page. This page is included into the pagelet IFrame as an internal hidden IFrame. This page collects the sizing information and passes it on to the parent consumer page. This page must be deployed in the same directory as the consumer page.

The example below resizes the pagelet IFrame after it finishes loading. To add dynamic auto-resizing capabilities to user interaction activities after the initial load, simply add more event listeners for mouse and keyboard events.

```
<html>
 <head>
 <title>Resizing Page</title>
 <script type="text/javascript">
 function onLoad() {
var params =window.location.search.substring( 1 ).split( '&' );
 var height;
 var width;
 var iframe;

for( var i =0, l =params.length; i < l; ++i ) {
var parts =params[i].split( '=' );
 switch( parts[0] ) {
case 'height':
 height =parseInt( parts[1] );
 break;
 case 'width':
 width =parseInt( parts[1] );
 break;
 case 'iframe':
 iframe =parts[1];
 break;
 }
}
window.top.updateIFrame( iframe, height, width );
 }

if (window.addEventListener) {
```

```
window.addEventListener("load", onLoad, false)
 }else if (window.attachEvent) {
window.detachEvent("onload", onLoad)
 window.attachEvent("onload", onLoad)
 }else {
window.onload=onLoad
 }
</script>
 </head>
<body>
</body>
</html>
```

18.2 Adding a Pagelet to a Portal Page

Use the Page Editor to add a pagelet to a portal page. By default, pagelets appear in the **Mash-Ups** folder in the Default Portal Catalog. To add a pagelet to a page, navigate to the pagelet in the resource catalog and select it.

For detailed information about how to add resources to pages in a portal, see Adding and Editing Resource Catalog Components on a Page in *Building Portals with Oracle WebCenter Portal*

To configure a pagelet within a page, view the page in Edit mode and click the **View Actions** menu for the pagelet, then select **Parameters**.

19

Creating Pagelets: Examples and Advanced Topics

This chapter provides examples for how to create pagelets with Oracle WebCenter Portal's Pagelet Producer.

This chapter includes the following topics:

- Creating a Simple Pagelet (an Example)
- Consuming a Pagelet in WebCenter Portal (an Example)
- Consuming a Pagelet in WebCenter Portal (an Example)
- Consuming a Pagelet in WebCenter Interaction (an Example)
- Consuming a Pagelet in Oracle WebCenter Sites (an Example)
- Consuming WSRP Portlets as Pagelets
- Exposing Custom ADF Task Flows as WSRP Portlets
- Consuming WebCenter Portal Services as Pagelets in Sites
- Consuming Applications as Pagelets Using EBS11i
- Consuming OpenSocial Gadgets Using Pagelet Producer
- Guidelines for Effective Geometry Management and Pagination
- Advanced URL Rewriting

19.1 Creating a Simple Pagelet (an Example)

This section will give you a basic feel of how you can use Pagelet Producer to proxy a web page. We will proxy a simple static "Hello World" web page, cut one section out of that page, and present it as a pagelet that you can later insert in WebCenter Portal, WebCenter Interaction, on your own application page.

This section includes the following topics:

- Configuring the Initial Pagelet Producer Setup
- Creating a Resource
- Creating a Pagelet
- Clipping the Pagelet

19.1.1 Configuring the Initial Pagelet Producer Setup

For this example, let's assume that the Pagelet Producer server is running on `http://pageletserver.company.com:8889/pagelets/`.

1. First, let's check that Pagelet Producer is up and running.

   To do that we just need to access its URL (for example, `http://pageletserver.company.com:8889/pagelets/`). Figure 19-1shows what should be returned:

   Figure 19-1   Pagelet Producer Welcome Page
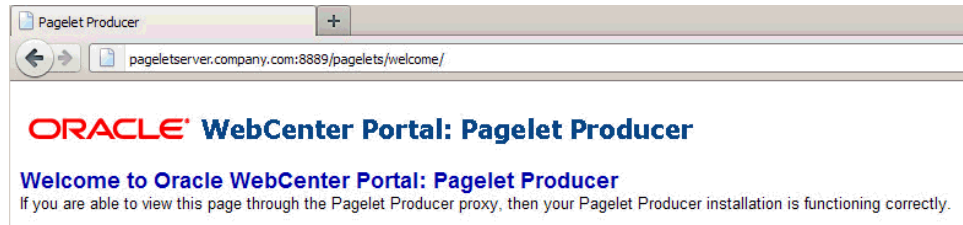
2. Log into the Pagelet Producer administration screens with your administrator credentials:

   For example, `http://pageletserver.company.com:8889/pagelets/admin`

   Figure 19-2   Pagelet Producer Administration Screen
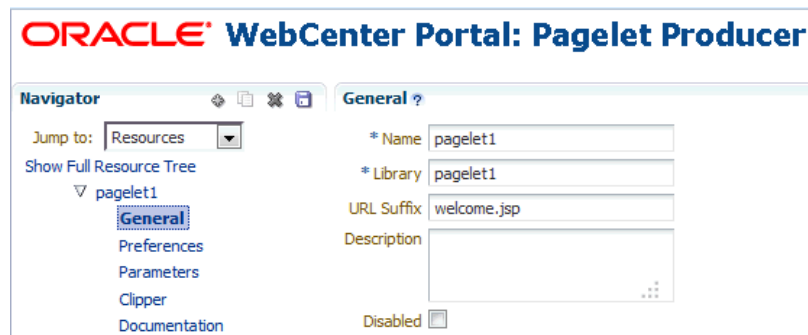
3. If you connect to the internet via a proxy server, you need to configure proxy in the Pagelet Producer settings:

   a. In the Navigator pane's **Jump To** drop down list, select **Settings**.

   b. Click **Proxy**.

   c. Enter your proxy server configuration as shown in Figure 19-3.

Figure 19-3   Pagelet Producer Proxy Settings
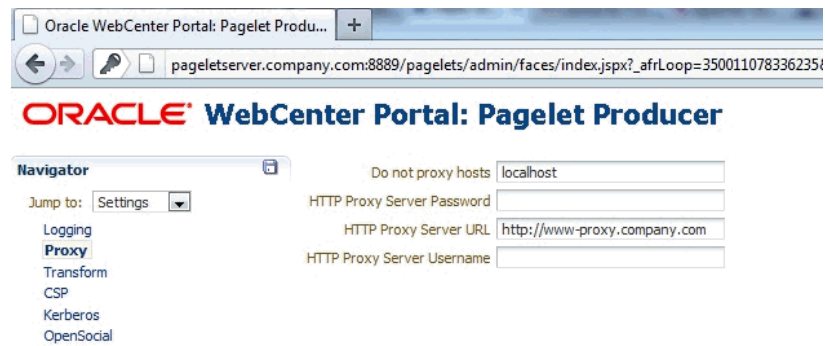
19.1.2 Creating a Resource

The first thing that you need to do is to create a resource for your web page. This will tell Pagelet Producer that all sub-paths of the web page should be proxied. It also will allow you to set up common rules for how your web page should be proxied and will serve as a container for your pagelets.

1. From the Navigator pane's **Jump To** drop down list select **Resources**.

2. Click any existing resource (for example, `welcome_resource`).

3. Click **Create selected type**.

4. From the Select Producer Type dialog, select **Web** and click **OK**.

5. After the resource is created, click the **General** node in the Navigation pane and specify the following values as shown in Figure 19-4:

    - **Name:** `AppServer`

    - **Source URL:** `http://appserver.company.com:1234/`

    - **Destination URL:** `/appserver/`

Figure 19-4   Pagelet Producer's General Settings Page
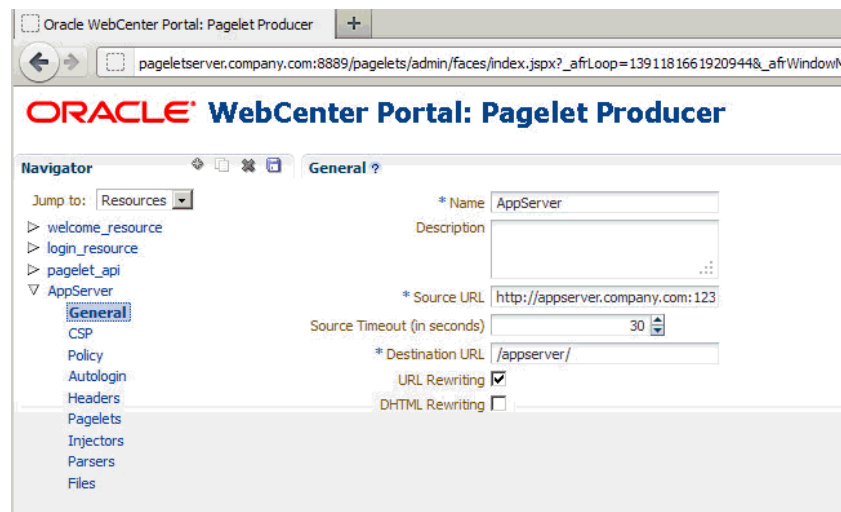
6. Click **Save**.

After the resource is created our web page becomes accessible by the URL:
`http://pageletserver.company.com:8889/pagelets/appserver/`

Figure 19-5   Hello World Pagelet

The original web page address Source URL has now been replaced with the Pagelet Producer URL (`http://pageletserver.company.com:8889/pagelets`) `+Destination URL`.

19.1.3 Creating a Pagelet

Let's continue by creating our "Hello World" pagelet.

1. Under the **Resource** node, open the newly created **Appserver** node.

2. Click **Create selected type**.

3. Click the **General** node of the newly created pagelet and specify the following values as shown in Figure 19-6:

Name: `Hello_World`

Library: `MyLib`

Library is used for logical grouping. The portals use the Library value to group pagelets in their respective UIs. For example, when adding pagelets to a WebCenter Portal portal you would see the individual pagelets listed under "Library".

URL Suffix: `helloworld/index.html`

The URL Suffix is where the Hello World page HTML is served from.

Figure 19-6   Pagelet Producer Settings for the Hello World Pagelet
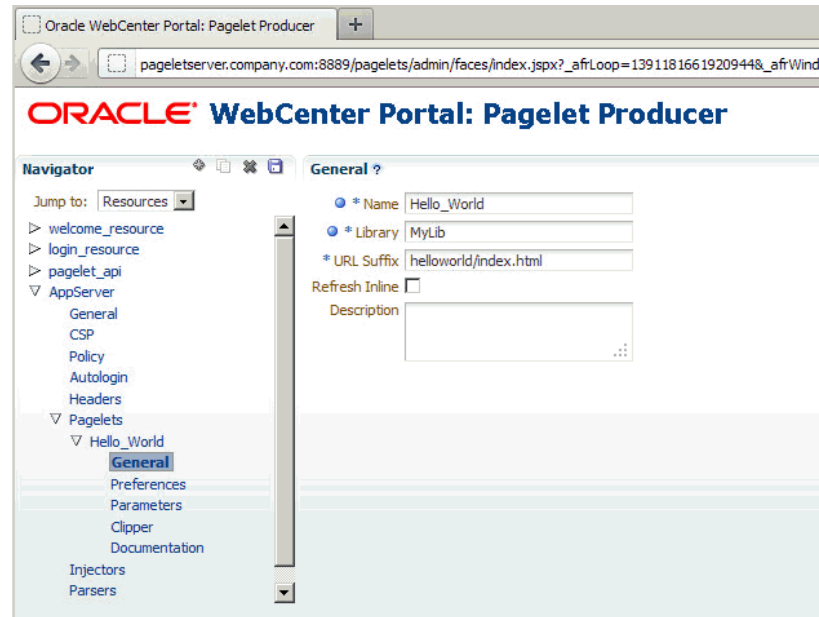
4. Click **Save**.

The Library name can be anything you want, it doesn't have to match the resource name at all. It is used as a logical grouping of pagelets, and you can include pagelets from multiple resources into the same library or create a new library for each pagelet.

After you save the pagelet you can access it here:

```
http://pageletserver.company.com:8889/pagelets/inject/v2/
pagelet/MyLib/Hello_World
```

which is:

```
http://pageletserver.company.com:8889/pagelets/inject/v2/
pagelet/ +[Library] +[Name]
```

Or to test the injection of a pagelet into iFrame you can click on the pagelet's **Documentation** node and use the **Access Pagelet using REST** URL.

Figure 19-7   Pagelet's Documentation Page
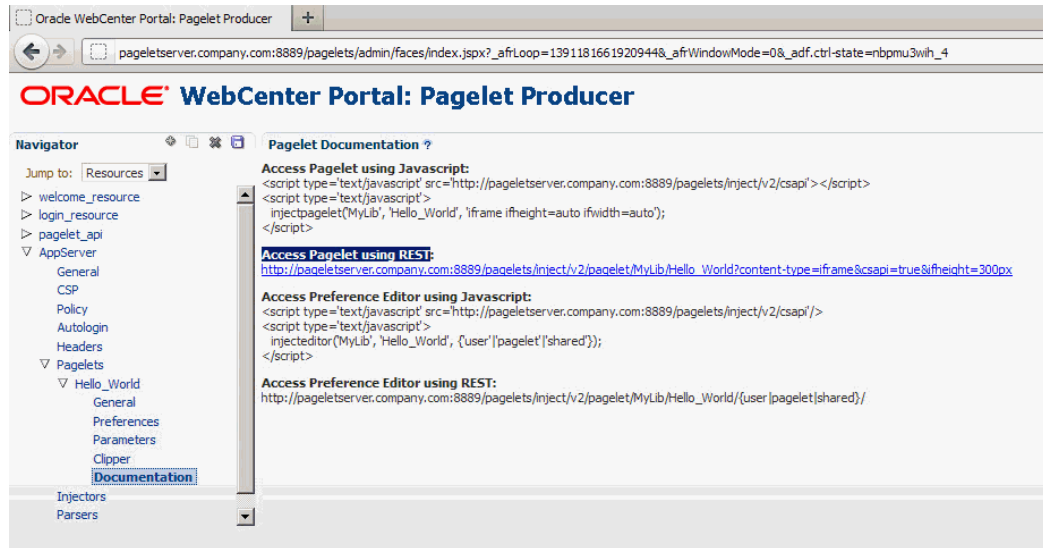
If you click the URL on the Documentation page you should see the following:

Figure 19-8   Hello World Pagelet

19.1.4 Clipping the Pagelet

The pagelet that we just created would cover the whole web page. Since we only want the "Hello World" segment of it we'll need to clip it as described in the following steps:

1. Under the **Hello_World** pagelet node, click **Clipper**.

2. Click **Create selected type**.

3. Specify a Name for the newly created clipper (for example: `c1`).

4. Click the clipper's **Content** node and click **Launch Clipper**.

Figure 19-9   Pagelet Producer Clipper
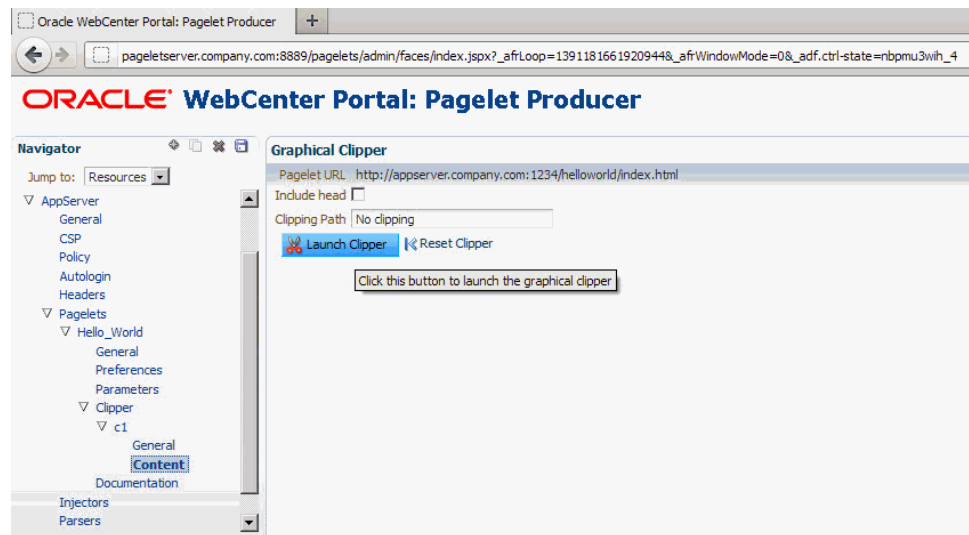
5. In the browser window, use the mouse to select the area you want to clip.

   When you click the mouse button the browser window disappears and a Clipping Path is automatically generated.

6. Save the clipped pagelet and access the link from the Documentation page again.

7. Our pagelet is now nicely clipped and ready to be used in a WebCenter Portal portal as shown in Figure 19-10:

Figure 19-10   Clipped Hello World Pagelet

19.2 Consuming a Pagelet in WebCenter Portal (an Example)

This section provides an example of how you can consume a simple "Hello World" pagelet in WebCenter Portal. Before continuing, first follow the instructions in Creating a Simple Pagelet (an Example) to create the "Hello World" pagelet that will be used in this example.

This section includes the following subsections:

- Registering the Pagelet Producer with WebCenter Portal
- Inserting the Pagelet into a Portal

19.2.1 Registering the Pagelet Producer with WebCenter Portal

In order to use our newly created pagelet from WebCenter Portal, we first need to register Pagelet Producer:

1. Log in to WebCenter Portal as an administrator.

   > **Note:**
   >
   > You must have the WebCenter Portal `Administrator` role or a custom role that grants the following permission:
   >
   > - `Portal Server-Manage Configuration`

2. From the **Portals** menu, select **Administration**, then click the **Settings** tab.

3. On the Tools and Service page, click **Portlet Producers**.

4. Click **Register**.

5. Select **Pagelet Producer** and enter the values shown for the following fields:

   Producer Name: `MyPageletProducer`

   Server URL: `http://pageletserver.company.com:8889/pagelets/`

6. Click **Test** to make sure that the connection was correctly configured.

   If everything is successful you should see the Test Status dialog display a message indicating that the test connection was successful.

7. Click **OK**.

   The Pagelet Producer is now registered.

19.2.2 Inserting the Pagelet into a Portal

For our exercise we will need to create a portal. So let's start by logging into WebCenter Portal and creating a portal called `myportal` using the default Portal template.

1. Log into WebCenter Portal as an administrator.

2. Create a portal called `myportal` using the Portal template.

   The Home page of the new portal opens in the portal editor.

3. Click **Add Content** in an area of the page where you want to add the pagelet to open the resource catalog.

4. Click **Pagelet Producers**.

5. Select the Pagelet Producer you previously registered.

   You will see the library `MyLib` containing the `Hello_World` pagelet.

6. Click **MyLib**, and select the `Hello_World` pagelet .

7. Save the page.

8. Click **View Portal**.

The Hello World pagelet is now inserted into the `myportal` Home page.

19.3 Consuming a Pagelet in WebCenter Interaction (an Example)

This section provides an example of how you can consume a simple "Hello World" pagelet in WebCenter Interaction (WCI) 10.3.0 or later. Before continuing, first follow the instructions in Creating a Simple Pagelet (an Example) to create the "Hello World" pagelet that will be used in this example.

This section includes the following subsections:

- Registering the Pagelet Producer Remote Server

- Creating the "Hello World" Web Service

- Creating the "Hello World" Portlet

- Using the Hello World Portlet on the WCI Home Page

19.3.1 Registering the Pagelet Producer Remote Server

In order to use our newly created pagelet from WCI, we first need to register Pagelet Producer:

1. Log into WCI as an administrator.

2. Click **Administration**.

3. Create the folder where we are going to keep all Pagelet Producer related objects by following the steps below:

a. Open the **Create Object...** dropdown and select **Administrative Folder**.

b. Enter `PageletProducer` as the **Name** and click **OK**.

Figure 19-11   WCI - Create Administration Folder
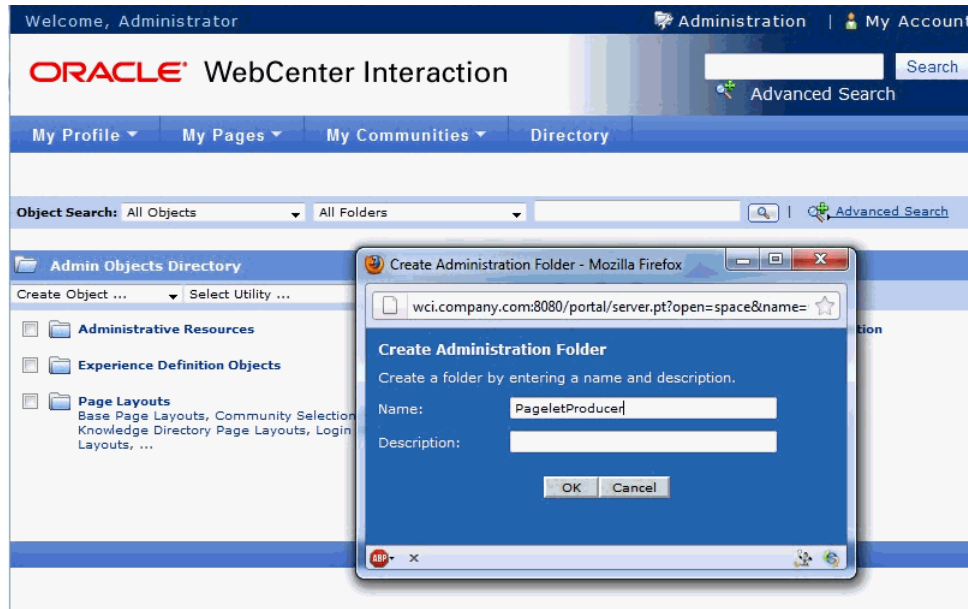
4. Click the newly created `PageletProducer` folder.

Figure 19-12   WCI - Newly Created Folder
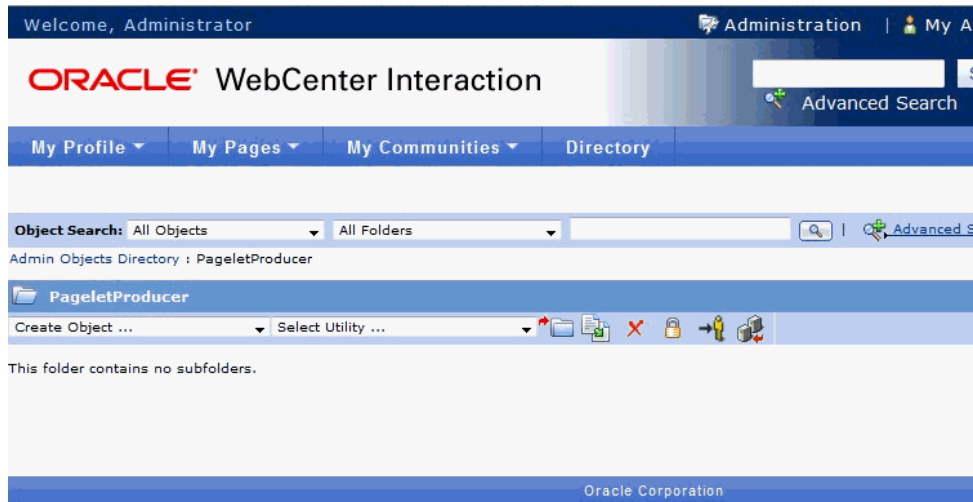
5. Open the **Create Object...** dropdown and select **Remote Server**.

Figure 19-13 WCI - Create Remote Server

6. In the **Base URL** field, enter `http://pageletserver.company.com:8889/pagelets` (this is the address of our Pagelet Producer server in this example).

7. Click **Finish**.

8. Select the `PageletProducer` folder and in the **Save As** field enter `Pagelet Producer Remote Server`.

Figure 19-14 WCI - Saving the Pagelet Producer Remote Server Object

9. Click **Save**.

You've now created a Pagelet Producer Remote Server connection in the PageletProducer folder.

Figure 19-15   WCI - Pagelet Producer Remote Server Object
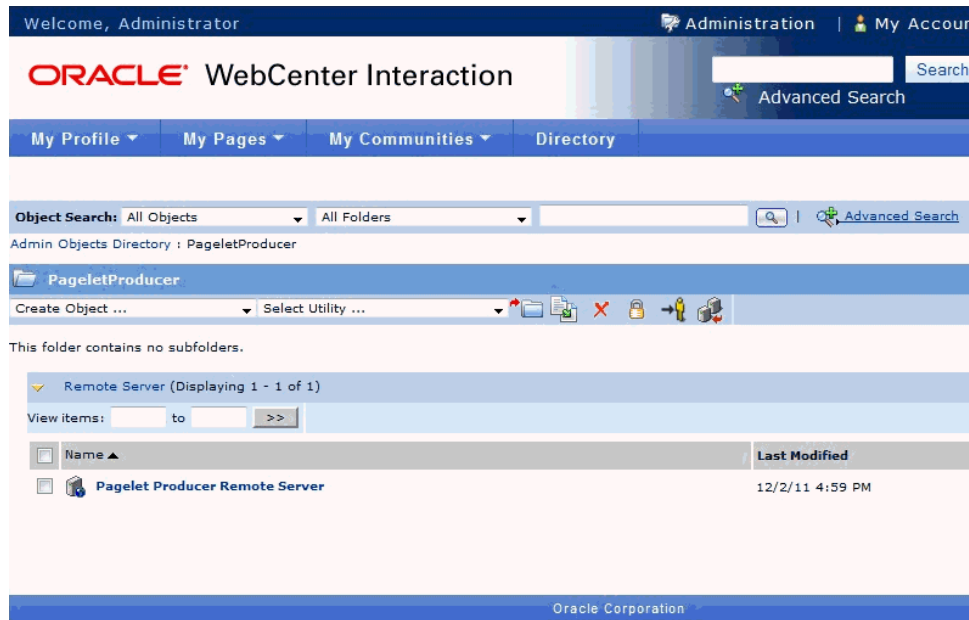
19.3.2 Creating the "Hello World" Web Service

In this section we'll continue by creating the "Hello World" web service.

1. Open **Create Object...** dropdown and select **Web Service - Remote Portlet**.

2. Click **Browse** and select **Pagelet Producer Remote Server**.

Figure 19-16   WCI - Choose Remote Server Dialog
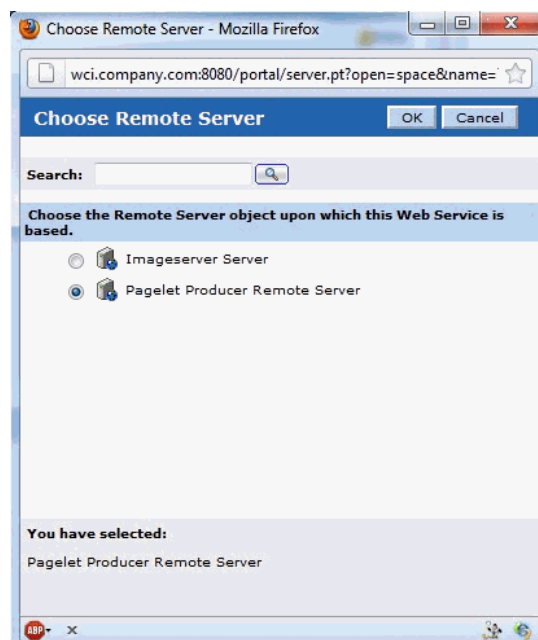
3. Click **OK**.

4. In the **Portlet URL** field enter:

```
inject/v2/portlet/MyLib/Hello_World?content-
type=iframe&csapi=true&ifheight=300px
```

Where `MyLib` is a library that contains our "Hello World" pagelet and `Hello_World` is the name of our pagelet.

Figure 19-17   WCI - Create Web Service - Remote Portlet Dialog

5. Click **Finish**.

6. Select the `PagaletProducer` folder and in the **Save As** field enter `Hello World Web Service`.

7. Click **Save**.

You've now created the Hello World web service.

19.3.3 Creating the "Hello World" Portlet

In this section we'll continue by creating a Hello World portlet.

1. Open the **Create Object...** dropdown list and select **Portlet**.

The Choose Template or Web Service dialog displays (Figure 19-18).

Figure 19-18 WCI - Choose Template or Web Service
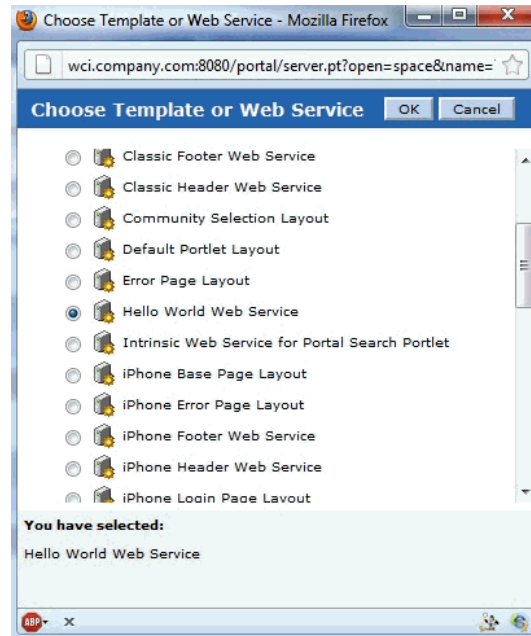
2. From the list of templates or web services select `Hello World Web Service` and click **OK**.

3. Under **Portlet Type/Size**, select the portlet type and size you want to use and click **Finish**.

4. Select the PageletProducer folder

5. In the **Save As** field, enter `Hello World Portlet` and click **Save**.

   You've now created the Hello World portlet.

19.3.4 Using the Hello World Portlet on the WCI Home Page

Now that we've registered the Pagelet Producer, set up the Pagelet Producer web service, and created the Hello World portlet, let's continue by using the portlet on a page in WCI. For the sake of simplicity, we'll use the WCI Home Page, but you could use any other WCI page.

1. In WCI, navigate to **My Pages > Home Page**.

2. Click **Edit Page** and open the `PageletProducer` folder.

3. Click Hello World Portlet.

4. Click **Close Editor**.

   Here's our Hello World pagelet inserted into the Home Page in WCI:

Figure 19-19   WCI - Hello World Pagelet on Home Page

19.4 Consuming a Pagelet in Oracle WebCenter Sites (an Example)

This section is aimed at developers who need to integrate content into Oracle WebCenter Sites 11*g* pages, including existing WSRP portlets or elements of web UI exposed by backend applications such as Oracle EBS. Developers should be familiar with Oracle WebCenter Sites and Oracle WebCenter Pagelet Producer and have a solid understanding of web technologies.

This section includes the following subsections:

- Adding Pagelets to Oracle WebCenter Sites

- Using Identity Propagation

- Propagating Identity from Pagelet Producer to the Backend Application

- Consuming WSRP Portlets as Pagelets

- Consuming WebCenter Portal Services as Pagelets in Sites

- Consuming Applications as Pagelets Using EBS11i

19.4.1 Adding Pagelets to Oracle WebCenter Sites

Once a pagelet and its related resources are configured, you can insert it into a web page using JavaScript or REST. For more information, see Adding a Pagelet to a Web Page.

Here, we'll use an approach in which pagelets are added directly to a page template in Oracle WebCenter Sites using an iFrame with a REST URL for accessing pagelets as a content source. This requires an addition to the page template script tag to load the

CSAPI JavaScript library (which adds necessary functions into the parent page), as well as the actual iFrame that loads pagelet content:

```
<script type="text/javascript" src="http://%PAGELET_PRD_HOST%/pagelets/inject/v2/
csapi"/>
<iframe id="pt-pagelet-iframe-1" width="100%" frameborder="0" src="http://
%PAGELET_PRD_HOST%/pagelets/inject/v2/pagelet/lib_name/pagelet_name?content-
type=html&consumepage=true&ifheight=auto"></iframe>
```

Where `lib_name` and `pagelet_name` refer to the library and pagelet configured in Pagelet Producer. For more information about parameters, see How to Access Pagelets Using REST.

19.4.1.1 Enabling IFrame Auto-Resizing

To provide a more seamless integration of the pagelet into the consuming page, you can make the pagelet iFrame behave more like inline markup by dynamically resizing to accommodate its content. To enable this feature, some extra configuration is required in both Sites and Pagelet Producer.

This section uses the AviSports sample site shipped with WebCenter Sites. To add any new asset to this site, you must log in to the Sites Administration page and open the Dev section in the left hand navigation as shown in Figure 19-20:

Figure 19-20 Sites Administration Page
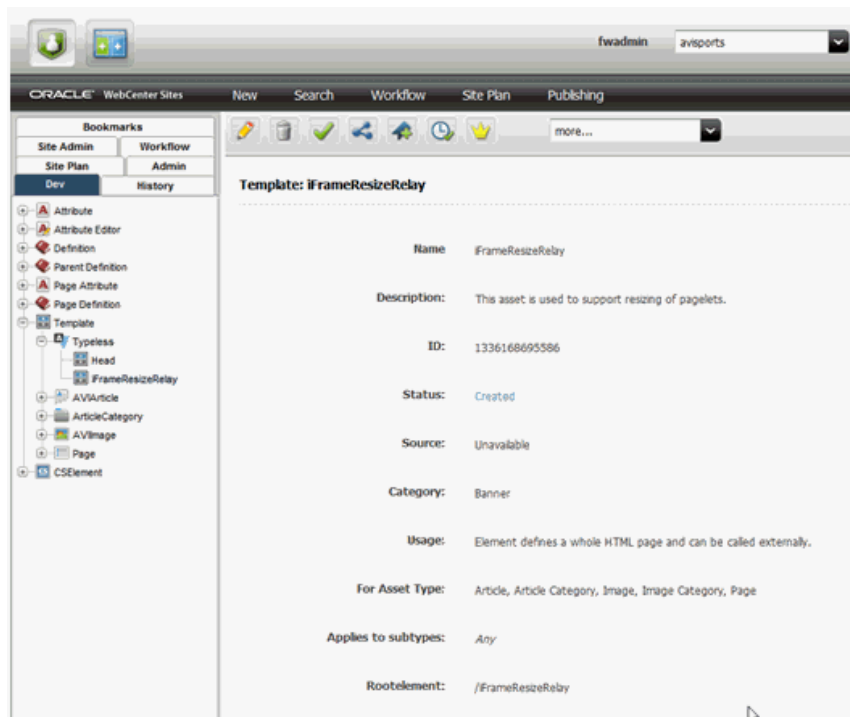
First, create a new template with the following settings:

- Name

  - Name: iFrameResizeRelay (or name of your choice)

  - For Asset Type: Applicable to various asset types (typeless)

- Element

- Usage: Element defines a whole HTML page and can be called externally

- Element Storage Path/Filename: iframe-resize.html (or name of your choice)

- Element Logic:

```
<html>
 <head>
 <title>Resizing Page</title>
 <script type="text/javascript">
function onLoad() {
var params =window.location.search.substring( 1 ).split( '&' );
var height;
var width;
var iframe;
for( var i =0, l =params.length; i < l; ++i ) {
var parts =params[i].split( '=' );
switch( parts[0] ) {
case 'height':
height =parseInt( parts[1] );
break;
case 'width':
width =parseInt( parts[1] );
break;
case 'iframe':
iframe =parts[1];
break;
}
}
window.top.updateIFrame( iframe, height, width );
}
if (window.addEventListener) {
window.addEventListener("load", onLoad, false);
} else if (window.attachEvent) {
window.detachEvent("onload", onLoad);
window.attachEvent("onload", onLoad);
} else {
window.onload=onLoad;
}
 </script>
 </head>
 <body>
 </body>
</html>
```

- Site Entry

  - Cache Rules: Cached (default)

- Save

- Record SiteCatalog Pagename as `%PAGENAME%` (if you used the default name, this would be `<site_name>/iFrameResizeRelay`)

The new template is addressable as follows:

```
http://%SITES_HOST%/cs/Satellite?pagename=%PAGENAME%
```

Make a note of this URL; it will be used to configure the Injector for the pagelet.

Next, apply the iFrame resizing feature by adding a new Injector to the pagelet using the Pagelet Producer Administration Console. Open the pagelet, select Injectors and create a new Injector with the following settings:

- General

  - Name: auto_resizer (or name of your choice)

  - URL Filter: <none>

  - MIME Filter: text/html

  - Inject Location: Before </head>

- Content: Select Text (default) and copy the JavaScript below into the text area

  Replace SITES_RESIZE_RELAY_PAGE in the JavaScript below with the URL to the template you created in the previous step.

```
<script type="text/javascript">
var SITES_RESIZE_RELAY_PAGE ="http://%SITES_HOST%/cs/Satellite?pagename=%PAGENAME
%"; //CHANGE ME!

if (window.addEventListener) {
window.addEventListener("load", calculateSizeFixed, false);
} else if (window.attachEvent) {
window.attachEvent("onload", calculateSizeFixed);
} else {
window.onload=calculateSizeFixed;
}
function calculateSizeFixed() {
var PTResizeIFrame =PTResizeIFrame ||{};
 if (PTPortalPage && PTPortalPage.portlets) {
for (var i in PTPortalPage.portlets) {
if ( PTPortalPage.portlets[i].id != "page") {
PTResizeIFrame.pageletInstanceID =PTPortalPage.portlets[i].id;
 break;
 }
}
}
else if (!PTResizeIFrame.pageletInstanceID) {
PTResizeIFrame.pageletInstanceID =1;
 }

if (!PTResizeIFrame.pageletResizePage) {
var match =window.location.search.match(/resizepage=[^&]*/);
 if (match != null && match.length > 0) PTResizeIFrame.pageletResizePage
=unescape(match[0].substr("resizepage=".length));
 else PTResizeIFrame.pageletResizePage =SITES_RESIZE_RELAY_PAGE;
 }
var agent =navigator.userAgent;
 var ffversion =agent.indexOf("Firefox") >= 0 ?
agent.substring(agent.indexOf("Firefox")).split("/")[1] : -1;
 var FFextraHeight =parseFloat(ffversion)>=0.1? 25 : 0;
 var scrollHeightExtra =15;
 if (agent.indexOf('MSIE') > 0) {
scrollHeightExtra =35;
 }
if (FFextraHeight > 0) scrollHeightExtra =FFextraHeight;
 var wrapper =document.getElementById( 'pt-inner-iframe-wrapper-'
+PTResizeIFrame.pageletInstanceID);
 if (wrapper ==null) wrapper =createRelayIFrame(PTResizeIFrame.pageletInstanceID);
```

```
 var iframe =document.getElementById( 'pt-inner-iframe-'
+PTResizeIFrame.pageletInstanceID);
 var height =0;
 var width =0;
 var iframename ='';

if( (document.contentDocument) &&
(document.contentDocument.documentElement.offsetHeight) ) {
height =document.contentDocument.documentElement.offsetHeight +FFextraHeight;
}else if( (document.contentDocument) &&
(document.contentDocument.body.offsetHeight) ) {
height =document.contentDocument.body.offsetHeight+FFextraHeight;
}else if (document && document.documentElement.scrollHeight ) {
height =document.documentElement.scrollHeight +scrollHeightExtra;
 }else if( document && document.body.scrollHeight ) {
height =document.body.scrollHeight +scrollHeightExtra;
 }else {
height =wrapper.offsetHeight;
 }
width =wrapper.offsetWidth;
 iframename ='pt-pagelet-iframe-' +PTResizeIFrame.pageletInstanceID;
 var qsSeparator =PTResizeIFrame.pageletResizePage.indexOf("?") >= 0 ? "&" : "?";

iframe.setAttribute("src", PTResizeIFrame.pageletResizePage +qsSeparator
+'height=' +height +'&' +'width=' +width +'&' +'iframe=' +iframename);
}

function createRelayIFrame(pageletInstanceId) {
var wrapper =document.createElement("div");
 wrapper.id ="pt-inner-iframe-wrapper-" +pageletInstanceId;
 var iframe =document.createElement("iframe");
 iframe.id ="pt-inner-iframe-" +pageletInstanceId;
 iframe.setAttribute("height", 0);
 iframe.setAttribute("frameborder", 0);
 iframe.setAttribute("width", 0);
 wrapper.appendChild(iframe);
 document.body.appendChild(wrapper);
 return wrapper;
}
</script>
```

Add the pagelet to the page template in Sites using a REST URL within an IFrame. In the IFrame, the id parameter should use the unique number identifying the pagelet IFrame (for example, "pt-pagelet-iframe-1"). You must add the following query string parameters to the pagelet URL to support IFrame auto-resizing:

- `resizepage=http://%SITES_HOST%/cs/Satellite?pagename=`
  `%PAGENAME%`

- `ifheight=auto`

For example:

```
<script type="text/javascript" src="http://%PAGELET_PRD_HOST%/pagelets/inject/v2/
csapi"/>
<iframe id="pt-pagelet-iframe-1" width="100%" frameborder="0" src="http://
%PAGELET_PRD_HOST%/pagelets/inject/v2/pagelet/lib_name/pagelet_name?content-
type=html& consumepage=true&resizepage=http://%SITES_HOST%/cs/Satellite?pagename=
%PAGENAME%&ifheight=auto">
</iframe>
```

> **Note:**
>
> In certain situations automatic resizing may not work properly when multiple pagelets are present on a page. This is known to occur with pagelets that require form auto-login to external authentication servers. In this case, only one pagelet can be configured for auto-resizing, while the others should use static IFrame sizing.

19.4.1.2 Changing Pagelet Styling

To make a pagelet fit better visually in a consuming WebCenter Sites page, an Injector may be used to add styles that override the original CSS. In order for the override to work correctly, it is important that the style definitions supplied by the Injector come after the styles defined by the backend application. The following example does this by injecting replacement CSS into the end of the <HEAD> section.

To restyle a pagelet, add a new Injector to the pagelet using the Pagelet Producer Administration Console. Open the pagelet, select Injectors and create a new Injector with the following settings:

- General:

  - Name: new_styles (or name of your choice)

  - URL Filter: <none>

  - MIME Filter: text/html

  - Injector Location: Before </head>

- Content

  ```
  Injector Location: Before </head>
  ```

Consuming Applications as Pagelets Using EBS11i includes a working example of an Injector used to re-style the EBS UI to match the sample AviSports site in WebCenter Sites.

19.4.2 Using Identity Propagation

The core purpose of Pagelet Producer is to proxy backend applications. Since Pagelet Producer acts as a "middle-man" between the browser and the backend application, identity propagation must be broken into two parts:

- Establishing Identity in Pagelet Producer

- Using a Login Page Supplied by Pagelet Producer

19.4.2.1 Establishing Identity in Pagelet Producer

Pagelet Producer typically injects web content into WebCenter Sites using an IFrame as shown in Figure 19-21.

Figure 19-21   Pagelet Producer Content as IFrame in Sites
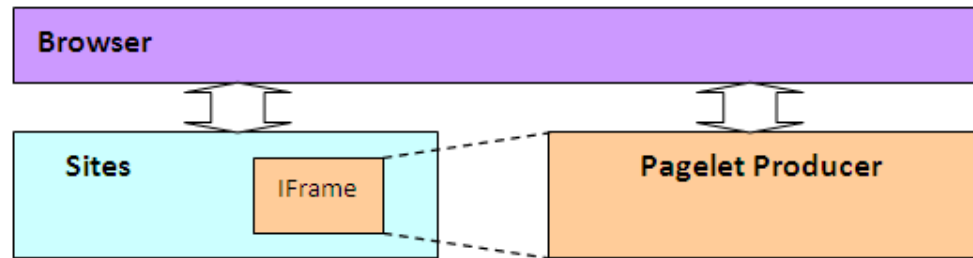
Since the content is injected as an IFrame, user identity must be established in both the Sites container and the Pagelet Producer container as shown in Figure 19-22. Note that Figure 19-22leaves out the user directory that is assumed to be shared between the two authentication schemes.

Figure 19-22   Different authentication schemes for Sites and Pagelet Producer
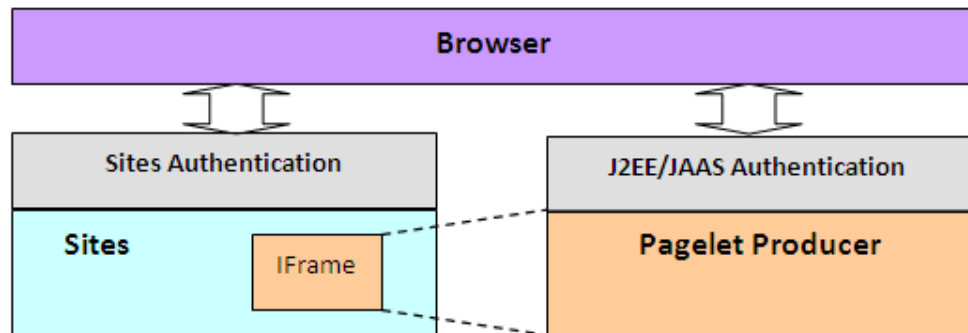
The ideal way to manage identity between the browser and the two containers would be to use Oracle Access Manager (OAM) as shown in Figure 19-23. Note that Figure 19-23leaves out the user directory that is assumed to be shared between the two authentication schemes, and excludes the OAM access server. Note also that this configuration is not supported for pre-11g versions.

Figure 19-23   Ideal Frontend Authentication Configuration Using OAM
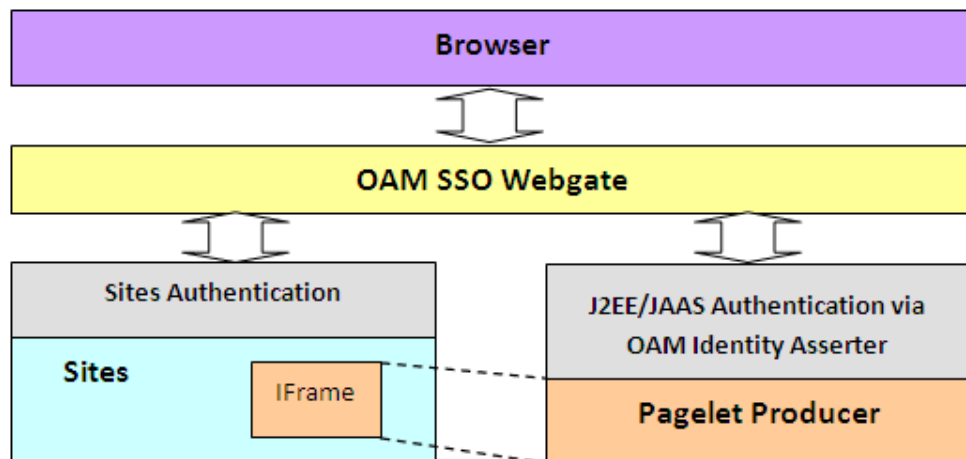
To set up OAM for WebCenter Sites 11*g*, refer to the chapter on "Oracle Access Manager Integration Setup" in *Oracle WebCenter Sites Configuring Supporting Software*.

19.4.2.2 Using a Login Page Supplied by Pagelet Producer

Before setting up OAM, it is sometimes useful to test the integration of Pagelet Producer in Sites. Without OAM, users must authenticate separately with Pagelet Producer and with Sites (see Figure 19-22). This section describes how to configure separate authentication with Pagelet Producer.

If OAM SSO is not present on either Pagelet Producer or Sites, the authentication scheme shown in Figure 19-24must be supported by ensuring that Pagelet Producer provides a login form.

To force Pagelet Producer to display a login form:

1. Identify a J2EE role to restrict access to the pagelet(s) or create a new role in the J2EE application server hosting Pagelet Producer.

   This example uses the global WebLogic Server (WLS) "Anonymous" J2EE role that is present in a default WLS installation.

2. Log in to the Pagelet Producer Administration Console, navigate to the resource(s) containing the pagelet(s) that must be surfaced in Sites and add the role you chose in step 1.

   Roles are defined on the Policy page for the resource.

   Figure 19-24   Pagelet Producer Administration Console - Policy Page
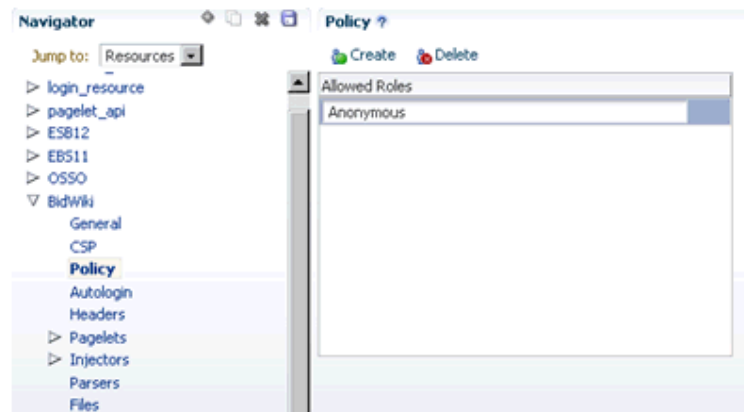
19.4.3 Propagating Identity from Pagelet Producer to the Backend Application

The way in which identity is established depends on the type of application, as described in the following topics:

- WSRP/JPDK Portlets

- Stand-alone Backend Application Protected with Native Authentication (Non-SSO)

- Identity Propagation with Autologin

- Stand-alone Backend Application Protected with SSO

- Reference: Common Autologin Configurations

19.4.3.1 WSRP/JPDK Portlets

WSRP portlets must use WSS tokens for identity propagation. For details on configuring security tokens, see Registering WSRP Portlet Producers in Pagelet Producer in *Administering Oracle WebCenter Portal*, and Consuming WSRP Portlets as Pagelets.

Each Oracle PDK-Java portlet that requires an identity will be passed credentials based upon information supplied by the external application's login form. For details, see Managing External Applications and Registering an Oracle PDK-Java Portlet Producer in *Administering Oracle WebCenter Portal*.

19.4.3.2 Stand-alone Backend Application Protected with Native Authentication (Non-SSO)

Pagelet Producer has a feature called Autologin that allows Pagelet Producer to interact with the "native" authentication mechanism established by a backend application.

19.4.3.3 Identity Propagation with Autologin

Since each backend application has its own means of authentication, a separate login prompt must be initiated by Pagelet Producer to collect any required credentials. Pagelet Producer resources can be configured to send credentials to backend applications as basic authentication headers, NTLM tokens, or Kerberos tokens, as well as manage forms-based authentication (typically via a session cookie). For information about configuring Autologin, see How to Configure Autologin.

> **Note:**
>
> Set the **Username** and **Password** fields to use the User Vault so that each user will be prompted for his or her own unique credentials to access the backend application.

19.4.3.4 Stand-alone Backend Application Protected with SSO

Determining the proper Autologin settings for a backend application can be very challenging. Doing so requires looking at request headers, identifying redirects, and examining markup (sometimes dynamically generated). This inspection and configuration process can be daunting if there are multiple applications for which Autologin is required. Therefore, if Oracle Access Manager (OAM) or Oracle SSO (OSSO) is available, it is highly recommended that one of these single sign-on solutions be used to protect backend applications.

Once the backend application is protected with OAM, there are two ways that Pagelet Producer can supply credentials as described in the following subsections:

- Using Direct Identity Propagation to OAM Protected Backend Application

- Using Identity Propagation with Autologin and SSO

19.4.3.4.1 Using Direct Identity Propagation to OAM Protected Backend Application

An OAM 11g WebGate is an Apache module running on Oracle HTTP Server (OHS) that manages all validation of user credentials with the OAM server. Once it has established the validity of a user it sends the OAM_REMOTE_USER header to the web

application on the application server that OHS is proxying. The application server will have an OAM identity asserter running on it that will set the JAAS user principal name to the OAM_REMOTE_USER header value.

If Pagelet Producer is protected with an OAM 11g WebGate it will receive an OAM_REMOTE_USER header. All web applications that it proxies will also receive this header. If the web application it proxies is on an application server with the OAM Identity Asserter, then the OAM Identity Asserter will use the OAM_REMOTE_USER header value passed from Pagelet Producer to set the user principal. This will all work as long as Pagelet Producer is not trying to proxy the URL to the WebGate for the protected web application. The rest of this section describes in details how this can be achieved in the context of Sites integration.

Example Setup

This example assumes that both WebCenter Sites and Pagelet Producer are protected by OAM WebGate as illustrated in Figure 19-23, and the following hosts and ports are used by the applications:

- OAM SSO WebGate: `www.example.com:80`

- WebCenter Sites: `sites_host:8080`

- WebCenter Pagelet Producer: `pp_host:8889`

- Backend application: `backendhost:8001`

- Full URL to backend application as viewed from browser (via WebGate)

  `http://www.example.com/backend/console`

- Full URL to backend application if accessed directly through application server (available only in internal network) `http://backendhost:8001/console`

Set up the Pagelet Producer resource to use the direct URL to the backend application, thus bypassing the WebGate (i.e., `http://backendhost:8001/console`).

Figure 19-25   Pagelet Producer Administration Console - General Properties Page
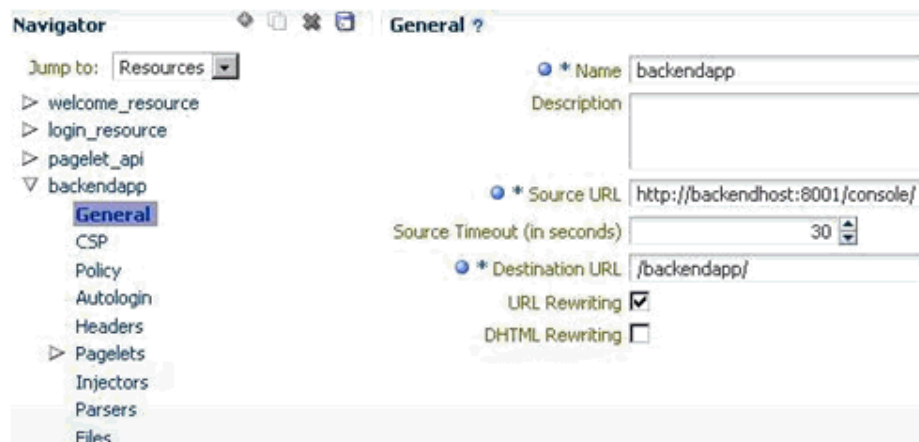
By setting up Pagelet Producer to call directly into the backend application (bypassing the WebGate) all the OAM secure headers that Pagelet Producer receives are passed to the OAM Identity Asserter on the application server hosting the backend application. Once the OAM Identity Asserter on the application server hosting the backend application receives the OAM headers it will set the proper user principal (see Figure

19-26). Note that Figure 19-26leaves out the user directory that is assumed to be shared between the two authentication schemes. It also excludes the OAM access server.

Figure 19-26   Pagelet Producer Calling the Backend Application Directly

---
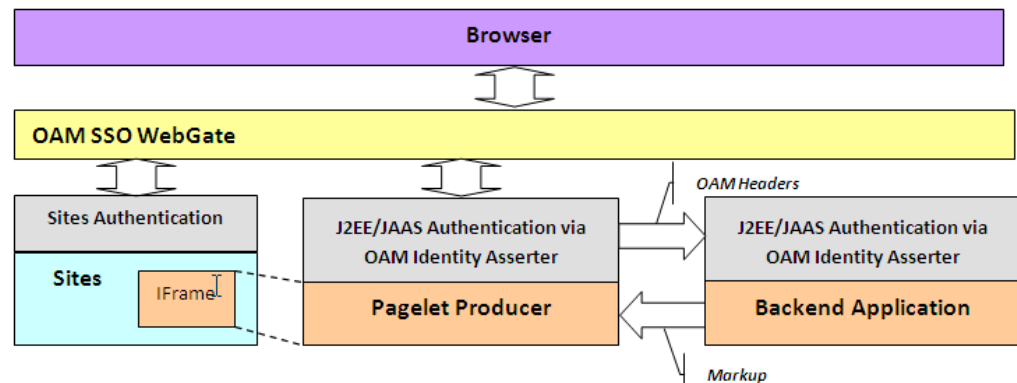
Note:

Although setting up Pagelet Producer to bypass the WebGate will achieve single sign-on, this solution should not always be applied. In cases where the OHS server hosting the WebGate is also managing load balancing for the backend application, bypassing the WebGate will also bypass load-balancing. All traffic to the application will be routed to one server. In this case we recommend configuring Pagelet Producer for Autologin as described in Using Identity Propagation with Autologin and SSO.

19.4.3.4.2 Using Identity Propagation with Autologin and SSO

If bypassing the OAM WebGate (as described in Using Direct Identity Propagation to OAM Protected Backend Application) is not a feasible option then credentials can still be supplied using Pagelet Producer's Autologin feature. The advantage of using Autologin with an Oracle SSO solution (OAM or OSSO) login page as opposed to individual login forms for each backend application (see Identity Propagation with Autologin) is that Autologin only has to be set up once and the configuration steps are the same for every implementation.

The steps for setting up Autologin with OSSO are described in Configuring Autologin.In a production environment, these steps can be followed as is with one exception: the `ssousername` and `password` form fields should be linked to the "User Vault".

Backend applications protected with OAM can also be set up following the steps described for OSSO in the Configure Autologin section in this document with the following exceptions:

- The login form URL will be a set location (e.g., `http://oam11g.mycompany.com:14100/oam/server/obrareq.cgi`).

  Be sure to remove the query string after the fixed location argument when setting the Autologin form identification URL.

- The form action URL will be a different value, but it should be a static value. Inspect the OAM form to determine the form action URL.

- The username and password fields may be different values. Inspect the OAM form fields to determine these values.

- Include the following two generated values: "request_id" and "OAM_REQ".

19.4.3.5 Reference: Common Autologin Configurations

This section provides a quick reference of Autologin settings in Pagelet Producer for common backend SSO scenarios.

This section contains the following topics:

- Autologin Settings for Oracle Access Manager (OAM)

- Autologin Settings for Oracle SSO (OSSO)

19.4.3.5.1 Autologin Settings for Oracle Access Manager (OAM)

Login Form Identification:

- URL: `http://%OAM_ROOT%/server/obrareq.cgi`

Form Submit Location:

- URL: `/oam/server/auth_cred_submit`

Form Fields:

- actionURL: static: `/oam/server/auth_cred_submit`

- request_id: Generated

- username: Unsecured or Credential Vault

- OAM_REQ: Generated

- password: Unsecured or Credential Vault

19.4.3.5.2 Autologin Settings for Oracle SSO (OSSO)

Login Form Identification:

- URL: `%OSSO_HOST%/sso/jsp/login.jsp`

Form Submit Location:

- URL: `%OSSO_HOST/sso/auth - POST`

Form Fields:

- appctx: Generated

- locale: Generated

- password: Unsecured or Credential Vault

- site2pstoretoken: Generated

- ssousername: Unsecured or Credential Vault

- v: Generated

19.5 Consuming WSRP Portlets as Pagelets

Pagelet Producer incorporates the WebCenter Common Consumer and therefore can expose WSRP portlets as pagelets for use in WebCenter Sites or any other web container that does not include a WSRP consumer.

The WSRP Portlet Producer can be registered with Pagelet Producer using Fusion Middleware Control, WLST, or the Pagelet Producer Administrative Console. For more information, see Registering WSRP Portlet Producers In Pagelet Producer in *Administering Oracle WebCenter Portal*. Once registration is complete, the WSRP producer appears as a resource in the Pagelet Producer Administrative Console, and the portlets associated with the WSRP endpoint will be listed in the pagelets collection for the resource.

> **Note:**
>
> Auto-generated WSRP resources and pagelets cannot be modified. To create a version that can be modified, choose the resource in the Pagelet Producer Administrative Console and click **Copy**. The cloned version of the resource can be edited and various elements such as Injectors can be added to customize pagelet functionality. For example, a custom Injector could be used to inject CSS classes to modify portlet markup to make it look and feel like the Sites that will host the pagelet.

If the WSRP Producer that you are with Pagelet Producer requires authentication, the following steps are required:

- Configure WS-Security between Pagelet Producer and the WSRP Producer as described in Configuring Web Services Security in *Administering Oracle WebCenter Portal*. You must ensure that the Java Key Store (JKS) is properly configured between the WSRP producer and Pagelet Producer; for details, see Setting Up the WebCenter Portal Domain Keystore in *Administering Oracle WebCenter Portal*.

- During the WSRP Producer registration, select the appropriate Token Profile in the Security section and enter the necessary configuration information. The path to the keystore must be absolute.

19.6 Exposing Custom ADF Task Flows as WSRP Portlets

Although it is possible to consume an ADF task flow in Pagelet Producer by 'clipping' it out of the page on which it is hosted, this is not the recommended approach. Considering the complexity of the markup generated by an ADF page and potential dependencies on other task flows or ADF page parameters, the recommended approach is to create a WSRP portlet based on the task flow and then consume the portlet in Pagelet Producer. One advantage of this approach is that ADF task flow parameters become portlet parameters that can be mapped to pagelet parameters, allowing information to be passed from the Sites page to the task flow. For step-by-step instructions on how to make a portlet from an ADF task flow, see How to Create a JSF Portlet Based on a Task Flow.

After the ADF task flow is exposed as a portlet, deploy a new WSRP Producer to a managed server in your WLS domain (for example, the `WC_Portlet` server on your WebCenter domain). Obtain the WSDL URL from the info page (`http://%PRODUCER_HOST%/%APP_CONTEXT%/info`) and make note of it. To register the

new WSRP Producer with Pagelet Producer, see WSRP Portlet Producers in Pagelet Producer in *Administering Oracle WebCenter Portal*.

19.6.1 Exposing WSRP Portlets Developed for Oracle WebLogic Portal

Oracle WebLogic Portal (WLP) supports a variety of portlet types, including Java Portlets, Java Server Faces Portlets, Java Server Page and HTML Portlets (for a complete list, see Portlet Types in *Portlet Development Guide for Oracle WebLogic Portal*). Remote (Proxy) Portlets are WSRP portlets and can be directly exposed as pagelets by the WSRP Producer with Pagelet Producer.

19.6.2 Exposing WLP Portlets Using WLP as WSRP Producer

WLP can expose Java, Page Flow, JSP, JSF, and Struts Portlets as a "complex producer" that includes the required WSRP interfaces, optional interfaces and some extended interfaces (for details see "WebLogic Portal Producers" in *Oracle Fusion Middleware Federated Portals Guide for Oracle WebLogic Portal*). Consequently, native WLP portlets can be consumed by Pagelet Producer as any other WSRP portlets. Configuration and best practices for WSRP interoperability between WLP and the WebCenter Common Consumer are described in WSRP Interoperability with Oracle WebCenter Portal and Oracle Portal in *Federated Portals Guide for Oracle WebLogic Portal*.

> **Note:**
>
> WLP portlets are WSRP-enabled by default since version 9.2. In earlier versions, you must edit the portlet properties and set the "Offer as Remote" property to true.

19.6.3 Configuring WS-Security between a WLP WSRP Producer and WebCenter Consumer

To consume WSRP portlets exposed by WLP in the WebCenter Common Consumer (Pagelet Producer), configuration is required on both the producer and consumer sides.

Typically it is impossible to consume WSRP portlets exposed by WebLogic Portal as an anonymous user, so it is necessary to configure SAML security on WLP for the producer and the WebCenter Common Consumer. Step-by-step instructions can be found in SAML Security Between a WebCenter Portal: Framework Application Consumer and a WebLogic Portal Producer in *Federated Portals Guide for Oracle WebLogic Portal*. These instructions are comprehensive, with the following adjustments:

- In section 5 of "Configuring the Producer," the steps for configuring keystore information on the WSRP consumer are executed in JDeveloper. You must complete this step in the WebCenter Common Consumer (Pagelet Producer). You can specify the signing alias and keystore password using WLST or Fusion Middleware Control. For details, see Setting Up the WebCenter Portal Domain Keystore in *Administering Oracle WebCenter Portal*.

> **Note:**
>
> Once you have configured the keystore values you must restart both the managed server hosting Pagelet Producer (by default `WC_Portlet`) and the WLS Admin Server for the JPS configuration changes to be picked up.

- In section 4 of "Configuring the Producer", the instructions do not emphasize the importance of the issuer name. The name placed here must match that sent by the consumer. For Oracle JPS, the default issuer ID sent is `www.oracle.com`.

19.6.4 Registering the WLP WSRP Producer in Pagelet Producer

To register the WLP WSRP Producer in Pagelet Producer, follow the standard steps for creating and configuring a new WSRP Producer described in WSRP Portlet Producers in Pagelet Producer in *Administering Oracle WebCenter Portal*. Make sure to include the following settings:

- WSDL URL:

  ```
  http://%WLP_HOST%/%PORTAL_APP_NAME%/producer/wsrp-1.0/markup?
  WSDL
  ```

- Security:

  Token Profile: Select "WSS 1.0 Token with Message Integrity"

After registration, a new Pagelet Producer resource is automatically created and populated with pagelets to represent the WLP portlets associated with this WSRP endpoint.

> **Note:**
>
> To allow user identity propagation to the portlets, both WLP and Pagelet Producer should be configured as described in Propagating Identity from Pagelet Producer to the Backend Application. (For testing purposes or for portlet content that can be accessed without authentication, you can specify a valid username in the Default User field under Security.)

19.6.5 Adding WLP WSRP Portlets to Sites

To add WLP WSRP portlets to a page in Sites, follow the steps in Adding Pagelets to Oracle WebCenter Sites. The URL to use for the `src` attribute in the IFrame tag that loads pagelet content can be found under "Documentation" in How to Access Pagelets Using REST.

Auto-generated WSRP resources and pagelets cannot be modified. To use Pagelet Producer features to alter the markup of the WLP portlet such as modify look and feel of the portlet UI by injecting custom CSS styles, you must create a new version of the resource. Choose the resource that was created for your WLP WSRP Producer in the Pagelet Producer Administrative Console and click **Copy**. The cloned version can be modified, including adding elements such as Injectors to customize pagelet functionality.

19.7 Consuming WebCenter Portal Services as Pagelets in Sites

WebCenter Portal includes services and tools, such as announcements and documents, that can be used to facilitate user collaboration in WebCenter Sites. For an overview of these tools, see Staying Informed and Staying Organized in *Using Oracle WebCenter Portal*.

The recommended approach for exposing these services and tools in Sites is to consume the associated WSRP portlets in Pagelet Producer by leveraging the Services Producer component included with Oracle WebCenter Portal (11.1.1.6 and later).

The following portal tools are exposed as WSRP portlets by default:

- Activity Stream

- Discussion Forums

- Mail

- Document Manager

- Lists

- Tag Cloud

- Blogs

- Announcements

> **Note:**
>
> Additional tools can be exposed by extending the Services Producer to a new JSF page and then adding the task flow to the page.

19.7.1 Requirements

Install WebCenter Portal following the instructions in *Oracle Fusion Middleware Installation Guide for Oracle WebCenter*. Make sure that all dependent components are also installed and configured, including WebCenter Content Server, the discussions server, and Pagelet Producer. For more information, see Requirements.

19.7.2 Configuring Security and Single Sign-On

There are two prerequisites to allow access to WebCenter Portal services from a Sites-driven web site:

- Create a common user base between the web site and WebCenter Portal

- Establish user identity propagation from the web site to WebCenter Portal

These are described in the following topics:

- Creating a Common User Base: LDAP Integration

- Establishing User Identity Propagation: OAM Configuration

- Configuring the GUID Attribute in the Identity Store

- Configuring SSO for Discussions Server

19.7.2.1 Creating a Common User Base: LDAP Integration

Creating a common user base requires configuring a common LDAP repository to be used by WebCenter Sites and all WebCenter Portal components. When selecting an LDAP server, make sure it is compliant with the WebLogic Server that runs WebCenter Portal components.

Things to remember when configuring security on WLS:

- Set the priority of the Authenticators to SUFFICIENT

- Set the Control Flag (in parenthesis) and order of authentication providers as follows:

  - OAM Identity Asserter (REQUIRED)

  - LDAP Authenticator (SUFFICIENT)

  - Default Authenticator (SUFFICIENT)

  - Default Identity Asserter (SUFFICIENT)

19.7.2.2 Establishing User Identity Propagation: OAM Configuration

To establish user identity propagation, we recommend configuring the SSO solution for WebCenter Sites and WebCenter Portal using Oracle Access Manager (OAM).

If you are using OAM, make sure it is configured to pass the user principal name in the ORACLE_REMOTE_USER header. Both ObSSOCookie and OAM_REMOTE_USER must be set to active on the OAM Identity Asserter on the WebLogic Server.

19.7.2.3 Configuring the GUID Attribute in the Identity Store

Specify the GUID attribute value to ensure that the value that is used in the identity store matches the value configured in the LDAP authentication provider. This value is configured in the `jps-config.xml` file.

Set the GUID for your LDAP authenticator in the `jps-config.xml` file.

```
<serviceInstance provider="idstore.ldap.provider" name="idstore.ldap">
<property
value="oracle.security.jps.wls.internal.idstore.WlsLdapIdStoreConfigProvider"
name="idstore.config.provider"/>
<property value="oracle.security.idm.providers.stdldap.JNDIPool"
name="CONNECTION_POOL_CLASS"/>
<property value="GUID=uuid" name="PROPERTY_ATTRIBUTE_MAPPING"/>
</serviceInstance>
```

19.7.2.4 Configuring SSO for Discussions Server

To use discussions, configure the discussions server for SSO before ordering the Authenticators in WebLogic Server (using the Default Authenticator and Default Identity Asserter). It is also possible to use WLST to configure SSO for the discussions server. For details, see Configuring the Discussions Server for SSO in *Administering Oracle WebCenter Portal*.

19.7.3 Importing WebCenter Services Exposed as WSRP Portlets

To import the WSRP portlets that expose WebCenter Services into Pagelet Producer, simply register the Services Producer as a WSRP Producer in Pagelet Producer using Enterprise Manager, WLST or the Pagelet Producer Administrative Console.

Figure 19-27 shows an example of WSRP endpoint registration in the Pagelet Producer Administrative Console. For details, see WSRP Portlet Producers in Pagelet Producer in *Administering Oracle WebCenter Portal*.

Figure 19-27   Pagelet Producer Administrative Console - WSRP Endpoint

The Token Profile in the Security section of this page must be set to WSS 1.0 SAML Token to support the Services Producer. You should also enter a suitable execution timeout for requests made by Pagelet Producer to the Services Producer (the default is 30 seconds).

Once registration is complete, the Services Producer will appear as a resource in the Pagelet Producer Administrative Console and all WSRP portlets exposed by the Producer will be listed in the pagelets collection for the resource:
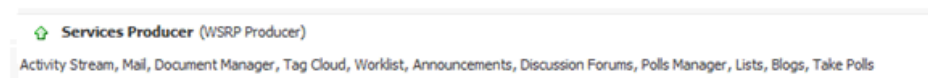
19.7.3.1 Adding Pagelets to Sites Pages

Any pagelet exposing WebCenter Services can be added to a WebCenter Sites page following the steps described in Adding Pagelets to Oracle WebCenter Sites. Figure 19-28 shows the Document Manager Service embedded into a page of the AviSports sample site in WebCenter Sites.

> **Note:**
>
> Auto-generated WSRP resources and pagelets such as the ones associated with the Services Producer cannot be modified. To add an Injector or make other modifications, you must create a version that can be edited. Choose the resource in the Pagelet Producer Administrative Console and click **Copy**. The cloned version of the resource can be edited and various elements such as Injectors can be added to customize pagelet functionality.

Figure 19-28   AviSports Sample Site with Document Embedded
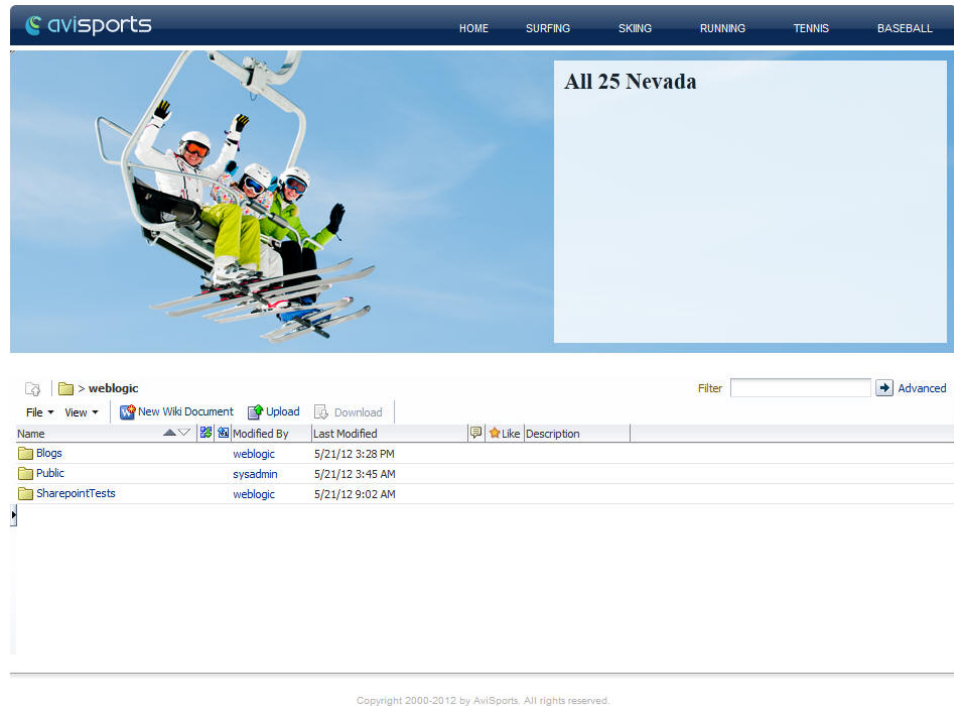
19.8 Consuming Applications as Pagelets Using EBS11i

This section provides detailed instructions for consuming the Oracle E-Business Suite 11i Order Information module UI as a pagelet. It covers autologin, navigation suppression, restyling, and URL rewriting.

This example was created using an EBS 11i instance that is protected with Oracle SSO. The following common terms and variables are used in the example:

- `%EBS_HOST%`: root URL of the EBS host (for example, `http://my-ebs.company.com:port/`)

- `%OSSO_HOST%`: root URL of the OSSO host (for example, `http://sso.company.com:port/`)

This section contains the following topics:

- Creating a Resource for Basic URL Mapping (Proxy)

- Configuring Autologin

- Creating a Pagelet

- Making Corrective Configurations

19.8.1 Creating a Resource for Basic URL Mapping (Proxy)

The following steps set up the basic URL mapping for a new EBS resource in Pagelet Producer:

1. Create a new a web resource with the following settings:

   - Name: EBS 11 (or your choice)

   - Source URL: %EBS_HOST%

     Make sure the URL is not too specific to avoid missing URLs used in the UI

   - Destination URL: /ebs11/ (or your choice)

   - URL Rewriting: on

   - DHTML Rewriting: on

2. Click **Save**.

19.8.2 Configuring Autologin

This step assumes that the EBS System is protected by OSSO and that Pagelet Producer will be configured to provide form Autologin rather than participate in SSO (a common scenario if the web site is not protected by SSO). EBS credentials may be shared for all users (useful for testing) or stored in the Pagelet Producer credential vault for each user.

1. Create a new "Web" resource with the following settings:

   - Name: OSSO (or your choice)

   - Source URL: %OSSO_HOST%

   - Destination URL: /osso/ (or your choice)

   - URL Rewriting: on

   - DHTML Rewriting: off

2. Click **Save**.

3. Under the newly created OSSO resource, configure Autologin using the Form Login option as follows:

   - Login Form Identification: URL - %OSSO_HOST%/sso/jsp/login.jsp

   - Form Submit Location: URL - %OSSO_HOST/sso/auth - POST

   - Form Fields:

     appctx: Generated

     locale: Generated

     password: Unsecured or User Vault

     site2pstoretoken: Generated

     ssousername: Unsecured or User Vault

- v: Generated

4. Click **Save**.

> **Note:**
>
> The names of login form fields are usually obtained by investigating the HTML source for the login page that protects the backend resource.

19.8.3 Creating a Pagelet

This step associates a pagelet with the Order Status page in the EBS Order Information module. To create a new pagelet, select the Pagelets section under the EBS resource you created and click the Create icon. Create the new pagelet with the following settings:

- Name: order_status (or your choice)

- Library: ebs11 (or your choice)

- URL Suffix: OA_HTML/RF.jsp?function_id=1005664&resp_id=22480&resp_appl_id=660&security_group_id=0&lang_code=US

- Refresh Inline: off

You can test the configured pagelet by accessing it via the REST link on the Documentation page. You should be taken to the Sales Order page without logging in (unless the pagelet uses the User Vault to store credentials and you are accessing it for the first time). From this page, you can use Simple Search to locate orders, using '%' as a wildcard character.

19.8.4 Making Corrective Configurations

After a pagelet has been created, it often requires additional features (Parser, Injector and/or Clipper) to either address issues with URL re-writing in the proxied markup, or simply to modify the markup for style or to hide unnecessary elements. This section describes corrective configurations that were applied to the sample pagelet that exposes the Order Information UI in EBS.

This section includes the following subsections:

- Creating a Pluggable Parser for Popup URL Rewriting

- Creating an Injector to Disable Frame-busting

- Creating an Injector to Auto-Resize the iFrame

- Setting Up Clipping by JavaScript Injection

- Creating an Injector for Pagelet Restyling

- Testing the Pagelets

- Final Result

- Troubleshooting

19.8.4.1 Creating a Pluggable Parser for Popup URL Rewriting

Parsers allow you to supplement or change the built-in logic for parsing content and finding URLs that need to be rewritten. When built-in parsers fail to identify URLs, custom parsers can be used to correct this failure. In the case of the EBS 11i UI, this step is required to fix popup handling in Advanced Search.

Under the EBS Resource, create a new Parser with the following settings:

- Name: popupRewriter (or your choice)

- URL Filter: *.jsp*

- Use Base Parser: on

- Fragment Locations

    - `var _jspDir='(.*?)';` (type Static URL)

    - `<frame .? src="(.?)` (type Static URL)

Figure 19-29 shows the settings for the popupRewriter.

Figure 19-29   popupRewriter

19.8.4.2 Creating an Injector to Disable Frame-busting

Injectors insert content into a specified location in the proxied resource page. The content can be any text, including HTML, CSS, JavaScript, and pagelet declarations. An empty injector may also be used to remove unwanted content from the page. The following injector was created for the EBS pagelet to prevent it from taking over the page, which was happening on the first displayed page where the user is asked to choose responsibility.

Under the EBS resource create a new Injector with the following settings:

- General

    - Name: frameBustDisabler

    - URL Filter: <none>

    - MIME Filter: text/html

    - Inject Location: Replace `target=_top`

- Content: (Text) `alt=''`

Figure 19-30 shows the parameter settings for the new Injector.

Figure 19-30   frameBustDisabler Injector

19.8.4.3 Creating an Injector to Auto-Resize the iFrame

In order to provide a more seamless integration into the consuming page, the pagelet iFrame can be configured to behave more like inline content by dynamically resizing to accommodate its contents. For details, see Enabling IFrame Auto-Resizing in this document.

19.8.4.4 Setting Up Clipping by JavaScript Injection

One option for suppressing page elements exposed as a pagelet is to create a clipper using graphical or Regular Expression-based clipping, available in Pagelet Producer. However, the recommended approach is to use custom JavaScript that is injected into the proxied page. This approach is often more flexible as it can easily accommodate the need to dynamically adjust rules that identify the element(s) to suppress or accommodate rules for suppressing multiple elements on the page.

Custom JavaScript can be added to the proxied resource page using an injector. To suppress EBS 11i standard header and navigation elements, use the following injector definition to insert a snippet of JavaScript that hides the unnecessary elements at page load time. (The standard Clipper feature in Pagelet Producer is not used due to the complexity of the EBS web UI.)

Under the EBS Resource, create a new Injector with the following settings:

- General:

    - Name: nav_suppressor (or your choice)

    - URL Filter: <none>

    - MIME Filter: text/html

    - Inject Location: Before `</body>`

- Content: (Text - copy and paste the content below)

```
<script type="text/javascript">
// this script is injected into the page by ensemble
// it hides header and footer tables at page load

//register handler function to the page load event
if (window.addEventListener) {
window.addEventListener('load', hideHeaderFooter, false);
} else if (document.attachEvent) {
window.attachEvent('onload', hideHeaderFooter);
```

```
}

//this function does the actual hiding
function hideHeaderFooter() {
var form =document.forms[0];
 if (typeof(form) =='undefined') return;
 //main span is form's second child span
 var spansFound =0;
 var mainSpan =null;
 for (var i =0; i < form.childNodes.length; i++) {
var child =form.childNodes[i];
 if (child.tagName && child.tagName.toLowerCase() =='span') {
if (++spansFound ==2) {
mainSpan =child;
 break;
 }
 }
 }
 if (typeof(mainSpan) != "undefined" && mainSpan != null) {
for (var i =0; i < mainSpan.childNodes.length; i++) {
var child =mainSpan.childNodes[i];
 if (child.tagName && child.tagName.toLowerCase() =='table') {
child.style.display ='none';
 }
 }
 }
 }
//call this function directly
hideHeaderFooter();
</script>
```

19.8.4.5 Creating an Injector for Pagelet Restyling

To make the EBS pagelet fit better visually in the consuming page, an injector can be used to add styles to override the original CSS.

The example injector definition below shows how to implement restyling for the AviSports sample web site in WebCenter Sites.

> **Note:**
>
> In order for the override to work correctly, it is important that the style definitions supplied by the injector come after the styles defined by the backend application. In the example below, the content is injected into the end of the <HEAD> section on the page.

Under the EBS Resource, create a new Injector with the following settings:

- General:

  - Name: avisports_styles

  - URL Filter: <none>

  - MIME Filter: text/html

  - Inject Location: Before </head>

- Content: (Text - copy and paste the content below)

```
<style>
* {
font: 11px Arial,Helvetica,sans-serif;
}
body {
background: url("http://sites-host:port/cs/avisports/images/BlueSliver.png")
repeat-x scroll center bottom transparent;
}
.OraTableCellText, .x1l, .OraTableCellNumber, .x1n, .OraTableCellIconButton, .x1p,
 .OraTableCellSelect, .x55,
.OraTableControlBarTop, .x1i, .OraTableControlBarBottom, .x1j {
background-color: #eceef1;
 border-color: #0b2a55;
}
.OraTableColumnHeader, .x1r, .OraTableSortableColumnHeader, .x20, .OraTableSortabl
eHeaderLink, .x23 {
background-color: #0b2a55;
 border-color: #F7F7E7;
 color: #336699;
}
.OraTableHeaderLink, .x24, .OraTableColumnHeaderNumber, .x1s, .OraTableColumnHeade
rIconButton, .x1t {
background-color: #0b2a55;
 color: #ffffff;
}
.p_InContextBrandingText, .x2l, .OraHGridNavRowInactiveLink, .x3v, .OraNavBarInact
iveLink, .x42,
.OraBINavBarInactiveLink, .x7n, .OraBINavBarInactiveLink_small, .x7o {
color: #000000;
}
.OraLink:link, .xd:link, .OraBreadCrumbs, .xh, .OraBulletedList a, .xj
a, .OraLinkText, .x2v,
.OraHGridNavRowActiveLink, .x3u, .OraNavBarActiveLink, .x41, .OraBINavBarActiveLin
k, .x7l, .OraBINavBarActiveLink_small, .x7m {
color: #0b2a55;
}
.OraBreadCrumbs a, .xh a {
color: #0b2a55;
}
</style>
```

19.8.4.6 Testing the Pagelets

The EBS11 resource now includes the pagelet, the custom injectors, and the custom parser as shown in .

Figure 19-31   EBS11 Project Resources

Test the pagelet by browsing to the Documentation page and using the URL shown below, substituting the local host ID for "example.com."

```
http://example.com:8889/pagelets/inject/v2/pagelet/ebs11/order_status?content-
type=iframe&csapi=true&ifheight=300px
```

19.8.4.7 Final Result

The following screenshots show the EBS 11i Order Information screens embedded into a page of the AviSports sample site in WebCenter Sites using a REST URL for pagelet access in an iFrame:

19.8.4.8 Troubleshooting

If some images are not rendered properly, make sure DHTML rewriting is enabled. (The DHTML Rewriting setting is on the General page for the resource.)

19.9 Consuming OpenSocial Gadgets Using Pagelet Producer

Pagelet Producer is an OpenSocial container that exposes the following features:

- People (exposes People Connections)

- Activities (exposes Activity Stream)

- Appdata (supports gadget personalization)

- Pub-Sub mechanism

Note that the People and Activities features require WebCenter Portal. To use the People or Activities features, target the WebCenterDS at the `WC_Portlet` managed server (i.e., at Pagelet Producer). For Appdata, Pagelet Producer supports user, instance and pagelet -scoped preferences.

To follow the example:

1. Open Pagelet Producer and create a new OpenSocial resource.

2. Define the URLs and policies as required.

3. Create a new pagelet specifying the URL for an external gadget.

   For example:

   ```
   http://bayareacoder.com/gogo/localweather/localweather.xml
   ```

4. If the gadget supports Preferences, configure the required parameters (see Figure 19-32and Figure 19-33).

   Figure 19-32   Preferences Editor

   Figure 19-33   Pagelet Parameters Dialog

5. Create or upload the gadget XML file (see Figure 19-34).

Figure 19-34   Pagelet Producer with Activities JavaScript and XML Options
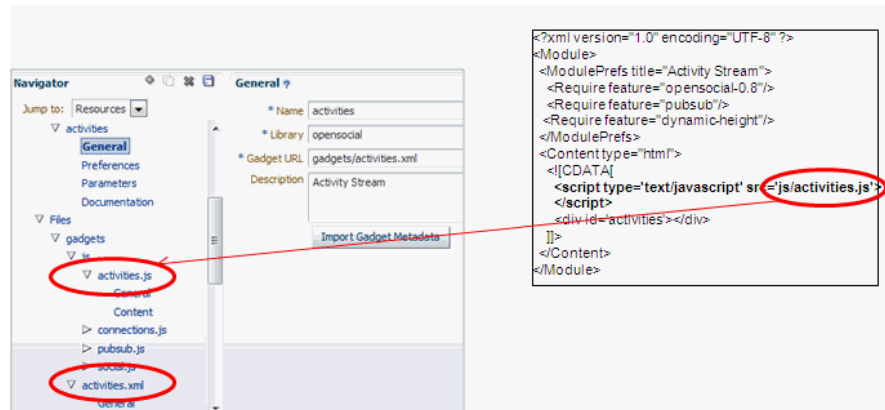
Note that the path supplied in the **Name** field is a 'virtual' URL that Pagelet Producer will use to serve the file and any path or name can be used.

6. Create or upload any other files that the new gadget requires (for example, any JavaScript files, images, and so forth), and you're done.

19.10 Guidelines for Effective Geometry Management and Pagination

Effective geometry management results in efficient sizing and placement of task flows on the page. If the task flows are not designed properly, you end up with pages that have a slow response time or lots of unnecessary scroll bars or white spaces between components. For example, if you include a task flow containing very little data inside a container that stretches it (for example, a Show Detail Frame component), there is a lot of white space below the data in the task flow. Poor design can also cause scrolling to be slow and unpredictable in a task flow with a table displaying a large amount of data. To avoid such issues, be sure to design your task flows so that they flow and use a conventional pagination model when displaying large data sets.

Follow the guidelines in this section when designing new task flows for WebCenter Portal. Also, be sure to adhere to these guidelines when you edit existing WebCenter Portal task flows.

This section includes the following topics:

- Using an Injector with JavaScript

- Using an Injector with SSL

19.10.1 Using an Injector with JavaScript

To avoid unnecessary white spaces in your task flows, you must ensure that the task flow content flows and is not stretched by the task flow container. A *stretching* task flow is one that is stretched to the height of the container, whereas a *flowing* task flow is one whose height is determined by its content, rather than by a stretching container or by itself having a fixed height. A flowing task flow with a small amount of data is shorter than one with large amounts of data

To design a flowing task flow, you must ensure that:

- The child component of the task flow does not have a fixed height.

- The Show Detail Frame component surrounding the task flow region does not have a fixed height (specified using its contentStyle attribute).

19.10.2 Using an Injector with SSL

Consuming an SSL resource requires a Pagelet Producer connection over HTTPS, including:

- Secure port mapping between WLS and Pagelet Producer

- Container should also be accessed via HTTPS

The injector can modify HTML markup that Pagelet Producer proxies:

- Based on URL pattern (resource scope)

- Based on MIME type

- Based on location (top or bottom) or exact text match (before, after, replace)

- Content page defines what gets injected

19.11 Advanced URL Rewriting

This section describes how to use Pagelet Producer's auto-login feature, and how to design a custom parser.

This section contains the following topics:

- Using Auto-Login

- Using a Custom Parser

19.11.1 Using Auto-Login

Auto-login lets you pass username/password credentials to a back-end resource. These can be passed as either:

- Static credentials – one set for all users

- Dynamically acquired from user at first resource access and then stored in secure Credential Vault

- User Profile (capture and pass username)

The supported authentication mechanisms for auto-login are:

- Basic login

- Form login (used in the following example)

- NTLM login

- Kerberos login

The example in Figure 19-35and the code snippet below shows how you can use auto-login to redirect to a login page:

Figure 19-35 Pagelet Producer - Auto-Login Example

The following code reflects the settings in Figure 19-35:

```
<form action="wwv_flow.accept" method="post" name="wwv_flow" id="wwvFlowForm"
autocomplete="off">
 <input type="hidden" name="p_flow_id" value="53557" id="pFlowId" />
<input type="hidden" name="p_flow_step_id" value="101" id="pFlowStepId" />
<input type="hidden" name="p_instance" value="724836084846701" id="pInstance" />
<input type="hidden" name="p_page_submission_id" value="1431956852758801"
id="pPageSubmissionId" />
<input type="hidden" name="p_request" value="" id="pRequest" />
<div id="login"> <div id="messages"></div> <div id="login-main">
<table id="myapp_layout_45149677586857064102" border="0" class="formlayout"
summary="" role="presentation" datatable=0 >
<tr><td align="right"><label for="P101_USERNAME" tabindex="999"><a class="optional-w-
help"
 href="javascript:popupFieldHelp('45149677686365064111','724836084846701')"
tabindex="999">
User Name</a></label></td> <td colspan="1" rowspan="1" align="left"><input
type="hidden" name="p_arg_names" value="45149677686365064111" /><input type="text"
id="P101_USERNAME" name="p_t01" value="" size="20" maxlength="100" class="text_
field" /></td></tr><tr><td align="right"><label for="P101_PASSWORD"
tabindex="999"><a class="optional-w-help"
href="javascript:popupFieldHelp('45149677888869064121','724836084846701')"
tabindex="999">Password</a></label></td> <td colspan="1" rowspan="1"
```

```
align="left"><input type="hidden" name="p_arg_names" value="45149677888869064121"/>
<input type="password" name="p_t02" size="20"
maxlength="100" value="" id="P101_PASSWORD" class="password" onkeypress="return
submitEnter(this,event)" /> <input type="button"
value="Login"onclick="myapp.submit('LOGIN');" id="P101_LOGIN" /> </td></tr>
</table> Please refer to "Using Myapp" in the Myapp User's Guide for more
information.</div> </div> <input type="hidden" name="p_md5_checksum" value="" />
<input type="hidden" name="p_page_checksum"
value="D95DBE8607E971820E98E93FC0EC064E" /></form>
```

19.11.2 Using a Custom Parser

Parsers let you extend or change built-in logic for URL parsing. Using a **URL Filter** and regular expressions, you can narrow down the part of the page that the parser should modify (Figure 19-36).

Figure 19-36   Pagelet Producer - Custom Parser Example
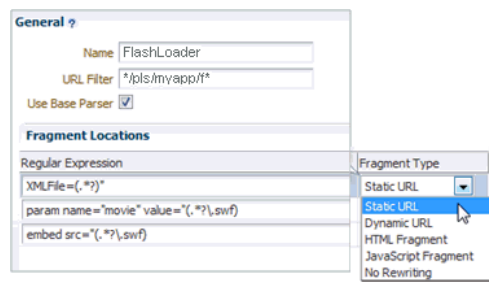
Using regular expressions lets you specify the HTML fragment to parse. The first grouping expression, denoted by parentheses, identifies the fragment itself, while the rest of the expression provides the context for finding it. A pluggable parser framework lets you fine-tune how the proxy recognizes different types of content based on MIME type, HTTP headers or URL.

In this example, there are three URLs in two places within the page markup that need to be modified to make Flash work:

```
<embed src=" /i/flashchart/anychart_5/swf/OracleAnyChart.swf?
XMLFile=http://example.com/pls/myapp/myapp_ util.flash?
p=53557:1:74175370346201:FLOW_FLASH_CHART5_R45192463162785599619_en"
 <param name="movie" value="/i/flashchart/anychart_5/swf/AnyChart.swf?
XMLFile=http://example.com/pls/myapp/myapp_ util.flash?
p=53557:1:74175370346201:FLOW_FLASH_CHART5_R45192463162785599619_en"
```

The following three regular expressions (shown under Regular Expressions in Figure 19-36) are used to achieve this:

- `XMLFile=(.*?)"`

- `param name="movie" value="(.*?\.swf)`

- `embed src="(.*?\.swf)`

Part V

Additional WebCenter Portal Customizations

This part contains the following chapters:

- Developing Shared Libraries

- Localizing Portals

- Extending Oracle Composer

- Customizing WebCenter Portal Impersonation Security

20

Developing Shared Libraries

This chapter provides an introduction to developing, extending, and deploying ADF assets such as task flows, data controls, and managed beans as a shared library for use in WebCenter Portal.

This chapter includes the following topics:

- Developing Shared Libraries for Use in WebCenter Portal
- Packaging and Deploying ADF Components for Use in WebCenter Portal

20.1 Developing Shared Libraries for Use in WebCenter Portal

Out-of-the-box, WebCenter Portal provides tools targeted at knowledge workers and developers to quickly and easily create portals, intranets, extranets, and self-service applications that offer users a more effective and efficient way to access and interact with information, business applications, and process. WebCenter Portal can be customized using browser-based tools and also using JDeveloper.

To help you understand how you can develop, test, deploy, and use your own ADF task flows, data controls, and managed beans in WebCenter Portal, Figure 20-1 shows the development lifecycle of a single task flow (called *mytaskflow*). Development steps 1 through 5 are described in Packaging and Deploying ADF Components for Use in WebCenter Portal.

Figure 20-1   Development Lifecycle for an ADF Task Flow
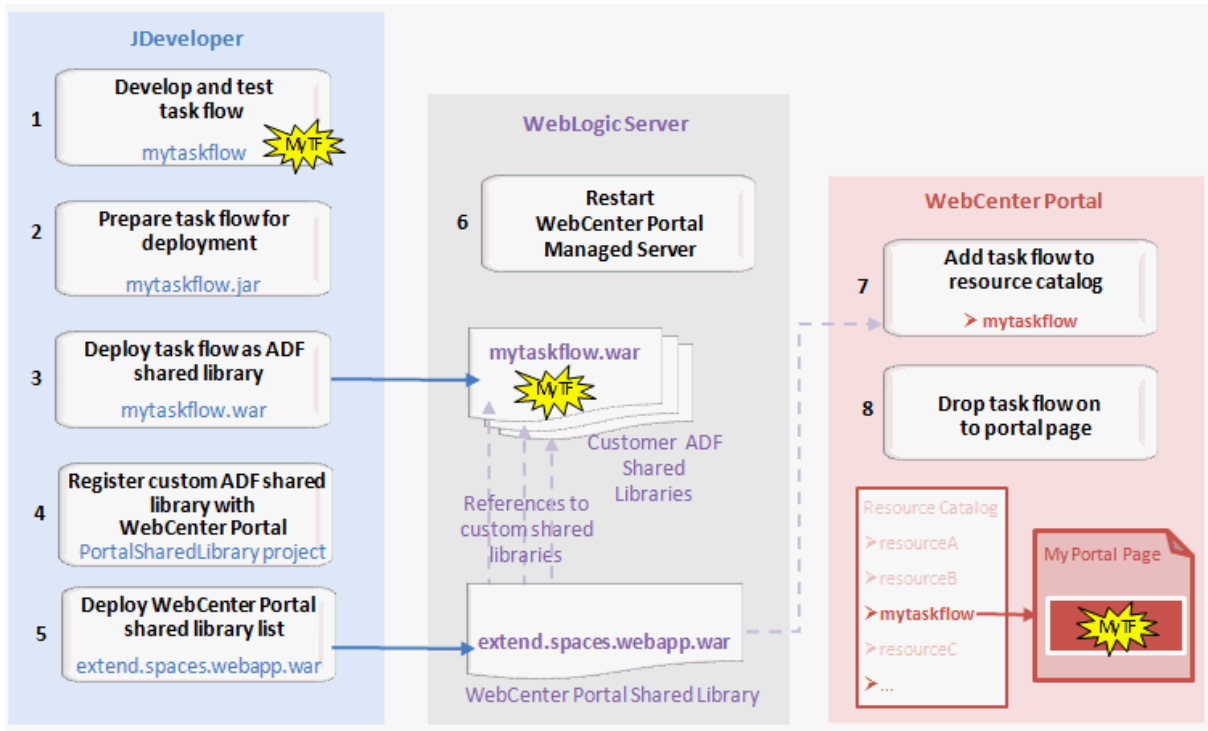
20.2 Packaging and Deploying ADF Components for Use in WebCenter Portal

This section describes how you can package and deploy your ADF task flows, data controls, and managed beans in your own custom shared libraries (WAR files) for use in WebCenter Portal.

WebCenter Portal provides a JDeveloper template called *WebCenter Portal Server Extension* that enables you to specify one or more custom shared libraries that you want to use with WebCenter Portal.

You can also use the same *WebCenter Portal Server Extension* template as a starter template to build and deploy ADF library components, such as task flows, data controls, and managed beans for use in WebCenter Portal.

This section includes the following topics:

- Understanding the WebCenter Portal Server Extension Template

- Understanding Shared Library Development for WebCenter Portal

- Creating a WebCenter Portal Server Extension Workspace

- Using Additional Shared Libraries with WebCenter Portal

- Deploying Extensions to the WebCenter Portal Shared Library (extend.spaces.webapp.war)

- Developing ADF Library Components for WebCenter Portal Using the PortalExtension Project

20.2.1 Understanding the WebCenter Portal Server Extension Template

The WebCenter Portal Server Extension template creates a workspace with two projects:

- **PortalSharedLibrary** – A project that when deployed creates `extend.spaces.webapp.war`, a shared library that:

  - References one or more custom shared libraries for WebCenter Portal.

  - Wraps the ADF Library JAR created from the PortalExtension project (only if used).

  > **Note:**
  >
  > Do not add any code to this project; this project only contains descriptor files that reference other shared libraries and/or includes ADF Library JARs.

- **PortalExtension** – A starter project in which you create custom ADF components, such as task flows, data controls, and managed beans. You can deploy this project to an ADF Library, which is added to the `extend.spaces.webapp.war` shared library.

  The components you create in a PortalExtension project typically require Java, ADF, and other related software development skills.

This section describes:

- How the WebCenter Portal Server Extension Workspace Is Organized

- The PortalSharedLibrary Project

- The PortalExtension Project

20.2.1.1 How the WebCenter Portal Server Extension Workspace Is Organized

The WebCenter Portal Server Extension workspace contains two default projects, PortalExtension and PortalSharedLibrary, each containing specific libraries and default files (Figure 20-2).

Figure 20-2   WebCenter Portal Server Extension Workspace in Application Navigator
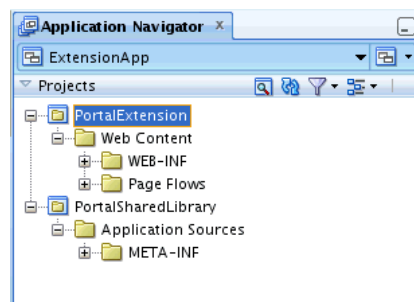
20.2.1.2 The PortalSharedLibrary Project

The PortalSharedLibrary project, shown in Figure 20-3, enables you to register one or more custom shared libraries that you want to use in WebCenter Portal and also

provides a convenient mechanism for deploying task flows, data controls, and managed beans (developed in the PortalExtension project) to a managed server running WebCenter Portal.

The PortalSharedLibrary project includes a WAR deployment profile called `extend.spaces.webapp`. When you deploy to `extend.spaces.webapp.war`, all the custom shared libraries that you register (in `weblogic.xml`) and any ADF library that you create and using a PortalExtension project are automatically included as dependencies of `extend.spaces.webapp.war`.

See Using Additional Shared Libraries with WebCenter Portal and Deploying Extensions to the WebCenter Portal Shared Library (extend.spaces.webapp.war).

Note:

Do not add any code to the PortalSharedLibrary project; this project only contains descriptor files that reference other shared libraries or include ADF Library JARs.

For general information on shared libraries, see General Documentation for Shared Library Deployment.

Figure 20-3   PortalSharedLibrary Project
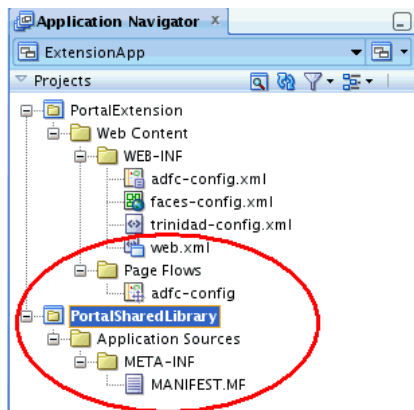
20.2.1.2.1 Versioning extend.spaces.webapp.war

The PortalSharedLibrary project contains a `MANIFEST.MF` file. Each time you update and deploy `extend.spaces.webapp.war`, edit this `MANIFEST.MF` file to increment the shared library implementation version. With each iteration, you must increment the `Implementation-Version` number in the `MANIFEST.MF` file, as shown in the example Manifest File for extend.spaces.webapp. The default implementation version is 12.2.1.0.1.

Manifest File for extend.spaces.webapp

```
Manifest-Version: 1.0
Ant-Version: Apache Ant 1.7.1
Created-By: 23.7-b01 (Oracle Corporation)
Extension-List: bpmSpaces
bpmSpaces-Extension-Name: oracle.bpm.spaces
Package:
Specification-Version: 2.0
Implementation-Version: 12.2.1.0.1
Implementation-Label: 12.2.1.0.1
```

```
Implementation-Vendor: Oracle
Implementation-Patch-Number:
Implementation-Patch-List:
Implementation-Title: Oracle WebCenter Spaces App Extension View V2
Extension-Name: extend.spaces.webapp
```

For detailed steps on how to change the implementation version, see Deploying Extensions to the WebCenter Portal Shared Library (extend.spaces.webapp.war).

20.2.1.3 The PortalExtension Project

The PortalExtension project is a standard ADF project where you can create custom components like task flows, data controls, and managed beans.

A PortalExtension project is deployed to an ADF Library. This library is automatically added as a dependency to the `extend.spaces.webapp.war` shared library file generated from the PortalSharedLibrary project. See Developing ADF Library Components for WebCenter Portal Using the PortalExtension Project.

For more information on ADF Libraries, see General Documentation for ADF Library Development.

Figure 20-4   The PortalExtension Project
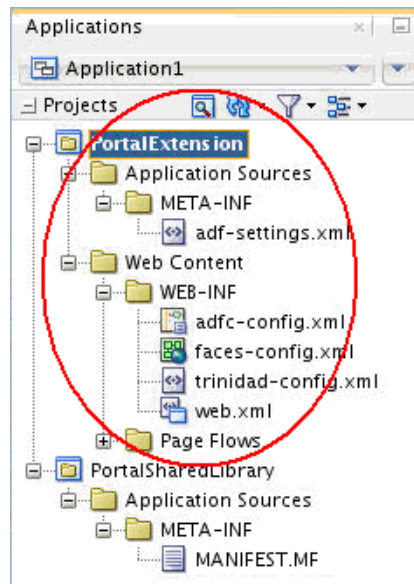
20.2.2 Understanding Shared Library Development for WebCenter Portal

WebCenter Portal includes the standard shared library `extend.spaces.webapp.war`. This WAR file can include a deployment descriptor (`weblogic.xml`) which references other shared libraries containing custom ADF library components.

If you have custom code or task flows deployed in several shared libraries from multiple sources that you want to use in WebCenter Portal you can list them in `extend.spaces.webapp.war`, as illustrated in Figure 20-5.

Figure 20-5   Reference Multiple Custom Shared Libraries in extend.spaces.webapp.war
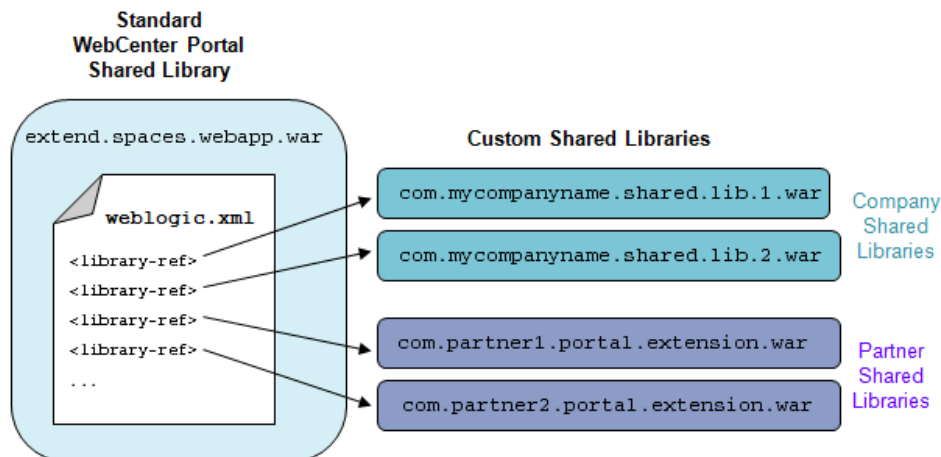
This development model provides an easy way to use additional shared libraries in WebCenter Portal from multiple contributors, including developers, customers, and partners.

Whenever you deploy a new shared library that includes extensions for WebCenter Portal you must register the name of your shared library with WebCenter Portal and redeploy `extend.spaces.webapp.war`. For details, see Using Additional Shared Libraries with WebCenter Portal.

This *Guide* does not attempt to teach you how to develop ADF task flows, data controls, and managed beans and package them into custom shared libraries since those techniques and procedures are not specific to WebCenter Portal. If you are new to shared library development, Oracle recommends that you familiarize yourself with the documentation listed at:

- General Documentation for ADF Library Development

- General Documentation for Shared Library Deployment

This *Guide* explains how to reference shared libraries that you want to use in WebCenter Portal using the WebCenter Portal Server Extension template. Table 20-1 provides an overview of the ADF library and shared library development processes, highlighting information that is specific to WebCenter Portal and pointing to other documentation where applicable.

Table 20-1   Developing and Deploying ADF Libraries and Shared Libraries for WebCenter Portal

Table 20-1   (Cont.) Developing and Deploying ADF Libraries and Shared Libraries for WebCenter Portal

Subject	WebCenter Portal Documentation	General Documentation
Creating ADF library components	WebCenter Portal supports several component types deployed to ADF Library JARs: task flows, data controls, and managed beans Create task flows, data controls, or managed beans for WebCenter Portal the same way as any standard ADF Library component. Alternatively, use the starter project supplied with WebCenter Portal. See Developing ADF Library Components for WebCenter Portal Using the PortalExtension Project.	About Reusable Components in *Developing Fusion Web Applications with Oracle Application Development Framework* describes various types of reusable ADF components, such as a task flows, that you can deploy to an ADF Library JAR. Note that Extension Libraries *does not* apply to WebCenter Portal. See also, Creating Reusable Components in *Developing Fusion Web Applications with Oracle Application Development Framework*.
- Naming conventions	Every ADF Library JAR file that you want to use with WebCenter Portal must have a unique file name. You cannot add two ADF Library JAR files with the same name even if their content is completely different.	See also, Naming Conventions in *Developing Fusion Web Applications with Oracle Application Development Framework*.
- Managing connections	If your ADF Library JAR uses a connection, you must manually register that connection with WebCenter Portal using the same connection name and details (just including the connection within the ADF Library JAR *does not* expose that connection in WebCenter Portal). To create connections for WebCenter Portal, you must use Fusion Middleware Control or WLST commands. To add a WebCenter Portal specific connection, refer to Managing Tools and Services in *Administering Oracle WebCenter Portal*. To add a database connection, refer to Creating and Managing JDBC Data Sources in the *Administering Oracle Fusion Middleware*. To add a web service connection, configure the ADFConnections MBean using the System MBean Browser. Refer to System MBean Browser in *Administering Oracle WebCenter Portal* and Web Service Connection in the *Administering Oracle ADF Applications*.	See also, Naming Considerations for Connections in *Developing Fusion Web Applications with Oracle Application Development Framework*.

Table 20-1   (Cont.) Developing and Deploying ADF Libraries and Shared Libraries for WebCenter Portal

Subject	WebCenter Portal Documentation	General Documentation
- Including descriptors	Descriptors in custom shared libraries, such as `web.xml` and `application.xml`, merge with corresponding descriptors deployed with the WebCenter Portal application. To avoid corrupting your WebCenter Portal installation, Oracle recommends that you *do not* include descriptor files in your custom shared libraries.	
- Versioning	Each time you redeploy the WebCenter Portal shared library WAR file (`extend.spaces.wepapp.war`) to add an ADF library JAR or to reference a custom shared library you must increment the shared library implementation version number. For details, see Versioning extend.spaces.webapp.war and Deploying Extensions to the WebCenter Portal Shared Library (extend.spaces.webapp.war).	
Packaging components into ADF Library JAR	Package your task flow, data control, or managed bean into a standard ADF Library JAR. Alternatively, use the starter **PortalExtension** project. For details, see Developing ADF Library Components for WebCenter Portal Using the PortalExtension Project.	Packaging a Reusable ADF Component into an ADF Library in *Developing Fusion Web Applications with Oracle Application Development Framework* describes how to package a reusable component, such as a task flow, to a ADF Library JAR file. Note that How to Place and Access JDeveloper JAR Files *does not* apply to WebCenter Portal.

Table 20-1   (Cont.) Developing and Deploying ADF Libraries and Shared Libraries for WebCenter Portal

Subject	WebCenter Portal Documentation	General Documentation
Adding ADF library components to a shared library	To use ADF Library components in WebCenter Portal, you must: 1. Deploy your ADF Library components to a shared library WAR file. See, Creating Shared Java EE Libraries and Optional Packages in *Developing Applications for Oracle WebLogic Server*. 2. Deploy your shared library to the managed server on which WebCenter Portal is deployed. 3. Register your shared library in the WebCenter Portal shared library (`extend.spaces.webapp.war`). 4. Deploy a new version of the WebCenter Portal shared library (`extend.spaces.webapp.war`). 5. Use Fusion Middleware Control or WLST to create any connections that your custom ADF Library components require. 6. Restart the managed server on which WebCenter Portal is deployed. **Note:** You can add ADF Library components directly to the WebCenter Portal shared library WAR (`extend.spaces.webapp.war`). For details, see Deploying Extensions to the WebCenter Portal Shared Library (extend.spaces.webapp.war).	Adding ADF Library Components into Projects in *Developing Fusion Web Applications with Oracle Application Development Framework* describes how to deploy ADF library components, such as a task flow, to a project. The information in this section *does not* apply to WebCenter Portal.

Table 20-1   (Cont.) Developing and Deploying ADF Libraries and Shared Libraries for WebCenter Portal

Subject	WebCenter Portal Documentation	General Documentation
Using ADF library components	An ADF Library adapter in WebCenter Portal's resource catalog automatically picks up any task flows, data controls and managed beans that you deploy and reference through `extend.spaces.webapp.war` and adds them to WebCenter Portal's resource registry. Custom components in the resource registry are exposed to WebCenter Portal users through resource catalogs as components that you can drop onto portal pages. 1. Log in to WebCenter Portal. 2. Add your custom component to a resource catalog. See, Adding a Resource to a Resource Catalog in *Building Portals with Oracle WebCenter Portal*. 3. Open a page with access to the resource catalog. 4. Add your custom component to the page. See, Adding a Component to a Page in *Building Portals with Oracle WebCenter Portal*.	What You May Need to Know About Using ADF Library Components in *Developing Fusion Web Applications with Oracle Application Development Framework* describes how various different ADF Library component types appear in JDeveloper. For WebCenter Portal, only the information pertaining to task flows, data controls, and managed beans applies.
Creating a shared library WAR	The WebCenter Portal Server Extension template provides a deployment profile for the WebCenter Portal shared library WAR `extend.spaces.webapp.war`. For details, see Creating a WebCenter Portal Server Extension Workspace.	Creating Shared Java EE Libraries in *Developing Applications for Oracle WebLogic Server* describes how to create a shared library WAR file.
Deploying a shared library WAR	The WebCenter Portal Server Extension template enables you to redeploy the WebCenter Portal shared library WAR `extend.spaces.webapp.war`. For details, see Deploying Extensions to the WebCenter Portal Shared Library (extend.spaces.webapp.war).	Deploying Shared Java EE Libraries and Dependent Applications in *Developing Applications for Oracle WebLogic Server* describes how to deploy any shared library WAR file.
Reverting to a previous library version	Use the WebLogic Server Administration Console to revert to a previous shared library WAR version or remove unwanted versions. For details, see Reverting to a Previous WebCenter Portal Shared Library Version.	Removing an ADF Library JAR from a Project in *Developing Fusion Web Applications with Oracle Application Development Framework* describes how to remove ADF Library components in JDeveloper. The information in this section *does not* apply to WebCenter Portal.

20.2.2.1 General Documentation for ADF Library Development

- For information about creating ADF task flows, data controls, and managed beans, refer to Developing Fusion Web Applications with Oracle Application Development Framework.

- See also, Reusing Application Components in *Developing Fusion Web Applications with Oracle Application Development Framework*.

20.2.2.2 General Documentation for Shared Library Deployment

- For additional information on versioning of shared libraries, see Deploying Shared Java EE Libraries and Dependent Applications in *Developing Applications for Oracle WebLogic Server*.

- See also, Deploying Shared Java EE Libraries and Dependent Applications in *Developing Applications for Oracle WebLogic Server*.

20.2.3 Creating a WebCenter Portal Server Extension Workspace

This section explains how to create a WebCenter Portal Server Extension workspace in JDeveloper in which you can:

- Register one or more custom shared libraries to use with WebCenter Portal.

- Create and deploy a custom ADF component for WebCenter Portal such as a task flow, data control, or managed bean.

To create a WebCenter Portal Server Extension workspace:

1. Download and install Oracle JDeveloper 12c (12.2.1.0.0).

    For details, see Installing Oracle JDeveloper.

2. Install the WebCenter Portal Extension for JDeveloper (12.2.1.0.0).

    For details, see Installing the WebCenter Portal Extensions for JDeveloper.

3. Open JDeveloper and choose **File** > **New**.

4. In the New Gallery dialog, open the **General** node and select **Applications**.

5. In the Items list, select **WebCenter Portal Server Extension**, and click **OK**.

    See Figure 20-6.

Figure 20-6   New Gallery Dialog
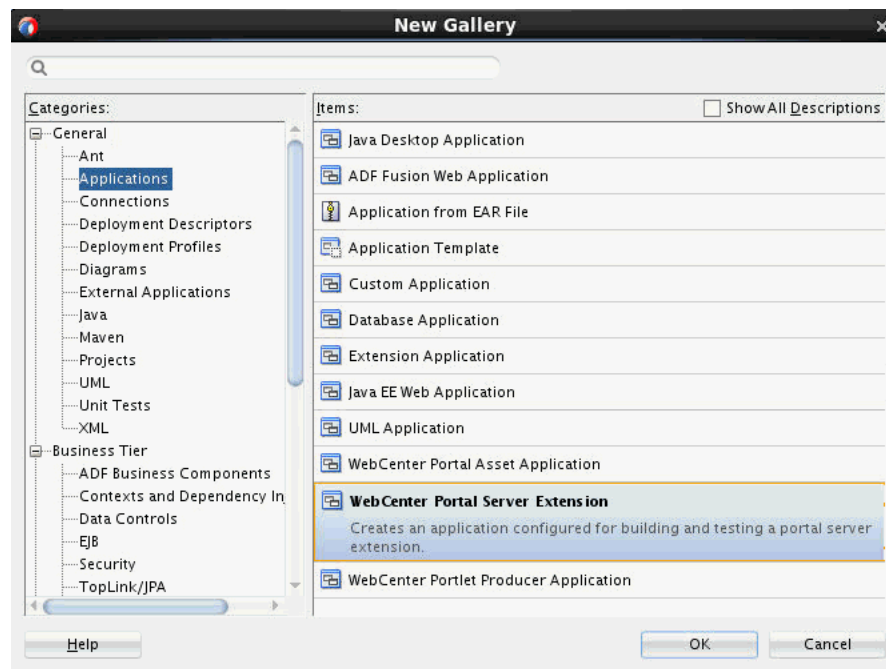
> **Note:**
>
> At this point, you are on the first page of the Create WebCenter Portal Server Extension wizard. The wizard includes four pages, which are explained in the following steps.

6. In the Create WebCenter Portal Server Extension dialog, enter a name for the workspace in the **Application Name** field, as shown in Figure 20-7.

Figure 20-7   Create WebCenter Portal Extension Wizard Step 1 of 4
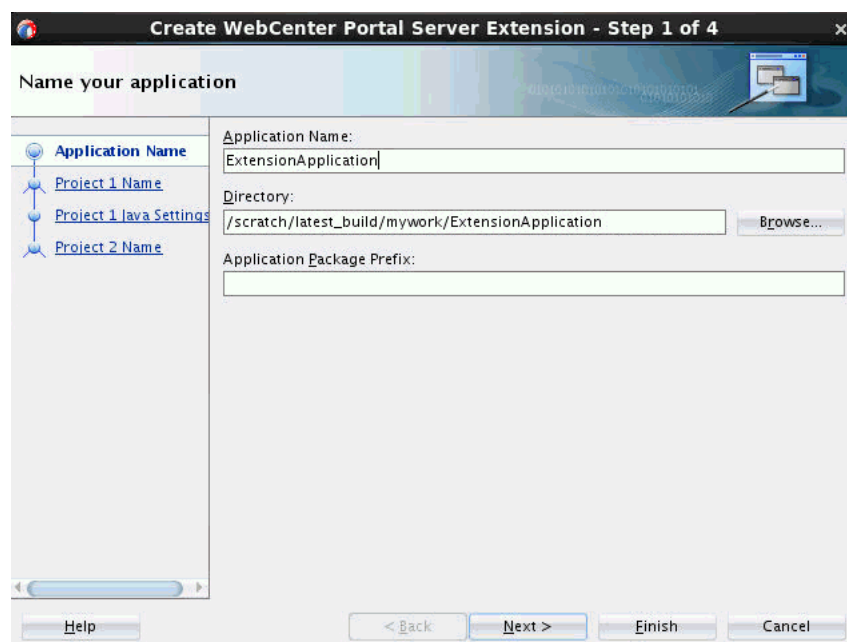
7. In the **Directory** field, enter a path to the base directory in which to store the workspace, or accept the default path.

For example:

```
/scratch/latest_build/mywork/ExtensionApplication
```

Optionally, click the **Browse** button to navigate to the desired directory.

8. Optionally, enter an **Application Package Prefix**.

This prefix is used for Java classes, interfaces, and packages created within the workspace. For example, a common prefix pattern is `com.companyDomainName`, like `com.oracle`.

> **Tip:**
>
> At this point, you could click **Finish** to create a project with default portal project names and directories. The following steps continue and review the rest of the wizard pages.

9. Click **Next**.

10. In the Project 1 Name part of the wizard, enter a **Project Name**, or use the default name as shown in .

This project provides all of the project technology components that are required for building and testing custom ADF components for WebCenter Portal (task flows, data controls, and managed beans).

Figure 20-8   Create WebCenter Portal Server Extension Wizard Step 2 of 4
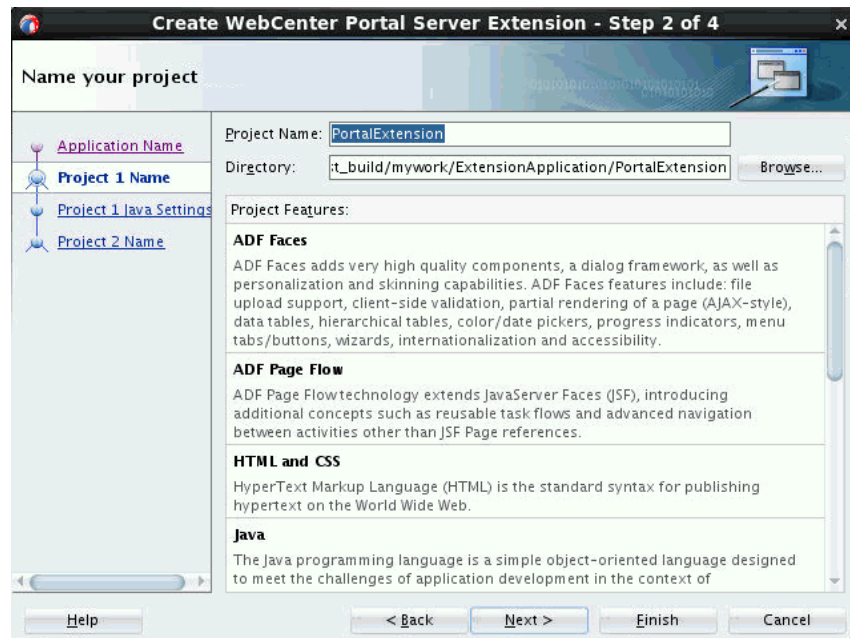
11. If you want to change the default directory for the project, enter it or use the **Browse** button to select a directory.

12. Click **Next**.

13. Optionally, edit the default package name and directories for Java source files and output class files for the project, as shown in Figure 20-9.

Figure 20-9   Create WebCenter Portal Server Extension Wizard Step 3 of 4
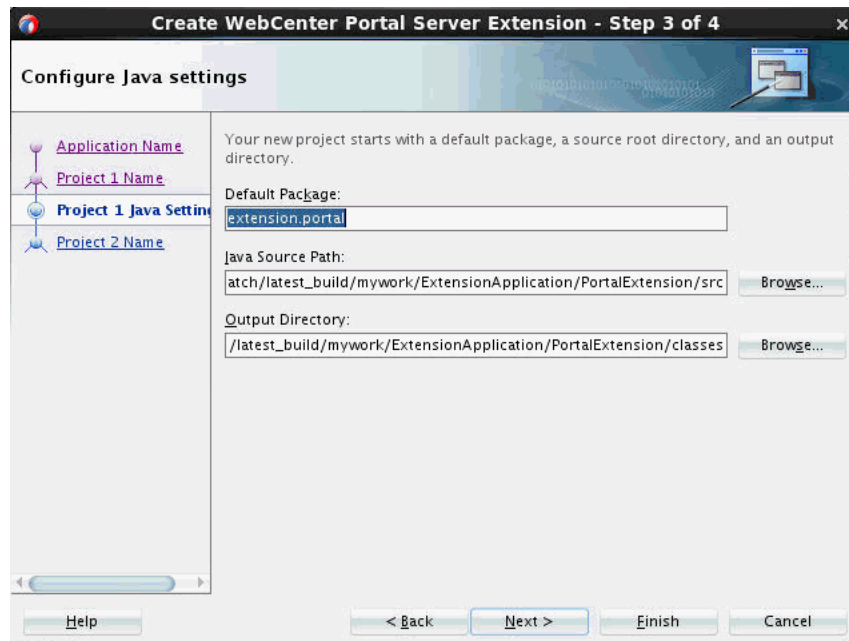
14. In the Project 2 Name part of the wizard, enter a **Project Name**, or use the default name as shown in Figure 20-10.

This project allows you to register one or more shared libraries that you want to use in WebCenter Portal, as well as deploy custom ADF components built using the PortalExtension project. See Deploying Extensions to the WebCenter Portal Shared Library (extend.spaces.webapp.war).

Figure 20-10   Create WebCenter Portal Server Extension Wizard Step 4 of 4
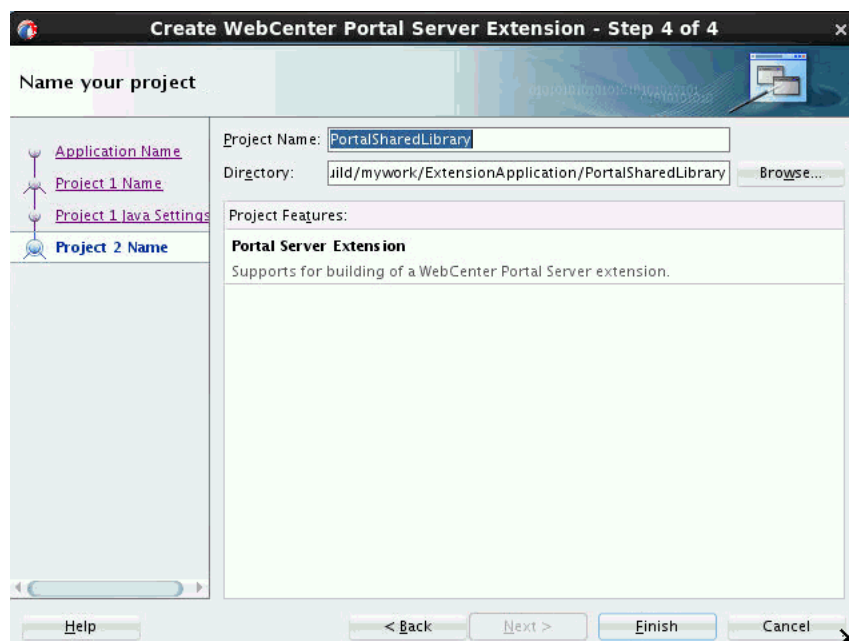
15. If you want to change the default directory for the project, enter it or use the **Browse** button to select a directory.

16. Click **Finish**.

20.2.4 Using Additional Shared Libraries with WebCenter Portal

If you have one or more shared libraries containing custom task flows, data controls or managed beans that you want to use in WebCenter Portal, you must register them in the `weblogic.xml` file associated with WebCenter Portal's shared library `extend.spaces.webapp.war`.

1. Select the **PortalSharedLibrary** project (Figure 20-11).

If you have not created a WebCenter Portal Extension workspace yet, refer to Creating a WebCenter Portal Server Extension Workspace.

Figure 20-11   PortalSharedLibrary Project
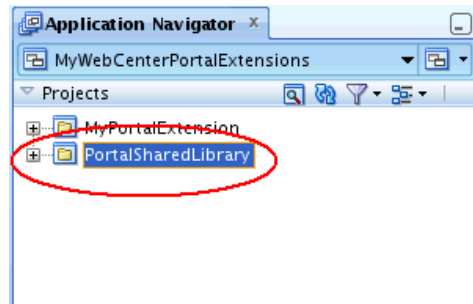
2. If the deployment descriptor `weblogic.xml` does not exist yet under `PortalSharedLibrary\Web Content\WEB-INF`, create the deployment descriptor as follows:

    a. Select **New > Deployment Descriptors > WebLogic Deployment Descriptor**

    b. Select **OK**.

    c. Select `weblogic.xml` from the list (Figure 20-12).

Figure 20-12   Create WebLogic Deployment Descriptor weblogic.xml
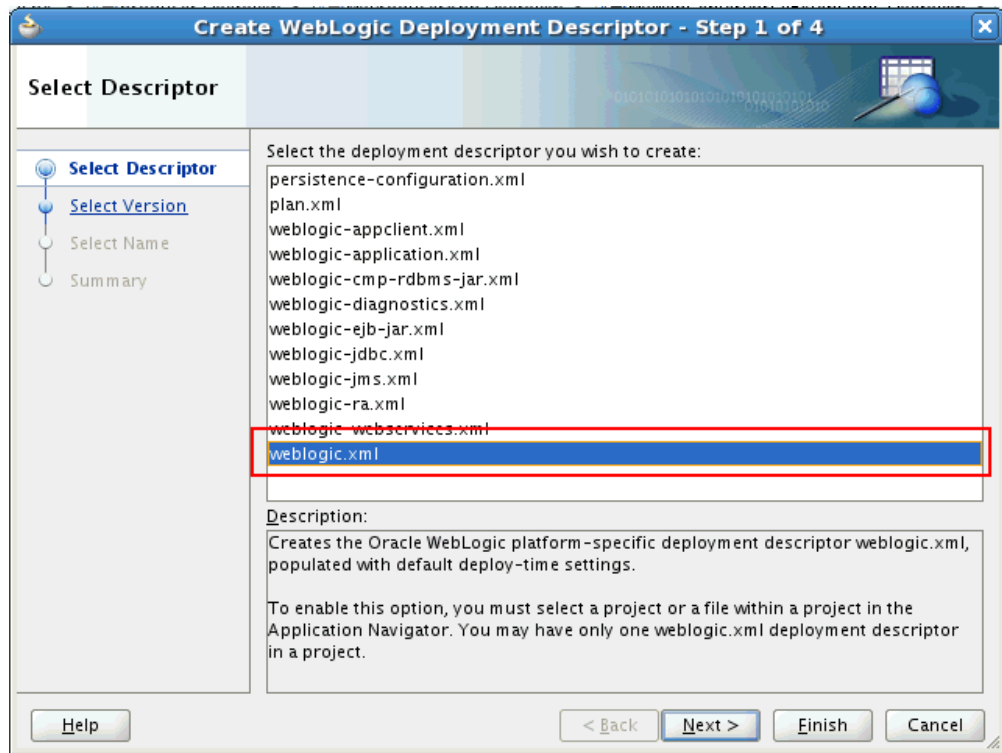
    d.   Select **Finish**.

3.   Open `weblogic.xml` (under `PortalSharedLibrary\Web Content\WEB-INF`).

    Initially, no additional shared libraries are listed in the file.

4.   In the Overview page, select **Libraries**.

5.   Specify the **Library Name** for each shared library that you want to use in WebCenter Portal.

    You can register a single shared library or multiple shared libraries, as shown in Figure 20-13.

    Click the Help icon for more information, if required.

Figure 20-13   weblogic.xml - Multiple Shared Library References
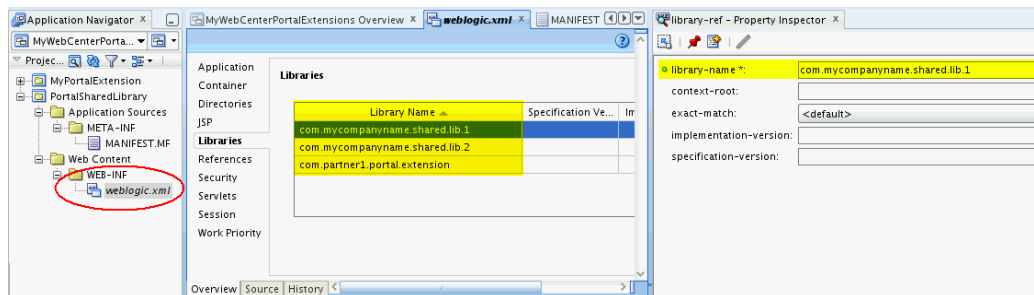
6.   Ensure each shared library that you reference is deployed on the WebCenter Portal managed server (named `WC_Portal` by default).

7. Redeploy a new version of the WebCenter Portal shared library WAR (`extend.spaces.webapp.war`) that includes your shared library references.

Remember to increment the version number in `MANIFEST.MF`. For details, see Deploying Extensions to the WebCenter Portal Shared Library (extend.spaces.webapp.war).

20.2.5 Deploying Extensions to the WebCenter Portal Shared Library (extend.spaces.webapp.war)

This section explains how to deploy extensions to the WebCenter Portal shared library (`extend.spaces.webapp.war`) where your extensions are either:

- Existing shared libraries containing task flows, data controls, or managed beans that you have registered in `weblogic.xml`.

  See Using Additional Shared Libraries with WebCenter Portal

- An ADF Library JAR created from the PortalExtension project and included in the WebCenter Portal shared library.

  See Developing ADF Library Components for WebCenter Portal Using the PortalExtension Project

This section includes the following topics:

- Deploying Extensions Directly to the Portal Server

- Deploying Extensions to an Archive

- Reverting to a Previous WebCenter Portal Shared Library Version

20.2.5.1 Deploying Extensions Directly to the Portal Server

Before deploying extensions to WebCenter Portal from JDeveloper:

- Verify that you have at least the WebLogic `Monitor` role.

  For information on roles, see Users, Groups, And Security Roles in *Securing Resources Using Roles and Policies for Oracle WebLogic Server*.

- Ensure a connection exists to the WebCenter Portal server.

  For information about creating a connection see Creating a WebCenter Portal Server Connection.

- Review background information on shared libraries in Creating Shared Java EE Libraries and Optional Packages in *Developing Applications for Oracle WebLogic Server*.

To deploy extensions for WebCenter Portal from JDeveloper:

1. In JDeveloper, open the workspace containing extensions to be deployed.

See also Using Additional Shared Libraries with WebCenter Portal and Developing ADF Library Components for WebCenter Portal Using the PortalExtension Project.

2. In the Application Navigator, navigate to the `PortalSharedLibrary >` `Application Sources > META-INF` folder.

3. Open the `MANIFEST.MF` file in a text editor.

4. Increment the `Implementation-Version` number. The default version is
`Implementation-Version: 12.2.1.0.1`.

Note:

You must increment the shared library `Implementation-Version` number
each time you update and redeploy the WebCenter Portal shared library
`extend.spaces.webapp.war`. Otherwise, an error is reported from
WebLogic Server in the Deployment - Log tab. For more information, see The
PortalSharedLibrary Project.

---

5. In the Application Navigator, right-click the **PortalSharedLibrary** project, select
Deploy, and then select **extend.spaces.webapp** (Figure 20-14).

Note:

Note that `extend.spaces.webapp` is the name of the deployment profile for
the shared library that you are deploying to the server. The shared library
created by a subsequent deployment action is called
`extend.spaces.webapp.war`.

Figure 20-14   Selecting the Shared Library extend.spaces.webapp
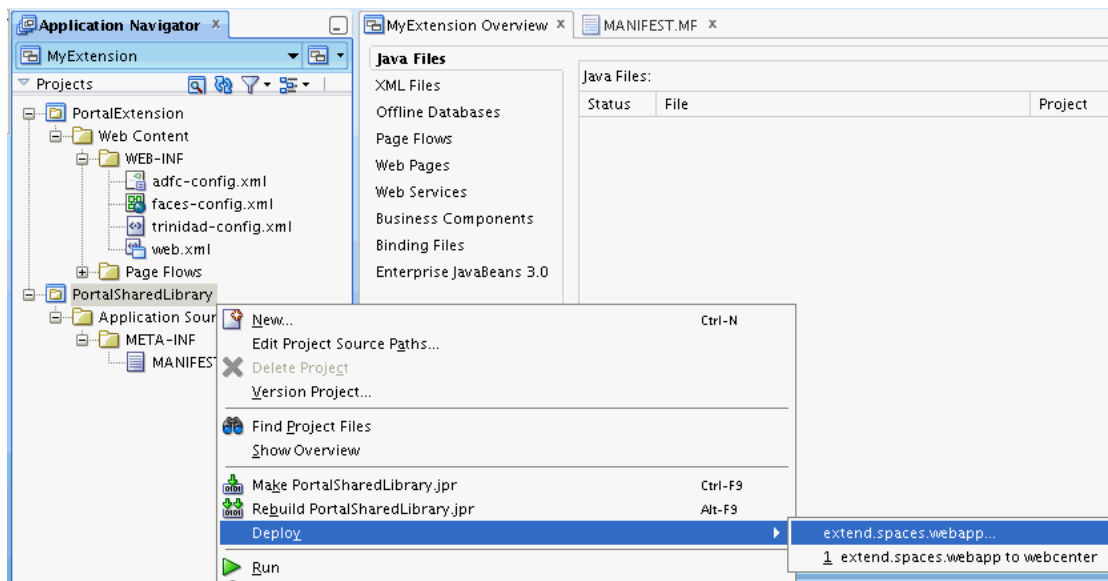
6. In the Deploy dialog, select **Deploy to Application Server** and click **Next** (Figure
20-15).

Figure 20-15   Portal Extension Deployment Actions
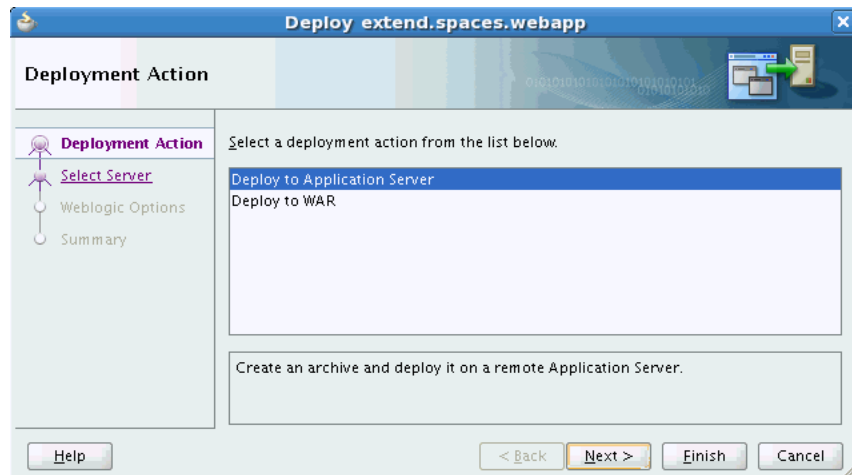
7. Select the connection that points to the WebLogic Server where you want to deploy the shared library, for example, `WC_Portal`, and click **Next**.

8. In the Deploy extend.spaces.webapp dialog, select **Deploy to selected instances in the domain**, as shown in Figure 20-16.

Figure 20-16   Selecting Deployment Option
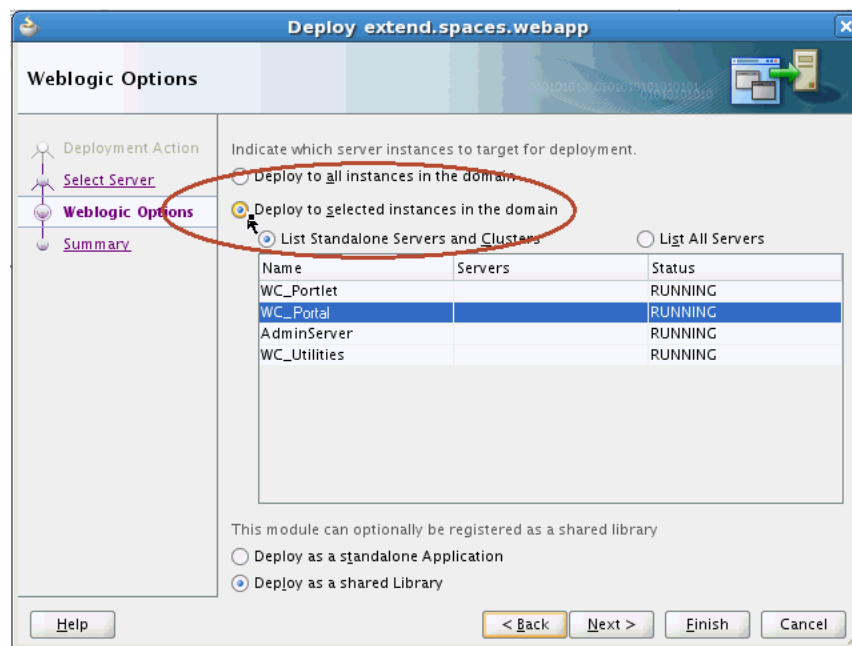
9. In the list of servers, select the managed server on which WebCenter Portal is deployed, as shown in Figure 20-17.

Note:

The name of the managed server depends on how the system administrator set up the server. By default, the name is `WC_Portal`; however, this name might be different on your system.

Figure 20-17 Selecting the Managed Server
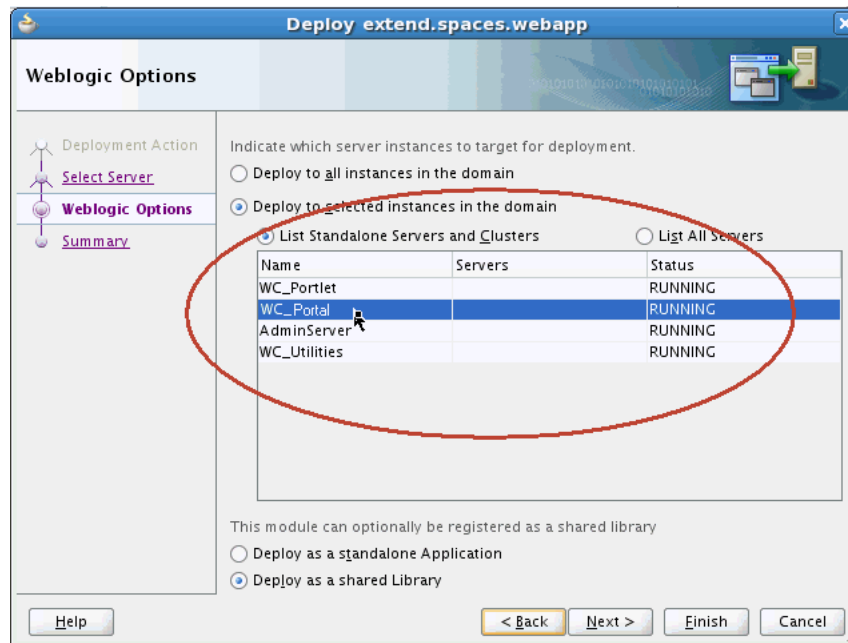
10. Select **Deploy as a shared library**, as shown in Figure 20-18

> **Note:**
>
> This step is critical: you *must* select **Deploy as a shared library**, otherwise, the deployment will not be successful.

Figure 20-18 Selecting Deploy As Shared Library
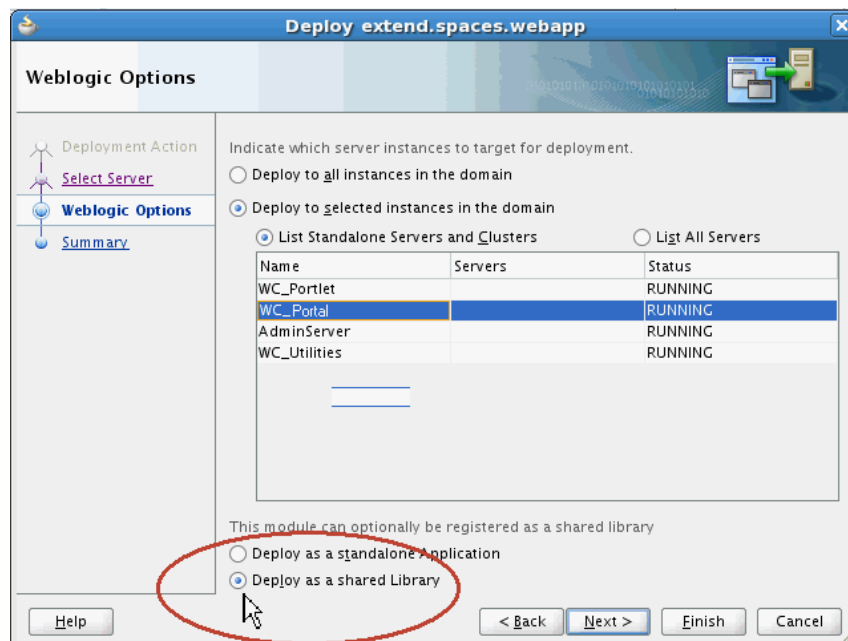
11. Click **Next**.

12. On the Deployment Summary screen, verify the deployment options, and click **Finish** to deploy the portal extension to the server.

13. Open the Deployment -Log tab to examine for any deployment issues. If deployment is successful, you should see this log entry: "`Application Deployed Successfully,`" as shown in Figure 20-19.

Figure 20-19   Deployment - Log Tab

14. To verify the new deployment, log in to the WebLogic Server Administration Console, navigate to the Deployments Overview page, and check the implementation version displayed.

    a. Log in to the WebLogic Server Administration Console.

    b. Click **Deployments**, and locate **extend.spaces.webapp**.

    c. Note the entries for the `extend.spaces.webapp` shared library and verify the one with the implementation version that you updated previously in the `MANIFEST.MF` file. See also Versioning extend.spaces.webapp.war.

Note:

WebLogic Server only uses the latest shared library version when an application starts up. If you go through several "change-build-deploy-test" iterations, incremental versions are retained by default. You can use the Administration Console to remove unwanted shared library versions if you want, but it is recommended that you retain the first version (`extend.spaces.webapp(12.2.1,12.2.1)`) as a backup so you can revert to the default behavior if necessary.

---

15. (Optional) Create any connections that your custom ADF Library components may require:

- To add a database connection, refer to Creating and Managing JDBC Data Sources in *Administering Oracle Fusion Middleware*.

- To add a web service connection, configure the ADFConnections MBean using the System MBean Browser in Fusion Middleware Control. Refer to System MBean Browser in *Administering Oracle WebCenter Portal* and Web Service Connection in *Administering Oracle ADF Applications*.

- To add a connection type specific to WebCenter Portal, for example to a portlet producer or content repository connection, refer to Managing Tools and Services in *Administering Oracle WebCenter Portal*.

16. Restart the managed server on which WebCenter Portal is running.

This step is required for the task flows, data controls, and managed beans in the shared library to show up in the WebCenter Portal Resource Registry. For more

information, see Starting and Stopping Managed Servers for WebCenter Portal Application Deployments in *Administering Oracle WebCenter Portal*.

> **Note:**
>
> The name of the managed server depends on how the system administrator set up the server. By default, the name is `WC_Portal`; however, this name might be different on your system.

17. Log in to WebCenter Portal and verify that all newly deployed task flows, data controls, or managed beans are available in the Resource Registry.

Once successfully deployed components are added to the Resource Registry, you can add them to resource catalogs where they are made available for drag and drop onto portal pages. See also, About the Resource Registry in *Building Portals with Oracle WebCenter Portal*.

20.2.5.2 Deploying Extensions to an Archive

If you do not have a connection to the portal server or permissions to deploy the `extend.spaces.webapp` shared library on the portal server, you can deploy extensions to the `extend.spaces.webapp` shared library to an archive (`.war` file) and give the `.war` file to an administrator for deployment to a managed server running WebCenter Portal.

To deploy extensions to the `extend.spaces.webapp` shared library to an archive:

1. In JDeveloper, open the workspace containing extensions to be deployed.

See also Using Additional Shared Libraries with WebCenter Portal and Developing ADF Library Components for WebCenter Portal Using the PortalExtension Project.

2. In the Application Navigator, navigate to the **PortalSharedLibrary** > **Application Sources** > **META-INF** folder.

3. Open the `MANIFEST.MF` file in a text editor.

4. If required, increment the `Implementation-Version` number. The default version is **Implementation-Version: 12.2.1.0.1**.

> **Note:**
>
> You do not need to increment the version number every time you generate a new archive (`.war` file). You must however, increment the shared library `Implementation-Version` number each time you deploy a new extension version *to the server* or an error is reported from WebLogic Server in the Deployment - Log tab. For more information, see The PortalSharedLibrary Project.

5. In the Application Navigator, right-click the **PortalSharedLibrary** project, select **Deploy** and then select the deployment profile **extend.spaces.webapp**.

> **Note:**
>
> Note that `extend.spaces.webapp` is the name of the deployment profile for the shared library that you are deploying to a file. The shared library created by a subsequent deployment action is called `extend.spaces.webapp.war`.

6. In the Deploy dialog, select **Deploy to WAR,** and then click **Next** (Figure 20-20).

Figure 20-20   Deployment Actions
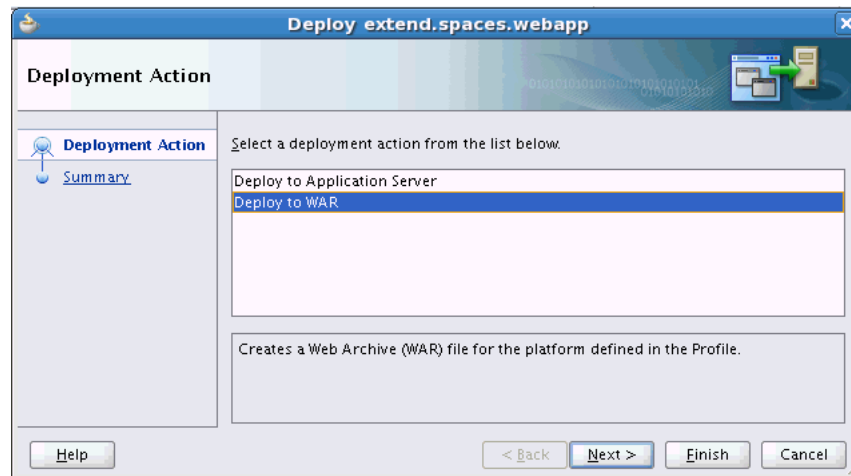
7. Review the Summary page and click **Finish**.

   The location of the `.war` file displays in the "Deployment - Log Tab." This new shared library `.war` file version can now be deployed to any server instance on which WebCenter Portal is deployed.

For More Information

See "Deploying Applications" in JDeveloper Online Help. See also Creating Shared Java EE Libraries and Optional Packages in *Developing Applications for Oracle WebLogic Server*.

20.2.5.3 Reverting to a Previous WebCenter Portal Shared Library Version

If there is a problem with the latest WebCenter Portal shared library or you want to revert to a previous version for some reason, you can undeploy (remove) the current version and revert to the previous version, using the WebLogic Server Administration Console.

You can remove unwanted shared library versions too. If you go through several "change-build-deploy-test" iterations, each incremental version is retained by default. As WebCenter Portal only uses the latest shared library version you can clean up or delete previous versions if you want.

Before undeploying the latest version, you must shut down the managed server on which WebCenter Portal is running. Once you have removed the latest version, you can restart the managed server.

Note: Oracle recommends that you do not delete the original `extend.spaces.webapp` shared library (version 12.2.1) as this enables you to revert to the out-the-box version if necessary.

1. Stop the managed server on which WebCenter Portal is deployed.

   For details, see Starting and Stopping Managed Servers for WebCenter Portal Application Deployments in *Administering Oracle WebCenter Portal*.

2. Log in to the Weblogic Server Administration Console.

3. Click **Deployments**, then locate and select the latest **extend.spaces.webapp** version you want to remove.

4. Click **Delete** to undeploy the latest shared library version.

   To revert to the *original* `extend.spaces.webapp.war` shared library, delete all other shared library versions, except for `extend.spaces.webapp.war` version 12.2.1.

5. Start the managed server on which WebCenter Portal is deployed:

   For details, see Starting and Stopping Managed Servers for WebCenter Portal Application Deployments in *Administering Oracle WebCenter Portal*.

20.2.6 Developing ADF Library Components for WebCenter Portal Using the PortalExtension Project

Developing an ADF Library for WebCenter Portal is exactly the same as any other ADF Library. If you are not familiar with ADF library development, refer to the documentation links in General Documentation for ADF Library Development.

Development teams typically build and deploy ADF libraries to a set of well defined shared libraries that they can reuse across multiple applications, such as WebCenter Portal.

To help you get started, Oracle provides the *PortalExtension* project in which you can build ADF libraries containing task flows, data controls, and managed beans and deploy them to WebCenter Portal's own shared library `extend.spaces.webapp.war`.

If you want to use the PortalExtension project, follow these steps:

1. Create a WebCenter Portal Server Extension workspace.

   For details, see Creating a WebCenter Portal Server Extension Workspace.

2. Open the **PortalExtension** project and add the custom code for your ADF library.

   See also The PortalExtension Project.

3. Deploy the ADF Library to the WebCenter Portal shared library `extend.spaces.webapp.war`.

   See also Deploying Extensions to the WebCenter Portal Shared Library (extend.spaces.webapp.war).

21

Localizing Portals

This chapter describes how to configure your application to display text in the correct language of a user's browser.

This chapter includes the following topics:

- Guidelines for Building Multilanguage Portals

- Language Support in ADF Faces Components

- Using Resource Bundles to Support Multiple Languages

- Adding Support for a New Language

21.1 Guidelines for Building Multilanguage Portals

If your application will be viewed by users in multiple countries, you can configure your JSF page or application to use different locales so that it displays the correct language for the language setting of a user's browser. For example, if you know your page will be viewed in Italy, you can localize your page so that when a user's browser is set to use the Italian language, text strings in the browser page appear in Italian.

Additionally, locale selection applies special formatting considerations applicable to the selected locale. For example, whether information is typically viewed from left to right or right to left, how numbers are depicted (such as monetary information), and the like.

When you develop a multi-language portal:

- Make use of the inherent multi-language support in Oracle WebCenter Portal tools and services. For more information, see Language Support in ADF Faces Components.

- Put all resources in resource bundles, provide alternate-language resource bundles, and code your application to respond to users' language selection. For more information, see Using Resource Bundles to Support Multiple Languages.

- Make sure your database character set supports all required languages.

  For more information, see Choosing a Character Set in *Oracle Database Globalization Support Guide*.

- Use number, date, and time formatting that supports all required locales.

  For more information, see Setting Up a Globalization Support Environment in *Oracle Database Globalization Support Guide*.

- Use linguistic sort parameters to get the correct sort ordering for locale.

For example, the same characters have different sort ordering in French and Canadian-French.

For more information, see Linguistic Sorting and Matching in *Oracle Database Globalization Support Guide*.

*   Write web pages that correctly render left to right or right to left (for example, if Arabic languages must be supported).

For more guidelines on globalization, see Internationalizing and Localizing Pages in *Developing Web User Interfaces with Oracle ADF Faces* or refer to Oracle Database Globalization Support Guide.

21.2 Language Support in ADF Faces Components

Some ADF Faces components include text that is part of the component, for example the `af:table` component uses the resource string `af_table.LABEL_FETCHING` for the message text that is displayed in the browser while the table is fetching data during the initial load of data or while the table is being scrolled. Oracle JDeveloper provides translations of these text resources into 28 languages. Therefore, when using the supplied components, you do not need to perform additional steps to translate the text in the components.

21.3 Using Resource Bundles to Support Multiple Languages

For any text you add to a component, for example if you define the label of an `af:commandButton` component by setting the text attribute, you must provide a resource bundle that holds the actual text, create a version of the resource bundle for each locale, and add a `<locale-config>` element to define default and support locales in the application's faces-config.xml file. You must also add a `<resource-bundle>` element to your application's faces-config.xml file to make the resource bundles available to all the pages in your application. Once you have configured and registered a resource bundle, the Expression Language (EL) editor displays the key from the bundle, making it easier to reference the bundle in application pages.

To simplify the process of creating text resources for text you add to ADF components, JDeveloper supports automatic resource bundle synchronization for any translatable string in the visual editor. When you edit components directly in the visual editor or in the Property Inspector, text resources are automatically created in the base resource bundle.

For more information about automatic resource bundles, see Using Automatic Resource Bundle Integration in JDeveloper in *Developing Web User Interfaces with Oracle ADF Faces*.

For more information about manually defining resource bundles, see Manually Defining Resource Bundles and Locales in *Developing Web User Interfaces with Oracle ADF Faces*.

21.4 Adding Support for a New Language

Oracle WebCenter Portal provides run-time translations for 28 languages and 100 different locales. If you want to support a language that is not supported out-of-the-box, you need to provide string files for the new language, and add a `<language>` tag for the new language in the `supported-languages.xml` configuration file.

For information about adding support for a new language, see the "Using WebCenter Spaces Extension Samples" white paper on the *Oracle WebCenter Portal White Papers and Technical Notes* page on Oracle Technology Network.

22

Extending Oracle Composer

This chapter describes extensions for WebCenter Portal that can be made to Oracle Composer (referred to as the *page editor* in the runtime) using JDeveloper.

This chapter includes the following topic:

- Adding Custom Actions to Components

22.1 Adding Custom Actions to Components

To augment the built-in actions available to users when editing components in a portal, you can add custom actions to a component.

Figure 22-1   Actions Menu on a Component
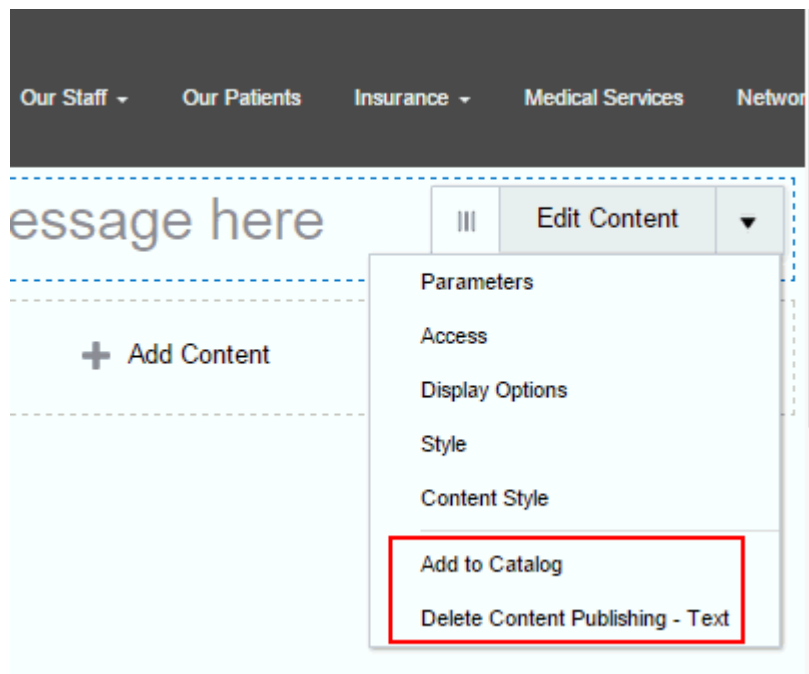
There are three types of custom actions that you can add to a component:

- **Java-based** custom actions are the most flexible choice, as they work for any component, and can implement any business logic. See:

    - Adding a Java-Based Custom Action to a Show Detail Frame Component in Design View

    - Adding a Java-Based Direct Select Custom Action to a Component in Select View

- **ID driven** custom actions. See:

    –

- **Task flow action driven** custom actions are only for task flow components, and the action is navigation for the task flow. See:

    –

    –

22.1.1 Adding a Java-Based Custom Action to a Show Detail Frame Component in Design View

You can add a custom action that displays in an **Actions** menu in the chrome of a `Show Detail Frame` component in Design View in Composer. The custom action is a Java class that implements interface `CustomActionListener`. When the action is selected from the menu, the method `processAction` performs the indicated operation on the component (such as opening a dialog, for example).

To add a custom action to a `Show Detail Frame` component in Composer's Design view:

1. Implement interface `CustomActionListener`, using method `processAction()` to implement the custom action.

   Method `processAction` takes class `CustomActionEvent` as parameter. The public methods in `CustomActionEvent` are:

   - `getPagePath`. Returns the path of the page that the component is on.

   - `getComponent`. Returns the selected component from Design view.

   - `setMessageSeverity`. Sets the severity level for `FacesMessage`.

   - `setShowMessage`. Sets the flag to show message after processing the custom action.

   - `setMessageSummary`. Sets the message summary.

   - `setMessageDetail`. Sets the message detail.

   For example:

```
package custom
public class SaveTaskflowToCatalog implements CustomActionListener
{
  public void processAction(CustomActionEvent event)
  {
     // get the the page the component is on
     String pagePath = event.getPagePath();
     // get the SDF component
     UIComponent sdf = event.getComponent();
     try
     {
         // process the task flow and save to the cata log

         // set the error message in case exception occurs
         event.setErrorSummary("Failed to save the task flow to the resource
catalog");
```

```
        }
        catch(AbortProcessingException ae)
      {
        throw ae;
      }
      catch(Exception ee)
      {
        throw new AbortProcessingException("unexpected error");
      }
    }
  }
}
```

2. Configure the custom action in `pe_ext.xml`. For example:

```
<event-handlers>
   <event-handler event="custom-action">custom.SaveTaskflowToCatalog</event-
handler>
</event-handlers>
```

3. To configure a custom action for an *individual* Show Detail Frame component on a page, **<where does this code go?>**. For example:

```
<cust:showDetailFrame shortDesc="Mac Rumors " displayHeader="true"
               id="showDetailFrame1"  text="Mac Romors">
  <af:region value="#{bindings.customactionstaskflowdefinition1.regionModel}"=
                         id="r1"/>
   <cust:customAction location="menu" action="custom.SaveTaskflowToCatalog"
                   id="ca1"  text="Save to Catalog"/>
</cust:showDetailFrame>
```

4. To configure a custom action for all Show Detail Frame components, add the custom action definition in `adf-config.xml`. For example:

```
<cust:adf-config-child xmlns="http://xmlns.oracle.com/adf/faces/customizable/
config">
   <enableSecurity value="true" />
     <customActions>
       <cust:customAction action="custom.SaveTaskflowToCatalog"
                     displayName="Save to Catalog"
                     location="menu"
                     rendered="#{custom.util.isSaveToCatalogEnabled}"/>
     </customActions>
```

22.1.2 Adding a Java-Based Direct Select Custom Action to a Component in Select View

You can add a direct select custom action to a component that displays in a popup menu when the component is selected in Select view in Composer. The custom action is a Java class that implements interface SelectActionListener. When the action is selected from the menu, the method processAction performs the indicated operation on the component (such as opening a dialog, for example).

To add a direct select custom action to a component in Composer's Select view:

1. Implement interface SelectActionListener, using method processAction() to implement the custom action.

   Method processAction takes class SelectActionEvent as parameter. The public methods in CustomActionEvent are:

- `getPagePath`. Returns the path of the page that the component is on.

- `getComponent`. Returns the selected component from Select view.

- `setMessageSeverity`. Sets the severity level for `FacesMessage`.

- `setShowMessage`. Sets the flag to show message after processing the custom action.

- `setMessageSummary`. Sets the message summary.

- `setMessageDetail`. Sets the message detail.

For example:

```
package custom
public class RemoveSelectedComponent implements SelectActionListener
{
  public void processAction(SelectActionEvent event)
  {
     // get the the page the component is on
     String pagePath = event.getPagePath();
     // get the selected component
     UIComponent comp = event.getComponent();
     try
     {
         // set the error message in case exception occurs
         event.setErrorSummary("Failed to delete the component");

         // popup deletion confirmation dialog or delete the component here
     }
     catch(AbortProcessingException ae)
     {
       throw ae;
     }
     catch(Exception ee)
     {
       throw new AbortProcessingException("unexpected error");
     }
  }
}
```

2. Optionally, sequence the menu items using the `seq` attribute set to `first`, `last`, or an integer representing an item's position on the menu.

   If no `seq` attribute is specified, actions are positioned in the order defined in the configuration.

   The actions with a `seq` value will be listed before those that do not have `seq` value. If a property panel is configured, the built-in actions `Edit Component` and `Edit Parent Component` will always be listed before other custom actions.

   For example:

```
<selection-config>
  <selection-taglib-filter namespace="http://xmlns.oracle.com/adf/faces/rich">
        <tag name="inputText">
      <selection view="design" enabled="true"/>
      <operation name="view.SelectActionText2"
                 label="InputText Action2"
                 seq="2"
                 filtered="false"/>
      <operation name="view.SelectActionTest3"
```

```
                        label="InputText Action3"
                        seq="3"
                        filtered="false"/>
            <operation name="view.SelectActionTest"
                        label="InputText Action1"
                        seq="first"
                        filtered="false"/>
            <operation name="separator" seq="1" filtered="false"/>
          </tag>
        </selection-taglib-filter>
    </selection-config>
```

3. Optionally, add separators to the custom actions menu by setting the operation name to `separator` in the selection configuration. For example:

```
<selection-taglib-filter namespace="http://xmlns.oracle.com/adf/faces/rich">
  <tag name="goLink">
    <selection view="design" enabled="true"/>
    <operation name="oracle.adf.pageeditor.pane.inline-style-editor"
                  label="Inline Style"
                  filtered="false"/>
    <operation name="separator" filtered="false"/>
    <operation name="custom.action1" label="Action 1" filtered="false"
icon="icon.png"/>
    <operation name="separator" filtered="false"/>
    <operation name="custom.action2" label="Action 2" filtered="false"/>
  </tag>
</selection-taglib-filter>
```

4. Include custom actions in Select view on an individual component, or globally for all component instances, by adding the `adf-custom-actions` operation in the select configuration for the component.

For example:

```
<selection-config>
  <global-filter>
    <selection view="design" enabled="false"/>
    <operation name="oracle.adf.pageeditor.pane.generic-property-inspector"
                label="Change Property"
                filtered="false"/>
    <operation name="oracle.adf.view.page.editor.event.SelectRemoveListener"
                label="Remove"
                icon="/adf/images/composer_delete_ena.png"
                filtered="#{testBean.compSelected ? 'false' : 'true'}"/>
  </global-filter>

  <selection-taglib-filter namespace="http://xmlns.oracle.com/adf/faces/rich">
    <tag name="goLink">
      <selection view="design" enabled="true"/>
      <operation name="oracle.adf.pageeditor.pane.inline-style-editor"
                  label="Inline Style"
                  filtered="false"/>
      <operation name="view.SelectActionTest"
                  label="#{testBean.goLinkActionLabel}1"
                  seq="last"
                  filtered="false"/>
      <operation name="view.SelectActionTest2"
                  label="#{testBean.goLinkActionLabel}2"
                  filtered="false"/>
      <operation name="separator" seq="first" filtered="false"/>
```

```
      </tag>
    </selection-taglib-filter>

  </selection-config>
```

5. In `pe_ext.xml`, configure the custom action.

For example:

```
<selection-taglib-filter namespace="http://xmlns.oracle.com/adf/faces/rich">
  <tag name="inputText">
      <selection view="design" enabled="true"/>
      <operation name="oracle.adf.pageeditor.pane.generic-property-inspector"
                            label="Change Property"
                            filtered="false"/>
      <operation name="custom.RemoveSelectedComponent"
                            label="Remove"
                            filtered="false"/>
  </tag>
</selection-taglib-filter>
```

6. In `pe_ext.xml`, register the event handler.

For example:

```
<event-handlers>
  <event-handler event="select-action">
      oracle.adf.view.page.editor.event.SelectRemoveListener</event-handler>
</event-handlers>
```

22.1.3 Adding Custom Actions to a Show Detail Frame Component by Using Facets

You can use the `Show Detail Frame` component's 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.

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:

1. Right-click a `Show Detail Frame` component in the Structure window, and select **Facets - Show Detail Frame**, and click the arrow displayed to the right of this option.

2. 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.

3. 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 Example: Adding a Custom Action to a Show Detail Frame Component by Using Facets.

22.1.3.1 Example: Adding a Custom Action to a Show Detail Frame Component by Using Facets

Assume a JSF page, `Page1.jspx`, 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:

- How to Create an ADF Task Flow

- How to Include an Additional Actions Facet

- How to Create a Redirection Page

- How to Create Navigation Rules Between Pages

- What Happens at Runtime

22.1.3.1.1 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 shown 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.

22.1.3.1.2 How to Include an Additional Actions Facet

To include an Additional Actions facet:

1. Create a JSF 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 22-2.

   Figure 22-2   Page1.jspx in Structure Navigator
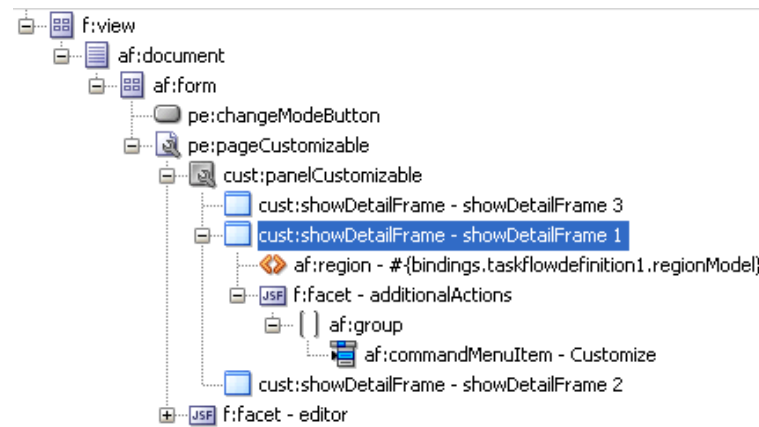
8. Save the page.

22.1.3.1.3 How to Create a Redirection Page

To create the redirection page:

1. Create a JSF 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.

22.1.3.1.4 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** icon in the Managed Beans 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.

22.1.3.1.5 What Happens at Runtime

In JDeveloper, run `Page1.jspx`. The **Actions** menu on the `showDetailFrame1` component displays the **Customize** action, as shown in Figure 22-3.

Figure 22-3   Customize Action on the Actions Menu
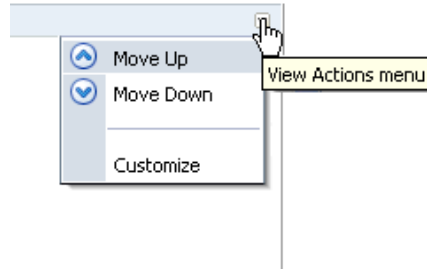
Clicking **Customize** takes you to `Page2.jspx` (Figure 22-4), where you can update the values for `Label1` and `Label2`.

Figure 22-4   Page with Option to Edit Text
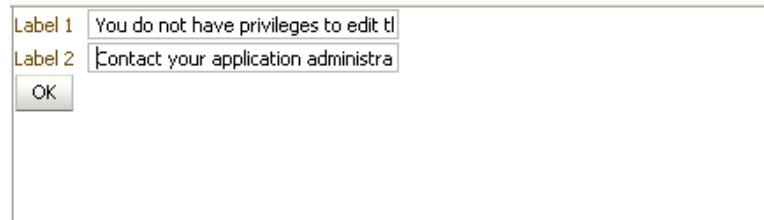
Clicking **OK** takes you back to `Page1.jspx`, which reflects the recent text changes.

22.1.4 Adding Custom Actions to a Task Flow

You can customize a task flow by including it in a `Show Detail Frame` component. The `Show Detail Frame` component provides certain default actions to rearrange, show, and hide components. Further, you can define custom actions to trigger the desired navigational flow within the task flow at runtime. There are two ways in which you can enable custom actions on a task flow:

- Enable custom actions directly on the task flow so that they are displayed on the `Show Detail Frame` component's **Actions** menu.

- Enable custom actions on the `Show Detail Frame` component enclosing the task flow so that they are displayed on the `Show Detail Frame` component's **Actions** menu.

This section describes both approaches. It contains the following subsections:

- Adding a Custom Action Directly on a Task Flow

- Adding a Custom Action on a Show Detail Frame Enclosing a Task Flow

- What Happens at Runtime

- Example: Adding a Custom Action to a Show Detail Frame Enclosing a Task Flow

22.1.4.1 Adding a Custom Action Directly on a Task Flow

Perform the steps in this section if you want the task flow to be self-contained, without the need to inherit global custom actions defined at the application level. You can define custom actions on a task flow by configuring a `<customComps-config>` section with a nested `<customActions>` element in the `adf-settings.xml` file. The additional custom actions specified in the `adf-settings.xml` file are displayed on the `Show Detail Frame` component that surrounds this task flow.

Typically, task flows are packaged and deployed as ADF libraries. When you create an `adf-settings.xml` file containing custom actions for a task flow, this file is also packaged in the ADF library.

To enable custom actions on a task flow:

1. If it does not already exist, create the `adf-settings.xml` file in the `META-INF` directory under the project's Web context root (for example, in the `APPLICATION_ROOT/Portal/src/META-INF` directory):

   a. From the **File** menu, select **New**.

   b. In the New Gallery dialog, expand **General**, select **XML**, then **XML Document**.

   c. Click **OK**.

   Name the file `adf-settings.xml`.

2. Add a `<custComps-config>` section in the file, with a nested `<customActions>` element.

3. Add one `<customAction>` element for each internal task flow action that you want to display as a custom action on the `Show Detail Frame` Actions menu.

   You can add any number of custom actions under the `<customActions>` element.

   The following example shows the code of the `adf-settings.xml` file with a `<customAction>` entry.

```
<cust:custComps-config xmlns="http://xmlns.oracle.com/adf/faces/customizable/
config">
  <customActions>
   <customAction action="next" location="chrome"
                 rendered="true"
                 icon="/adf/webcenter/editheader_ena.png"
                 text="Next"
                  taskFlowId="/WEB-INF/task-flow-definition.xml#task-flow-
definition"
                 shortDesc="next"/>
    <customAction action="prev" location="chrome"
                 rendered="true"
                 icon="/adf/webcenter/editheader_ena.png"
                 text="Previous"
 taskFlowId="/WEB-INF/task-flow-definition.xml#task-flow-definition"
                 shortDesc="prev"/>
  </customActions>
 </cust:custComps-config>
```

   Custom action definitions are similar at the task flow-level and application-level, except for the `taskFlowId` attribute that is used in the task flow-level setting.

This attribute is used to identify the task flow on which the custom action must be defined. As an ADF library may have multiple task flows, this attribute helps identify the task flow that must render the custom action.

Note:

- 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 `Show Detail Frame` displays only those custom actions that correspond to the ADFc outcomes of the current view of the task flow.

---

4. Save the `adf-settings.xml` file.

22.1.4.2 Adding a Custom Action on a Show Detail Frame Enclosing a Task Flow

You can define custom actions for a task flow on the enclosing `Show Detail Frame` component. When these actions are invoked at runtime, they trigger the desired navigational flow in the task 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`.

You can specify custom actions on `Show Detail Frame` components by adding `Custom Action` components as children of a `Show Detail Frame` component on the page. Custom actions defined in this way would be available only on the `Show Detail Frame` instance, which has the custom action as its child. Alternatively, you can specify custom actions in the application's `adf-config.xml file`. Custom actions defined in this way are available on all `Show Detail Frame` instances in the application.

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

- Defining Custom Actions at the Instance Level

- Defining Custom Actions at the Global Level

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

- Configuring Custom Actions that Display Task Flow Views in a Separate Browser Window

22.1.4.2.1 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 Composer tag library. Custom actions are stored in the JSF 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 **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` Actions menu.

> **Note:**
>
> Add `Custom Action` components below the `af:region` element. You may 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

   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 following:

```
<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:**
>
> You can add custom actions for all of the task flow's ADFc outcomes, but depending on the task flow view that is displayed, several or none of the custom actions are available at runtime.
>
> 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 a Task Flow in *Developing Fusion Web Applications with 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.

22.1.4.2.2 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 on 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 follows:

    Tip:

    To render a custom action only in page Edit mode, you can set the `rendered` attribute on the `<customAction>` tag to `#{composerContext.inEditMode}`. This returns a value of `true` if the page is in Edit mode.

    ```
    <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>
    ```

    Note:

    - If you implemented restrictions on the `Show Detail Frame` component's actions by performing the steps in ??? 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 the example.

    - 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 Example: Adding a Custom Action to a Show Detail Frame Enclosing a Task Flow.

22.1.4.2.2.1 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 ???

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.

22.1.4.2.3 Configuring 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, you can define a custom action to display a 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"/>
```

Setting Properties on the Popup Window

For a command component inside a task flow region, you can specify the default behavior using the `useWindow`, `windowEmbedStyle`, `windowHeight`, `windowWidth`, and `returnListener` attributes. The command component may have other such attributes that you can use. If you specify a return listener, on closing the dialog a particular action event is called to decide on the next navigation outcome. By default, without this setting, a dummy `Rich Command Link` component is created to trigger the task flow action.

When you define a listener on a command component, you must also configure the custom action to call the action event on this component. The `actionComponent` attribute on a custom action definition (global- and instance-level) enables you to specify the ID of the command component that must be queued for the action event. When the `actionComponent` attribute is specified, the `Show Detail Frame` component queues the action event on this component. Since this command component is inside your task flow, you change its attribute values at any time.

Example

Consider an example where you have included a task flow inside a `Show Detail Frame` component and defined a `Simple Edit` custom action corresponding to the task flow's navigation outcomes. A `Command Button` component inside the task flow is configured to launch a modal inline popup of size `300x200` on clicking the `Simple Edit` custom action. A return listener is configured on the command component so that it is called whenever the popup is closed.

The source code of the `Command Button` component inside the region is as follows:

```
<af:commandButton text="dialog:simpleEditPoup"
                  id="SDFCustomActionCmd_simpleEditPoup"
                  action="dialog:simpleEditPoup" useWindow="true"
                  windowEmbedStyle="inlineDocument" windowWidth="300"
                  windowHeight="200"
                  windowModalityType="applicationModal"
                  returnListener="#{pageFlowScope.recentPagesBean.refreshMainView}"
                  visible="false"/>
```

The following example shows how you can specify a global custom action corresponding to the task flow outcome by defining the custom action in the `adf-config.xml` file. The ID of the `Command Button` component is specified against the `actionComponent` attribute on the custom action.

```
<customizableComponentsSecurity xmlns="http://xmlns.oracle.com/adf/faces/
customizable/config">
  <enableSecurity value="true"/>
    <customActions>
     <customAction action="dialog:simpleEditPoup"
                   text="Simple Edit"
                   shortDesc="Simple Edit"
                   location="menu"
                   rendered="#{!changeModeBean.inEditMode}"
                   icon="/adf/pe/images/editproperties_ena.png"
                   actionComponent="SDFCustomActionCmd_simpleEditPoup"/>
    </customActions>
</customizableComponentsSecurity>
```

The following example shows how you can specify an instance-level custom action corresponding to the task flow outcome by adding a `Custom Action` component from the Composer tag library to the JSF page. The ID of the `Command Button` component is specified against the `actionComponent` attribute on the custom action.

```
<cust:showDetailFrame id="sdf_for_RecentPagesTF1"
                      text="Recent Pages" stretchContent="false"
                      showResizer="never">
  <af:region id="RecentPagesTF1"
             value="#{bindings.regionBinding1.regionModel}"/>
  <cust:customAction action="dialog:simpleEditPoup"
                     text="My Simple Edit"
                     shortDesc="Simple Edit"
                     location="menu"
                     rendered="#{!changeModeBean.inEditMode}"
                     icon="/adf/pe/images/editproperties_ena.png"
                     actionComponent="SDFCustomActionCmd_simpleEditPoup"/>
</cust:showDetailFrame>
```

> **Note:**
>
> A custom action that is launched in a popup dialog is sent as a new request to the server. If you are using a Composer sandbox and are in Edit mode of a page, ensure that this request is launched in View mode by adding code in your servlet filter so that a new sandbox is not created for this page.

22.1.4.3 What Happens at Runtime

Custom actions configured in the `adf-settings.xml` file are merged with the custom actions configured in the `adf-config.xml` file and all actions relevant to the current view of the selected task flow are displayed on the parent `Show Detail Frame` component's **Actions** menu.

If you enabled custom actions at a global level, then the header of any `Show Detail Frame` displays these custom actions if they correspond to the navigation outcomes of the current view of the child task flow.

If you prefixed the `action` attribute value with `dialog:`, then the target view of the task flow opens in a separate browser window.

22.1.4.4 Example: Adding a Custom Action to a Show Detail Frame Enclosing a Task Flow

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 so 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 in such a way 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 ???

2. Add a Custom Action component from the 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 22-5).

   Figure 22-5   Custom Actions on a Show Detail Frame Enclosing a Task Flow

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

22.1.5 Adding Custom Actions that Display on Task Flows in the Component Navigator

The component navigator in Composer Structure view provides an option to zoom into a task flow and display the components on its page or fragment, as shown in Figure 22-6.

Figure 22-6   Edit Action on a Task Flow Instance
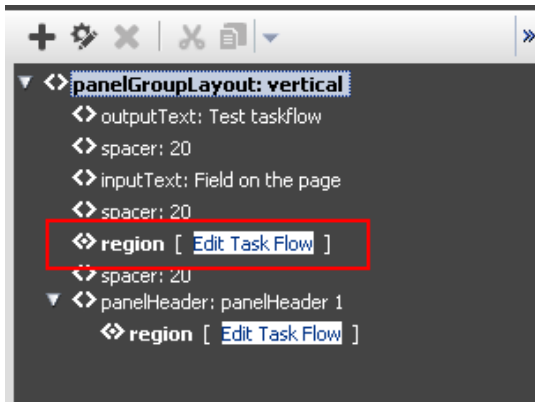
Users can zoom in, edit the page or fragment, and zoom out of the task flow to navigate back to the page containing the task flow. In addition to the Edit Task Flow and Close links displayed next to a task flow name, you can configure your application to also display custom actions next to a task flow name, as shown in Figure 22-7.

Figure 22-7   Custom Action on a Task Flow Instance
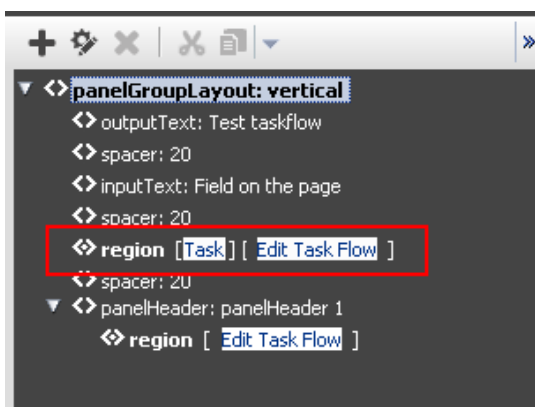
This section describes the procedure to enable custom actions on task flows in the component navigator. It contains the following sections:

- How to Configure Custom Actions in the Component Navigator

- What Happens at Runtime

22.1.5.1 How to Configure Custom Actions in the Component Navigator

To display custom actions against a task flow in the component navigator, you must create a Java bean that defines the custom action behavior and call this bean from the

application page containing the task flow. This section describes the steps to do so in detail. It contains the following subsections:

- Defining the Logic for the Custom Action

- Creating a JSF Page Containing the Custom Action

- Calling the JSF Page from the Application Page Containing the Task Flow

22.1.5.1.1 Defining the Logic for the Custom Action

To begin with, you must decide on the custom action you want to provide to users and create a Java bean with the logic to be implemented when a user selects the custom action. This section describes the steps to implement simple logic to display a message to users on clicking the custom action. The sample bean also contains the code to display the Close/Edit Task Flow links along with the custom link.

To create a Java bean:

1. 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, enter `BackingBean` in the **Name** field and click **OK**.

   The `BackingBean.java` file displays in Source view.

4. Import the required libraries as follows:

   ```
   import javax.faces.application.FacesMessage;
   import javax.faces.context.FacesContext;
   import javax.faces.event.ActionEvent;
   import oracle.adf.view.page.editor.sourceview.ComponentInfo;
   ```

5. Add the following code:

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

     public void action(ActionEvent actionEvent) {
         FacesContext context = FacesContext.getCurrentInstance();
         context.addMessage(null,
                            new FacesMessage(FacesMessage.SEVERITY_INFO, "Sample
   message to test whether the custom action works.",
                                            null));
     }

     public boolean isRendered() {
         return ComponentInfo.isRegion();
     }

     public String getZoomText() {
         return ComponentInfo.isRootNode() ? "Close" : "Edit Task Flow";
     }
   }
   ```

   With this logic, the sample message you defined is displayed when a user clicks the custom action link against a task flow region.

Use the `isRegion()` API to ensure that the custom action link is displayed only against task flow regions on the page. Use the `isRootNode()` API to ensure that the Edit Task Flow or Close link is displayed against the root component of the task flow.

6. Save the bean.

22.1.5.1.2 Creating a JSF Page Containing the Custom Action

This section describes the procedure to create a JSF page containing the custom action that you want to display to users in Composer.

To create the JSF page:

1. In your application project, create a JSF page called `customList.jspx`:

   a. From the **File** menu, select **New**.

   b. In the New Gallery dialog, expand **Web Tier**, select **JSF**, then **JSF Page**.

   c. Enter a name for the page and click **OK**.

2. Design the custom action UI using components like `Output Text` and `Command Link`, as shown in the following sample page:

```
<?xml version='1.0' encoding='UTF-8'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:h="http://java.sun.com/jsf/html"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich">
  <jsp:directive.page contentType="text/html;charset=UTF-8"/>
  <af:componentDef var="attrs" componentVar="component">
    <af:panelGroupLayout id="dc_pgl1" rendered="#{backingBean.rendered}">
      <af:outputText value="[" id="dc_ot1"/>
      <af:commandLink text="Test" id="dc_cl1"
                      actionListener="#{backingBean.action}"
                      binding="#{backingBean.customLink}"/>
      <af:outputText value="]" id="dc_ot2"/>
    </af:panelGroupLayout>
  </af:componentDef>
</jsp:root>
```

This sample creates a custom action called `Test`. The `actionListener` and `binding` attributes on this action are bound to `BackingBean`, which you created earlier.

3. To ensure that the default Edit Task Flow/Close options are also displayed next to the task flow, you must define a facet called `zoom`, as shown in the following example:

```
<af:xmlContent>
  <component xmlns="http://xmlns.oracle.com/adf/faces/rich/component">
    <facet>
      <facet-name>zoom</facet-name>
    </facet>
  </component>
</af:xmlContent>
```

4. Include the `zoom` facet in the page content as shown in the following example:

```
<af:outputText value="[" id="dc_ot3"/>
<af:facetRef facetName="zoom"/>
<af:outputText value="]" id="dc_ot4"/>
```

The source of the `customLink.jspx` page is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:h="http://java.sun.com/jsf/html"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich">
  <jsp:directive.page contentType="text/html;charset=UTF-8"/>
  <af:componentDef var="attrs" componentVar="component">
    <af:panelGroupLayout id="dc_pgl1" rendered="#{backingBean.rendered}">
      <af:outputText value="[" id="dc_ot1"/>
      <af:commandLink text="Test" id="dc_cl1"
                      actionListener="#{backingBean.action}"
                      binding="#{backingBean.customLink}"/>
      <af:outputText value="]" id="dc_ot2"/>
      <af:outputText value="[" id="dc_ot3"/>
      <af:facetRef facetName="zoom"/>
      <af:outputText value="]" id="dc_ot4"/>
    </af:panelGroupLayout>
    <af:xmlContent>
      <component xmlns="http://xmlns.oracle.com/adf/faces/rich/component">
        <facet>
          <facet-name>zoom</facet-name>
        </facet>
      </component>
    </af:xmlContent>
  </af:componentDef>
</jsp:root>
```

5. Save the JSF page.

22.1.5.1.3 Calling the JSF Page from the Application Page Containing the Task Flow

Let us assume that you have a simple JSF page, `MyPage.jspx`, containing a task flow. The source of the page is as shown in the following example:

```
<?xml version='1.0' encoding='UTF-8'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:h="http://java.sun.com/jsf/html"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
          xmlns:pe="http://xmlns.oracle.com/adf/pageeditor"
          xmlns:cust="http://xmlns.oracle.com/adf/faces/customizable">
  <jsp:directive.page contentType="text/html;charset=UTF-8"/>
  <f:view>
    <af:document id="d1">
      <af:form id="f1">
        <af:panelStretchLayout topHeight="50px" id="psl1">
          <f:facet name="top">
            <pe:changeModeLink id="cml1"/>
          </f:facet>
          <f:facet name="center">
            <!-- id="af_one_column_header_stretched"  -->
            <pe:pageCustomizable id="pageCustomizable1">
              <cust:panelCustomizable id="panelCustomizable1" layout="scroll">
                <af:region value="#{bindings.taskflowdefinition1.regionModel}"
                           id="r1"/>
              </cust:panelCustomizable>
```

```
                          <f:facet name="editor">
                             <pe:pageEditorPanel id="pep1"/>
                          </f:facet>
                      </pe:pageCustomizable>
                   </f:facet>
               </af:panelStretchLayout>
           </af:form>
       </af:document>
   </f:view>
</jsp:root>
```

The task flow, `taskflowdefinition1`, contains the following `view.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 layout="scroll" id="pgl1">
    <af:commandButton text="commandButton 1" id="cb1"/>
    <af:commandButton text="commandButton 2" id="cb2"/>
  </af:panelGroupLayout>
</jsp:root>
```

To display the custom action that you created, you must ensure that the `customLink.jspx` page is called from within Composer. Use the `sourceViewNodeAction` attribute on the `Page Customizable` component to reference the JSF page containing the custom action.

The `Page Customizable` tag appears as follows in the page source:

```
<pe:pageCustomizable id="pageCustomizable1"
                     sourceViewNodeAction="/customLink.jspx">
```

The `sourceViewNodeAction` attribute can take the name of a JSF (`.jspx`) file or an EL value that evaluates to a JSF (`.jspx`) file name.

22.1.5.2 What Happens at Runtime

When you run `MyPage.jspx` to the browser and open the page in Composer Structure view, the component navigator displays a Test link and an Edit Task Flow link next to each task flow instance on the page

23

Customizing WebCenter Portal Impersonation Security

This chapter describes how to use a set of WebCenter Portal Impersonation Expression Language expressions and a matching Impersonation API to customize impersonation sessions.

This chapter includes the following topics:

- About Customizing WebCenter Portal Impersonation

- Using the WebCenter Portal Impersonation ELs

- Using the WebCenter Portal Impersonation APIs

23.1 About Customizing WebCenter Portal Impersonation

WebCenter Portal Impersonation lets designated users impersonate other portal users and perform operations as those users. You can customize those sessions using a set of WebCenter Portal Impersonation Expression Language expressions (ELs) and a matching Java API

For instructions on how to initiate an impersonation session (by the impersonator) and how to allow an Impersonation session (by the impersonatee), see Using WebCenter Portal Impersonation in *Using Oracle WebCenter Portal*.

23.2 Using the WebCenter Portal Impersonation ELs

WebCenter Portal Impersonation offers a set of Expression Language expressions (ELs) that can be used to customize impersonation sessions.

The following ELs are exposed:

- `#{WCSecurityContext.impersonationConfigured}` - returns whether or not impersonation has been enabled for the current domain

  This EL can be useful when determining if an error was caused by an impersonation session ending prematurely, or to provide an additional indicator that a session has ended.

- `#{WCSecurityContext.userInImpersonationSession}` - returns whether the current user is in an impersonation session or not.

  You can use this EL to protect content and render it inaccessible during an impersonation session. For example, you could map the rendered attribute of an administration taskflow on a page to this EL only rendering the taskflow if the user is not viewing the taskflow in an impersonation session.

- `#{WCSecurityContext.currentImpersonator}` - returns the current impersonator, if any.

  This EL could be used to modify the page template to display the impersonator or render content accessible only to a particular impersonator.

For more information about impersonation and other ELs, refer to Expression Language Expressions.

23.3 Using the WebCenter Portal Impersonation APIs

WebCenter Portal Impersonation also Java APIs that can be used to customize impersonation sessions.

The following public APIs are exposed in `oracle.webcenter.security.common.WCSecurityUtility`:

- `isImpersonationConfigured()` - returns whether or not impersonation has been enabled for the current domain.

  This API can be useful to determine if an error was caused by an impersonation session ending prematurely, or to provide an additional indicator that a session has ended.

- `isUserInImpersonationSession()` - returns whether the current user is in an impersonation session or not.

  This API is recommended for use to protect content and render it inaccessible during an impersonation session. For example, you could map the rendered attribute of an administration taskflow on a page to this API throwing an authorization exception or returning an empty list if the user is viewing the taskflow in an impersonation session.

- `getCurrentImpersonatorId()` - returns the current impersonator, if any.

  This API could be used to modify the page template to display the impersonator (as shown in the example below), or render some content accessible only to a particular impersonator.

For more information about these and other APIs, refer to *Oracle WebCenter Portal Java API Reference*.

Example: getCurrentImpersonatorId API

```
import oracle.webcenter.security.common.WCSecurityUtility;
if (WCSecurityUtility.isUserInImpersonationSession())
{
 String impersonator =WCSecurityUtility.getCurrentImpersonatorId();
 String currentUser =ADFContext.getCurrent().getSecurityContext().getUserName();
 //Code to be executed when the user is in an impersonation session.
 ..log("User " +impersonator +" is impersonating as user " +currentUser);
}
```

Part VI

Appendixes

This part of *Developing WebCenter Portal Assets and Custom Components with Oracle JDeveloper* provides appendixes with supporting information for the chapters in this guide.

- Expression Language Expressions

- WebCenter Portal Accessibility Features

- Using the WebCenter Portal REST APIs

- Troubleshooting

A

Expression Language Expressions

This appendix describes the Expression Language (EL) expressions that you can use in portals built with WebCenter Portal.

This appendix includes the following topics:

> **See Also:**
>
> For additional information about EL APIs, see Oracle WebCenter Portal Java API Reference.

A.1 Introduction to Expression Language (EL) Expressions

When configuring page components or assets, you can express values as variables that take advantage of the current application context. For example, you can use variables to obtain the name of the current user, the user's assigned role, the state of the current

page, and so on. Such flexibility is made possible by using EL expressions created in JDeveloper using the Expression Builder, or at runtime using the Expression Editor.

This section includes the following subsections:

- Introducing the Expression Builder
- Introducing the Expression Editor in WebCenter Portal

A.1.1 Introducing the Expression Builder

You can use EL expressions in JDeveloper to bind attributes to object values determined at runtime. At runtime, the value of certain components is determined by their value attribute. While a component can have static text as its value, typically the value attribute will contain an EL expression that the runtime infrastructure evaluates to determine what data to display.

You can create EL expressions using the Expression Builder in JDeveloper. You can access the builder from the Property Inspector. In the Property Inspector, locate the attribute you wish to modify and click the Property Menu icon, and select choose **Expression Builder** from the popup (Figure A-1).

Figure A-1   Accessing the Expression Builder
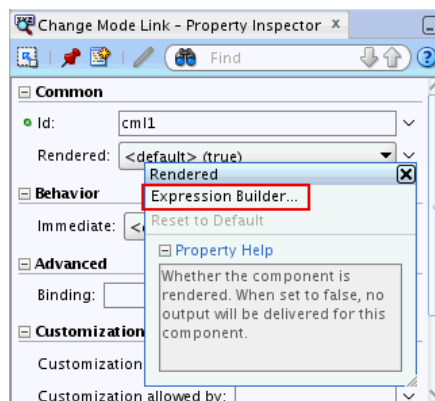
In the Expression Builder dialog (Figure A-2), you can directly type your EL expression in the Expression box. You can also use the Variables drop-down list to select items that you want to include in the expression. Use the operator buttons to add logical or mathematical operators to the expression.

For more information about Expression Builder, see Getting Started with ADF Faces in *Developing Web User Interfaces with Oracle ADF Faces*.

Figure A-2   Expression Builder Dialog
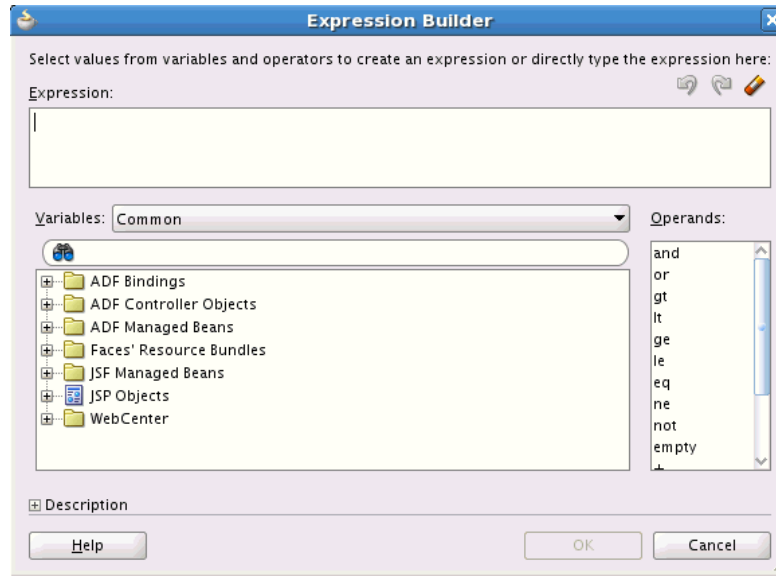
A.1.2 Introducing the Expression Editor in WebCenter Portal

WebCenter Portal provides a simple Expression Language (EL) editor, called the Expression Editor. Use the Expression Editor when you want to formulate a dynamic computation for an otherwise unknown property value, for example, to specify the current user, the current page mode, and so on.

The Expression Editor is available through the administration and editing screens in WebCenter Portal. You can open the Expression Editor by clicking the **Advanced Edit Options** icon next to a field, check box or drop-down list, and then clicking **Expression Builder** (see Figure A-3).

Figure A-3   Advanced Edit Options Icon Next to a Parameter Value Field and the Resulting Editor
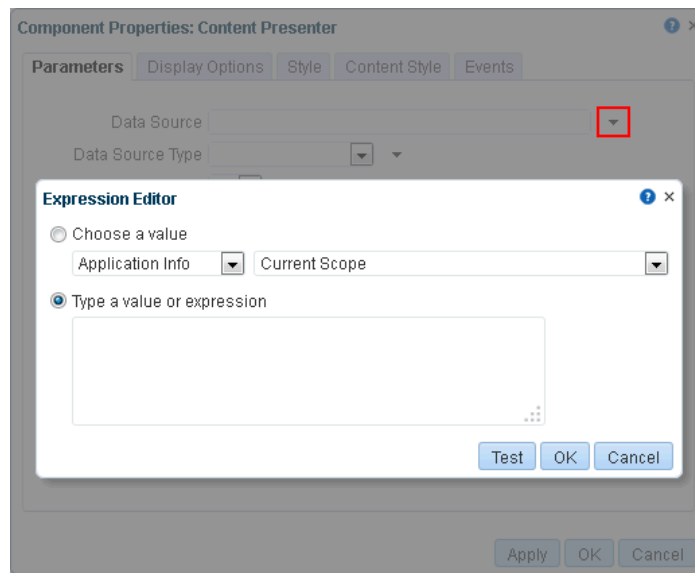
> **See Also:**
>
> For information about accessing component properties, see Modifying Components in *Building Portals with Oracle WebCenter Portal*.

The Expression Editor provides two options for entering expressions:

- **Choose a value**, for selecting a seeded EL expression

- **Type a value or expression**, for entering an expression manually

The options under **Choose a value** are categorized according to the type of information an expression returns. Select a category from the first menu, and then select the type of value you want returned from the second menu (Figure A-4).

Figure A-4   Options Under Choose a Value in the Expression Builder
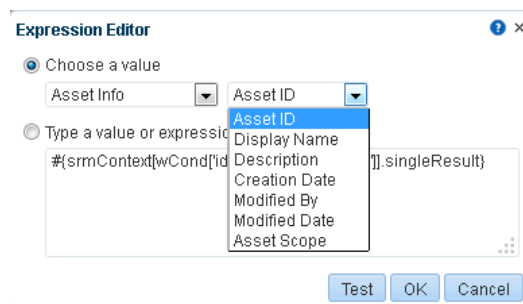
> **See Also:**
>
> For information about seeded EL expressions, see Built-In Expressions in the Expression Editor.

The option **Type a value or expression** is followed by a text box, which you can use to manually enter the expression you intend (Figure A-5).

Figure A-5   Text Box Under Type a value or expression in the Expression Editor
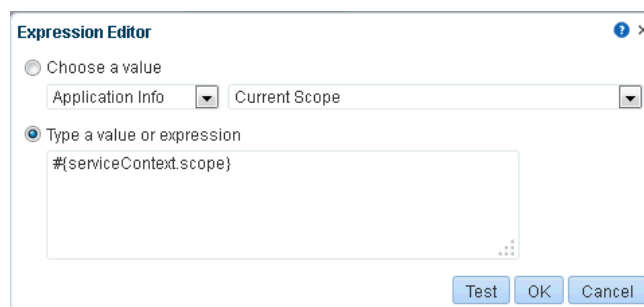
Many expressions in the tables in this appendix are building blocks for narrowing down the object or objects you want returned. That is, they are not necessarily meant to be used by themselves, but rather in an assembly of ELs to form the desired query.

For example, the following expression uses three asset-related ELs to form a single query that returns a particular portal template. It retrieves the template `storesMasterTemplate` from the parent portal `stores`:

```
#{srmContext.resourceScope['stores'].resourceType['siteTemplate'].displayName['stores
MasterTemplate'].singleResult}
```

In the Expression Editor, you can either:

- Select **Choose a value**, then select predefined values from the drop-down lists.

- Select **Type a value or expression**, then enter a value or EL expression. Use the following formats:

    - a literal number: `#{123}`

    - a literal string: `#{'string'}`

    - a literal Boolean: `#{true}`

    - a Java Bean called to return a value:
      `#{generalSettings.preferredTimeStyle}`

The editor provides a **Test** button for validating your EL entry. The **Test** button is available for expressions entered in the text box under **Type a value or expression**.

Validation checks the EL syntax and evaluates the expression. Because expression values vary according to the context in which they are executed, the resulting value that appears in the editor may differ from the value returned during actual use.

Note, however, that only an EL is validated when you click **Test**; other types of values are not validated. Test results are shown in a popup.

Note:

Wherever you enter EL on the generic **Display Options** tab in the Component Properties dialog, the entry is automatically validated. If the EL syntax is invalid, an error appears and the value is neither applied nor saved. Generic Display Options are those cataloged in the Display Options Properties table in *Building Portals with Oracle WebCenter Portal*.

EL validation is not performed on non-generic display options.

A.2 ELs Related to WebCenter Portal Information

The following table lists EL expressions relevant to WebCenter Portal information and describes the type of functionality they provide.

Table A-1   EL Expressions Relevant to WebCenter Portal Information

Expression	Function
`#{WCAppContext}`	An `oracle.webcenter.webcenter app.context.WCApplicationC ontext` object that provides an access point in the current web request for all WebCenter Portal-related information.

Table A-1 (Cont.) EL Expressions Relevant to WebCenter Portal Information

Expression	Function
`#{WCAppContext.currentWebCenterURI}`	Returns a URL representing the current web request with bookmarkable WebCenter Portal URL parameters of the request appended to the end (parameters are not necessarily in a fixed order). Example: `http://www.example.com/ webcenter/faces/oracle/ webcenter/page/scopedMD/ someguid/SomePage.jspx? wc.contextURL=/spaces/ somename&wc.pageScope=1234`
`#{WCAppContext.application.applicationCon fig}`	An `oracle.webcenter.webcenter app.beans.WebCenterType` bean with a payload of metadata from WebCenter Portal.
`#{WCAppContext.application.applicationCon fig.title}`	Returns the display name of the current WebCenter Portal application (as configured through WebCenter Portal Administration settings). Out of the box, this returns *WebCenter Portal*.
`#{WCAppContext.application.applicationCon fig.logo}`	If an application logo was uploaded through WebCenter Portal Administration settings, this expression returns the URL to the application logo image. Out of the box, this returns *null*.
`#{WCAppContext.application.applicationCon fig.helpPage}`	Returns the URL to the Help application used for WebCenter Portal (as configured through WebCenter Portal Administration settings). Out of the box, this returns */ webcenterhelp/spaces*. <<**REVIEWER**, should it be /portal instead of /spaces?>>
`#{WCAppContext.application.applicationCon fig.copyrightMessage.customValue}`	If a copyright message was configured through WebCenter Portal Administration settings, this expression returns the application copyright message. Out of the box, this returns `null`.

Table A-1   (Cont.) EL Expressions Relevant to WebCenter Portal Information

Expression	Function
`#{WCAppContext.application.applicationConfig.privacyPolicyUrl}`	Returns the URL to the document that contains the application's privacy policy (as configured through WebCenter Portal Administration settings). Out of the box, this returns `http://www.oracle.com/html/privacy.html`
`#{WCAppContext.application.applicationConfig.skin}`	Returns the name of the default ADF Faces skin family to use for rendering pages in the application (as configured through WebCenter Portal Administration settings). This expression represents only the application-level setting that may not necessarily be used in all web requests. For example, you cannot use it successfully if a user has chosen to override the skin through application Preferences.
`#{requestContext.skinFamily}`	Returns the name of the ADF Faces skin family being used for the current web request, depending on factors such as what has been configured at the application level, the current user's preference setting, and so on. Returns the same value as `#{adfFacesContext.skinFamily}`.

A.3 ELs Related to Specific Pages

Table A-2 lists EL expressions relevant to portal pages and describes the types of functionality they provide.

Table A-2   EL Expressions Relevant to Specific Pages

Expression	Function
`#{pageDocBean.title}`	Returns the page display name, for example: *FinanceProject*
`#{pageDocBean.createdBy}`	Returns the user name of the person who created the page, for example: *monty*
`#{pageDocBean.createDateString}`	Returns the date and time the page was created, for example: *2008-11-19T10:18:36*

Table A-2 (Cont.) EL Expressions Relevant to Specific Pages

Expression	Function
#{pageDocBean.lastUpdatedBy}	Returns the user name of the person who last updated the page, for example: *monty*
#{pageDocBean.lastUpdateDateString}	Returns the date and time the page was last updated, for example: *2008-11-19T10:18:36*
#{pageDocBean.pagePath}	Returns the file directory path to the page relative to the application root directory, for example: */oracle/webcenter/page/scopedMD/ s8bba98ff_4cbb_40b8_beee_296c916a2 3ed/user/Umonty/Page4.jspx*
#{pageDocBean.name}	Returns the file name of the page, for example: *Page4.jspx*
#{pageDocBean.UICSSStyle}	Returns the name of the style scheme used on the page, for example: *WCSchemeEggShell*
#{pageDocBean.UISchemeBGImage}	Returns the directory path and file name of the page scheme background image.
#{pageDocBean.UISchemeBGColorString}	Returns the hexadecimal value of the page scheme background color, for example: *#ffa500*
#{pageDocBean.pagePermission}	Returns the permission the current user has on the page, for example: *oracle.webcenter.page.model.security.C ustomPagePermission*
#{pageDocBean.pageSecurityTarget}	A string of 60 or so characters that uniquely identifies the current page to the security system, for example: *oracle_webcenter_page_scopedMD_s8b ba98ff_4cbb_40b8_beee_296c916a23ed _user_Umonty_Page4PageDef*
#{changeModeBean.inEditMode}	Returns *true* if current page is in Composer mode. Returns *false* if current page is not in Composer mode
#{pageDocBean.currentLayout.displayName} .	Returns the display name of the layout currently used by the page

Table A-2   (Cont.) EL Expressions Relevant to Specific Pages

Expression	Function
`#{pageDocBean.layoutViewId}`	Returns the layout view id used by the page
`#{pageDocBean.layoutCssPath}`	Returns the CSS file used by the layout

A.4 ELs Related to Specific Portals

Table A-3 lists EL expressions relevant to portals and describes the types of functionality each provides.

> **Note:**
>
> The portal *internal name* is the name specified for **Portal URL** on the **Overview** page of a portal's administration settings. The portal *display name* is the name specified for **Name**. Many of the EL expressions in Table A-3 call for the portal internal name.

Table A-3   EL Expressions Relevant to Specific Portals

Expression	Function
`#{spaceContext}`	An `oracle.webcenter.spaces.context.SpacesContext` object that provides an access point in the current web request for all portal-related information.
	The value of this expression is whatever is returned on invoking the java API: `SpacesContext.getCurrentInstance()`
`#{spaceContext.currentSpace}`	An `oracle.webcenter.spaces.Space` object that represents the portal associated with the current web request. If the current web request is in the Home portal context, it returns a value of `null`.
	The value of this expression is whatever is returned on invoking the Java API: `SpacesContext.getCurrentInstance().getCurrentSpace()`

Table A-3   (Cont.) EL Expressions Relevant to Specific Portals

Expression	Function
`#{spaceContext.currentSpaceName}`	The name of the portal associated with the current web request. If the current web request is in the Home portal context, it returns a value of `null`. The value of this expression is whatever is returned on invoking the java API: `SpacesContext.getCurrentInstance().getCurrentSpace()`
`#{spaceContext.space['portalName']}` `#{spaceContext.currentSpace}`	An `oracle.webcenter.spaces.Space` object that represents the portal that is named *spaceName* or the current portal (`currentSpace`). For example, `#{spaceContext.space['FinanceProject']}` returns the portal object for the portal called `FinanceProject`. The value of this expression is whatever is returned in Java on invoking `.getSpace(...)` on the current `SpacesManager` passing in the `MDSSession` of the current `ADFContext`.
`#{spaceContext.space['portalName'].metadataPath}` `#{spaceContext.currentSpace.metadataPath}`	The MDS path of the portal metadata document for the portal with specified name *portalName* or the current portal (`currentSpace`). For example, `#{spaceContext.space['FinanceProject'].metadataPath}` evaluates to `/oracle/webcenter/space/metadata/spaces/FinanceProject/space.xml` The value of this expression is whatever is returned in java on invoking `.getMetadataPath()` on the portal object for the portal.

Table A-3   (Cont.) EL Expressions Relevant to Specific Portals

Expression	Function

`#{spaceContext.space['portalName'].metadata}`
 	
`#{spaceContext.currentSpace.metadata}`
	An `oracle.webcenter.spaces.beans.SpaceType` bean that carries metadata about the portal that is named `portalName` or the current portal (`currentSpace`).
	
	The value of this expression is whatever is returned in java on invoking `.getMetadata()` on the portal object for the portal passing in the `MDSSession` of the current `ADFContext`.

`#{spaceContext.space['portalName'].metadata.displayName}`
 	
`#{spaceContext.currentSpace.metadata.displayName}`
	The display name of the portal that is named `portalName` or the current portal (`currentSpace`). For example, if a portal called `Web20Portal` has the display name `web 2.0 Portal`, then `#{spaceContext.space['Web20Portal'].metadata.displayName}` evaluates to `web 2.0 Portal`.

`#{spaceContext.space['portalName'].GSMetadata.groupSpaceURI}`
 	
`#{spaceContext.currentSpace.GSMetadata.groupSpaceURI}`
	The URL of the portal that is named `portalName` or the current portal (`currentSpace`).

`#{WCPrepareImageURL[spaceContext.space['portalName'].metadata.icon]['']}`
 	
`#{WCPrepareImageURL[spaceContext.currentSpace.metadata.icon]['']}`
	A URL to the icon associated with the portal that is named `portalName` or the current portal (`currentSpace`).

`#{spaceContext.space['portalName'].metadata.description}`
 	
`#{spaceContext.currentSpace.metadata.description}`
	The description of the portal that is named `portalName` or the current portal (`currentSpace`). For example, `#{spaceContext.space['FinanceProject'].metadata.description}` evaluates to *Conglomeration of all teams involved in financial activities.*

`#{spaceContext.space['portalName'].metadata.creationDate}`
 	
`#{spaceContext.currentSpace.metadata.creationDate}`
	A *java.util.Calendar* object representing the date-time on which the portal with specified name `portalName` or the current portal (`currentSpace`) was created.

`#{spaceContext.space['portalName'].metadata.createdBy}`
 	
`#{spaceContext.currentSpace.metadata.createdBy}`
	The user-name of the person who created the portal that is named `portalName` or the current portal (`currentSpace`).

Table A-3   (Cont.) EL Expressions Relevant to Specific Portals

Expression	Function

`#{spaceContext.space['`*`portalName`*`'].metadata.keywords}`
 	
`#{spaceContext.currentSpace.metadata.keywords}`
	A comma-delimited list of searchable keywords associated with the portal with the name *`portalName`* or the current portal (`currentSpace`). For example, if the portal *FinanceProject* has the keywords `finance`, `project`, `money`, then `#{spaceContext.space['FinanceProject'].metadata.keywords}` evaluates to *finance, project, money*.

`#{spaceContext.space['`*`portalName`*`'].metadata.offline}`
 	
`#{spaceContext.currentSpace.metadata.offline}`
	Boolean value that indicates whether the portal that is named *`portalName`* or the current portal (`currentSpace`) is offline.

`#{spaceContext.space['`*`portalName`*`'].metadata.closed}`
 	
`#{spaceContext.currentSpace.metadata.closed}`
	Boolean value that indicates whether the portal that is named *`portalName`* or the current portal (`currentSpace`) is closed.

`#{spaceContext.space['`*`portalName`*`'].metadata.selfRegistration}`
 	
`#{spaceContext.currentSpace.metadata.selfRegistration}`
	Boolean value that indicates whether users are allowed to register themselves with the portal that is named *`portalName`* or the current portal (`currentSpace`).

`#{spaceContext.space['`*`portalName`*`'].metadata.discoverable}`
 	
`#{spaceContext.currentSpace.metadata.discoverable}`
	Boolean value that indicates whether users can discover the existence of the portal that is named *`portalName`* or the current portal (`currentSpace`) by searching for it or seeing it listed on the My portals page.

`#{spaceContext.space['`*`portalName`*`'].metadata.publishRSS}`
 	
`#{spaceContext.currentSpace.metadata.publishRSS}`
	Boolean value indicating whether the portal that is named *`portalName`* or the current portal (`currentSpace`) publishes RSS feeds.

`#{spaceContext.space['`*`portalName`*`'].distributionList}`
 	
`#{spaceContext.currentSpace.distributionList}`
	The email address of the mailing list associated with the portal that is named *`portalName`* or the current portal (`currentSpace`).

Table A-3 (Cont.) EL Expressions Relevant to Specific Portals

Expression	Function

`#{spaceContext.space['`*`portalName`*`'].metada ta.customAttributes['`*`attributeName`*`']}`
 	
`#{spaceContext.currentSpace.metadata.cust omAttributes['`*`attributeName`*`']}`
	The value of a specific custom attribute of the name *`attributeName`* for the portal that is named *`portalName`* or the current portal (`currentSpace`). For example, if the *FinanceProject* portal has a custom attribute called `stockPrice` with a value of `13.9`, then `#{spaceContext.space['Fina nceProject'].metadata.cust omAttributes['stockPrice'] }` evaluates to *13.9*.
	
	Note: If you use this EL to provide a value for a field in WebCenter Portal Administration, clicking the **Test** button may return a blank value. However, the value will resolve correctly in the running portal.

`#{WCPrepareImageURL[spaceContext.space['p ortalName'].metadata.logo]['']}`
 	
`#{WCPrepareImageURL[spaceContext.currentS pace.metadata.logo]['']}`
	Returns the logo URL for the portal named *`portalName`* or the current portal (`currentSpace`).

Table A-3   (Cont.) EL Expressions Relevant to Specific Portals

Expression	Function

`#{spaceContext.spacesQuery.property['value']}`
 	
`#{spaceContext.spacesQuery.property['value'].listSpaces}`
	A means of querying a portal using a query parameter in the form of `property['value']`, where `property` means the name of the property, for example, `unionOf`, `shape`, and so on; and `value` means the criteria to use in fetching the list of all portals, discoverable portals, and so on.
	
	If `listSpaces` is appended to the expression, the query returns the list of portals of type `GSMetadata`. For more information, see `Interface GSMetadata` in Oracle WebCenter Portal Java API Reference.
	
	For example, the following EL expression returns a list of all discoverable portals.
	
	`#{spaceContext.spacesQuery.unionOf['DISCOVERABLE'].listSpaces}`
	
	If `listSpaces` is not appended to the EL, then the EL evaluates to an object of type `SpacesQueryParameter`. This object type must be evaluated using `SpacesManager.getSpaces(SpacesQueryParameters)`, which in turn returns a list of portals of type `GSMetadata`.
	
	For example, the following EL expression returns an instance of type `SpacesQueryParameters` with all the query conditions populated.
	
	`#{spaceContext.spacesQuery.unionOf['DISCOVERABLE']}`

`#{spaceContext.spacesQuery.unionOf['USER_JOINED'].shape['ROOT_LEVEL'].listSpaces}`	Returns a list of all portals of which the current user is a member To see an example, refer to Example: Using EL Expressions for Various Portals.
`#{spaceContext.spacesQuery.unionOf['USER_JOINED'].shape['RECURSIVE_FLATTENED'].listSpaces}`	Returns a list of all portals of which the current user is a member This EL also returns all the subportals under each of the parent portals to which the current user has access.

Table A-3 (Cont.) EL Expressions Relevant to Specific Portals

Expression	Function

`#{spaceContext.spacesQuery.unionOf['ALL_Q`
`UERIABLE'].sort['sortBy field picked from`
`WcSpaceHeader']}`
	Returns portals sorted into the order specified by the sort criteria
	
	For example, the following query returns a list of portals to which the user has access sorted by `discoverable` and `lastUpdateDate` fields.
	
	Note that in the following example, the identifier `sp` represents the JPA entity for a portal, `WcSpaceHeader`.
	
	`#{spaceContext.spacesQuery.union`
	`Of['ALL_QUERIABLE'].sort['sp.dis`
	`coverable']['asc']`
	`['sp.lastUpdateDate']`
	`['desc'].listSpaces}`

`#{spaceContext.spacesQuery.unionOf['ALL_Q`
`UERIABLE'].pageNumber[page_num].listSpace`
`s}`
	Allows specifying the page number (0-based) to select from the results matching the query criteria
	
	For example, the following expression returns a list of all portals to which a user has access and returns the third page of the result set:
	
	`#{spaceContext.spacesQuery.union`
	`Of['ALL_QUERIABLE'].pageNumber[2`
	`].listSpaces]}`

`#{spaceContext.spacesQuery.unionOf['ALL_Q`
`UERIABLE'].itemsPerPage[page_num].listSpa`
`ces}`
	Allows specifying the number of results to be included in each page when breaking down the result portal into pages
	
	For example, the following expression returns a list of all portals to which a user has access, listing 10 records per page.
	
	`#{spaceContext.spacesQuery.union`
	`Of['ALL_QUERIABLE'].pageNumber[1`
	`0].listSpaces]}`

`#{spaceContext.spacesQuery.unionOf.ALL_QUERIABLE.`
`where[wCond['sp.createdBy']['like']['monty%']`
`['and'] [wCond['sp.selfSubEnabled']['is']`
`['null'] ['or'] [wCond['sp.selfSubEnabled']['=']`
`['N']] ['not'] ]].listSpaces}`
	Returns a list of all portals a user has access which are created by a specified user, and on which self subscription is enabled. For more information, see `SpacesQueryParameters` and `SpacesQueryFilter` in Oracle WebCenter Portal Java API Reference.

Table A-3   (Cont.) EL Expressions Relevant to Specific Portals

Expression	Function
`#{spaceContext.currentSpace.metadata.portalColor}`	Returns color code used by portal in the portal browser
`#{spaceContext.currentSpace.metadata.acronym}`	Returns the acronym used by the portal in the portal browser

A.4.1 Example: Using EL Expressions for Various Portals

This section provides an example of using EL expressions to query various portals.

In this example, you will use the following EL to display a list of all portals of which you are a member:

```
"#{spaceContext.spacesQuery.unionOf['USER_JOINED'].shape['ROOT_LEVEL'].listSpaces}"
```

You will also use the following methods to display the logo, name, description, and number of members of a portal.

- `displayName` - to display the portal name

- `description` - to display the portal description

- `memberCount` - to display the number of portal members

To query a portal:

1. Create a new task flow, and mark it as available for use, as described in Working with Task Flows in *Building Portals with Oracle WebCenter Portal*.

2. Edit the source code of the task flow. On the **Assets** tab, select the task flow. From the **Actions** menu, select **Edit Source**.

3. In the Edit Source dialog, on the **Fragment** tab, replace the existing code with the following code:

```
<?xml version='1.0' encoding='UTF-8'?>
<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" xmlns:c="http://java.sun.com/jsp/jstl/core" xmlns:fn="http://
java.sun.com/jsp/jstl/functions">
 <af:panelGroupLayout id="pgl1">
 <cust:panelCustomizable id="pc1">
 <table border="0" width="100%">
 <af:iterator id="it1" var="row"
value="#{spaceContext.spacesQuery.unionOf['USER_JOINED'].shape['ROOT_LEVEL'].list
Spaces}">
  <c:set var="displayName" value="#{row.displayName}"/>
 <c:set var="description" value="#{row.description}"/>
 <c:set var="numMembers" value="#{row.memberCount}"/>
 <tr class="PortletText1">
 <af:image id="img1"
 source="#{WCPrepareImageURL[spaceContext.space[row.name].metadata.icon]}/
ICON16"/>
 </td>
```

```
<td>
<af:outputText id="otCol1"
value="#{displayName}"
inlineStyle="color:#333333; font-size:12px; font-weight:bold;"/>
</td>
</tr>
<tr>
<td>
<af:outputText id="otCol3"
value="#{description}"
inlineStyle="color:#666666; font-size:12px;"/>
</td>
</tr>
<tr>
<td>
<af:outputText id="otCol4"
value="members (#{numMembers}) -"
inlineStyle="color:#666666; font-size:10px;"/>
<af:spacer id="spacer1" width="5px"/>
<af:goLink id="gl1"
destination="/spaces/#{row.name}"
targetFrame="_blank"
text="view"/>
</td>
</tr>
</af:iterator>
</table>
</cust:panelCustomizable>
</af:panelGroupLayout>
</jsp:root>
```

4. Add the task flow to a page. In the resource catalog, the custom task flow is available under **UI Components** > **Task Flows**.

The page should now show a list of all the portals of which you are is a member.

A.5 ELs Related to Portal Event Contexts

Table A-4 lists EL expressions relevant to the types of portal events that can trigger the passing of payloads (rather than events relevant to the events feature).

You can access all or part of the event payloads provided in Table A-4 once they have been raised.

For example, for the whole payload, use the following EL:

```
#{wcEventContext.events.eventName}
```

For example:

```
#{wcEventContext.events.WebCenterUserSelected}
```

All of the payloads for the ELs in Table A-4 are Maps. To dereference a map entry, use the standard EL for Maps:

```
#{wcEventContext.events.WebCenterUserSelected.UserName}
```

> **See Also:**
>
> For more information about event wiring, see Wiring Pages, Task Flows, Portlets, and ADF Components in *Building Portals with Oracle WebCenter Portal*.

Table A-4   EL Expressions Relevant to Event Contexts

Expression	Function

Event Context
 	
`#{wcEventContext.events.WebCenterResourceSelected}`
 	
Document Name
 	
`#{wcEventContext.events.WebCenterResourceSelected.name}`
 	
Document Creator
 	
`#{wcEventContext.events.WebCenterResourceSelected.createdBy}`
 	
Document Last Modified By
 	
`#{wcEventContext.events.WebCenterResourceSelected.lastModifiedBy}`
	Use in context wiring between producer and consumer task flows. Returns a map that relates some piece of metadata from the producer to some piece of metadata from the consumer, for example, from a document creator to the creator's Profile.
	
	Producer task flows include:
	• Document Manager
	• Recent Documents
	• Document List Viewer

	Consumer task flows include: • Links (show the items that are linked to the selected document) • Profile (view the Profile of the document's author) • Activity Stream (view activities related to this document) • Tags (tags on this document and a means of accessing the Tag Center)

Table A-4 (Cont.) EL Expressions Relevant to Event Contexts

Expression	Function

Event Context
 	
`#{wcEventContext.events.WebCenterUserSelected}`
 	
User Name
 	
`#{wcEventContext.events.WebCenterUserSelected.userName}`
 	
User GUID
 	
`#{wcEventContext.events.WebCenterUserSelected.userGUID}`
 	
Portal Name
 	
`#{wcEventContext.events.WebCenterUserSelected.portalName}`
 	
Portal GUID
 	
`#{wcEventContext.events.WebCenterUserSelected.portalGUID}`
	Use in context wiring between producer and consumer task flows. Returns a map that relates some piece of metadata from the producer to some piece of metadata from the consumer, for example, from a user to the user's connections.
	
	Producer task flows include:
	• Connections
	• Portal Members
	Consumer task flows include:
	• Connections (show connections of the selected user)
	• Documents task flows (show documents created by the selected user)
	• Activity Stream (view this user's activities)
	• Tags (tags created by this user)
	• Profile (show this user's Profile)

Event Context
 	
`#{wcEventContext.events.WebCenterConnectionSelected}`
 	
Profile User Name
 	
`#{wcEventContext.events.WebCenterConnectionSelected.profileUserName}`
 	
Profile User GUID
 	
`#{wcEventContext.events.WebCenterConnectionSelected.profileUserGUID}`
 	
Connected User Name
 	
`#{wcEventContext.events.WebCenterConnectionSelected.userName}`
 	
Connected User GUID
 	
`#{wcEventContext.events.WebCenterConnectionSelected.userGUID}`
	--

`#{wcEventContext.events.WebCenterSpaceSelected.SpaceName}`	Returns the name of the selected portal
`#{wcEventContext.events.WebCenterSpaceSelected.SpaceGUID}`	Returns the GUID of the selected portal

A.6 ELs Related to Assets

Use the expressions in this section to query for assets. Querying for an asset through an EL expression is similar to querying for it through an API call. That is, you must set query parameters in the format `['property']['like']['value']`, where *property* is the name of the property, for example, `id`, `resourceScope`, and so on, and *value* is the search value for the attribute.

A query can result in single or multiple results. The query designer must decide what is wanted. The query designer determines whether to return one or multiple results by encountering one of the following values in the expression:

- `singleResult`—Returns a single asset. When no matching asset is found, null is returned.

- `resultList`—Returns a list of assets. When no matching assets are found, an empty list is returned.

> **Note:**
>
> Occurrences of `singleResult` or `resultList` in the expression are used internally as the query end point, and, after this, the query is executed immediately. Anything set after the end point can result in unexpected behavior.

The following example returns the first page template asset found with a display name that contains `myPage`:

```
#{srmContext[wCond['resourceType']['like']['siteTemplate']]['and']
[wCond['displayName']['like']['myPage']].singleResult}
```

The following example returns a list of page template assets residing in the directory `resourceDir`, with a description that contains `sampleDesc`:

```
#{srmContext[wCond['resourceType']['like']['siteTemplate']]['and']
[wCond['description']['like']['sampleDesc']]['and'][wCond['contentDirectory']['like']
['resourceDir'].resultList}
```

A property of this class includes any attribute of this class. Example properties include: `Id`, `DisplayName`, `iconURI`, `contentDirectory`, and so on.

The property value can be an explicit value or an EL expression that returns that type of value. For example, the following two queries return the same result:

```
#{srmContext[wCond['id']['like']['resourceId']].singleResult}
```

```
#{srmContext[wCond['id']['like']
['spacesContext.currentSpace.uiMetadata.siteTemplateId']].singleResult}
```

You can use any property of this class in an EL-based query in the format *property*`['value']` and in any order. For example, the following two queries return the same result:

```
#{srmContext[wCond['resourceScope']['like']['scopeName']]['and'][wCond['id]['like']
['resourceId']].singleResult}
```

```
#{srmContext[wCond['id']['like']['resourceId']]['and'][wCond['resourceScope']['like']
['scopeName']].singleResult}
```

Table A-5 lists EL expressions relevant to assets and describes the types of functionality each provides.

Table A-5   ELs Relevant to Assets

EL	Function
`#{srmContext[wCond['id']['like']` `['resourceGUID']]}`	Returns the asset with the specified ID
`#{srmContext[wCond['displayName']['like']` `['resourceDisplayName']]}`	Returns any assets with the specified display name
`#{srmContext[wCond['displayNameKey']` `['like']['displayNameKey']]}`	Returns any assets with the specified display name key
`#{srmContext[wCond['description']['like']` `['resourceDescription']].singleResult}`	Returns one result of an asset that contains the specified value in its description To get multiple results, use `resultList` in lieu of `singleResult`.
`#{srmContext[wCond['descriptionKey']` `['like']['descriptionKey']].singleResult}`	Returns one result of an asset with the specified description key The description key is the key in the `xsrt` file for the asset description. To get multiple results, use `resultList` in lieu of `singleResult`.
`#{srmContext[wCond['createdDate']['like']` `['resourceCreationDate']]}`	Returns any asset with the specified creation date
`#{srmContext[wCond['modifiedBy']['like']` `['resourceLastModifiedBy']]}`	Returns any asset that was last modified by the user with the specified user name
`#{srmContext[wCond['modifiedDate']` `['like']['resourceLastModifiedDate']]}`	Returns any asset that was last modified by the specified date
`#{srmContext[wCond['resourceScope']` `['like']['resourceScope']]}`	Returns any asset that falls within the specified scope

Table A-5   (Cont.) ELs Relevant to Assets

EL	Function
`#{srmContext[wCond['category']['like']['categoryName']]}`	Returns any asset that falls within the specified category For example: `#{srmContext[wCond['category']['like']['siteTemplates']]}` List of possible category names includes: `siteTemplate`, `pageStyle`, `dataPresenter`, `contentPresenter`, `resourceCatalog`, `taskFlow`, `dataControl`, `taskFlowStyle`, and `skin`.
`#{srmContext[wCond['contentDirectory']['like']['contentDirectory']]}`	Returns any asset that is stored within the specified directory
`#{srmContext[wCond['metadataFile']['like']['metadataFileLocation']]}`	Returns asset metadata from the specified metadata file For example, the following expression returns asset metadata from the following file `/home/metadata/data.xml`: `#{srmContext[wCond['metadataFile']['like']['/home/metadat/data.xml']]}`
`#{srmContext[wCond['jspx']['like']['jspxFileLocation']]}`	Returns any `jspx` file in the specified location For example, the following expression returns the `page.jspx` file: `#{srmContext[wCond['jspx']['like']['/home/web/page.jspx']]}`
`#{srmContext[wCond['pageDef']['like']['pageDefinition']]}`	Returns any `jspx` file with the specified page definition
`#{srmContext[wCond['iconURI']['like']['iconURI']]}`	Returns the icon at the specified icon URI
`#{srmContext[wCond['excludeFrom']['like']['excludeFromScopes']]}`	In a larger expression, returns all specified resources except those available in the excluded scopes

Table A-5   (Cont.) ELs Relevant to Assets

EL	Function
`#{srmContext[wCond['usesCustomSecurity']['like']['usesCustomSecurity']]}`	In a larger expression, returns any asset the either does or does not use custom security Set *usesCustomSecurity* to TRUE or FALSE. For example: `#{srmContext[wCond['usesCustomSecurity']['like']['TRUE']]}`
`#{srmContext[wCond['seeded']['like']['seeded']]}`	In a larger expression, returns any asset that is or is not seeded, depending on the provided value Set *seeded* to TRUE or FALSE. For example: `#{srmContext[wCond['seeded']['like']['TRUE']]}`
`#{srmContext[wCond['visibleType']['like']['visibleType']]}`	In a larger expression, returns any asset that is of the specified type of visibility Set *visibleType* to TRUE, FALSE, or NEVER to indicate that the asset is never shown. For example, the following expression returns assets that are visible, that is, assets that are set to **Show**: `#{srmContext[wCond['visibleType']['like']['TRUE']]}`
`#{srmContext[wCond['visible']['like']['visible']]}`	In a larger expression, returns one or multiple assets with visibility set to either TRUE or FALSE
`#{srmContext[wCond['createdBy']['like']['resourceCreator']]}`	In a larger expression, returns any asset created by the specified user
`#{srmContext[wCond['resourceType']['like']['resourceType']]}`	In a larger expression, returns one or multiple assets of the specified type For example, the following expression searches for an asset of the type SITE_TEMPLATE: `#{srmContext[wCond['resourceType']['like']['SITE_TEMPLATE']]}` Note that all asset types are listed in Oracle WebCenter Portal Java API Reference, in the `GenericSiteResourceTypes` class.

Table A-5   (Cont.) ELs Relevant to Assets

EL	Function
`#{srmContext[wCond['version']['like']` `['version']]}`	In a larger expression, returns one or multiple assets available in the application of the specified version
`#{srmContext[wCond['resourceScope']` `['like']['scopeName']]}`	In a larger expression, returns one or multiple assets available in the specified scope For example, the following expression searches for assets in the scope (in this instance, the portal) `MyPortal`: `#{srmContext[wCond['resourceScop` `e']['like']['MyPortal']]}` To search in the default scope, that is, the application scope, use `defaultScope`.
`#{srmContext[wCond['searchType']['like']` `['searchType']]}`	In a larger expression, returns one or multiple assets that contain or equal the values set by other included expressions Set `searchType` to `CONTAINS` or `EQUALS`.

A.6.1 Example: Using EL Expressions for Assets

This section provides an example of using an EL expression to display a page template based on a user's role.

In this example, you will use an EL expression to enable a specific page template to be displayed for portal managers so that only they can access the features or links available in that page template.

To display a page template based on a user role:

1. Go to the desired portal's administration settings.

2. On the **Settings** page, under **Assets** , for **Page Template** click the **Advanced Edit Options** icon next to the **Page Template** drop-down list, and then select **Expression Builder** to open the editor.

3. In the **Type a value or expression**, enter the following expression:

```
#{srmContext[wCond['resourceType']['like']['siteTemplate']][wCond['displayName']
['like'][WCSecurityContext.userInScopedRole['Moderator'] ? 'Fusion Side
Navigation' : 'WebCenter Portal Top Navigation']].singleResult.id}
```

Where:

- `srmContext[wCond['resourceType']['like']['siteTemplate']]`

   Gets a list of all page templates in the scope

- `[wCond['displayName']['like']`
  `[WCSecurityContext.userInScopedRole['Moderator'] ? 'Fusion`
  `Side Navigation' : 'WebCenter Portal Top Navigation']]`

  Reduces the list down to one depending on whether the current user is a portal manager. If the user is a portal manager, the Fusion Side Navigation page template is applied. For all other user roles, the portal is rendered using the WebCenter Portal Top Navigation page template.

- `singleResult.id`

  Returns the GUID of the required entry.

4. Click **OK**.

A.7 ELs Related to Security

ELs Related to Security lists EL expressions relevant to application security and describes the types of functionality they provide.

Table A-6   EL Expressions Relevant to Security

Expression	Function
`#{security.authenticated}`	Returns `true` when the current user is authenticated in the context in which the EL is being invoked, otherwise `false`.
`#{securityContext.userName}`	Returns the user name of the currently logged in user. If the current user is not logged in, this expression returns no value.
`#{WCSecurityContext.currentUser['`*`userN`*`ame`*`']}`	Returns the value `true` if the current user is the specified user, otherwise `false`.
`#{WCSecurityContext.userInGroup['`*`group`*`']}`	Returns the value *true* if the current user is assigned the specified group, for example: `#{WCSecurityContext.userInGroup['Administrators']}`
`#{security.pageContextCommunityModerator}`	Returns the value *true* if the current user is the Portal Manager of the current portal.
`#{WCSecurityContext.userInScopedRole['`*`role`*`']}`	Returns the value *true* if the current user is assigned the specified role in the current scope. Role can be `Moderator` (or Portal Manager, which is the display name), or a custom role defined in that scope. The scope is implicitly resolved to be the current scope. If you use this EL in the home portal, it resolves to the home portal's GUID and roles defined at the application level. If you use this EL in a portal scope, it resolves to roles defined for the portal.

A.8 ELs Related to General Settings

ELs Related to General Settings lists EL expressions relevant to general application settings and describes the types of functionality they provide. All listed ELs apply to both portals built with WebCenter Portal and Portal Framework applications built with JDeveloper.

Table A-7   EL Expressions Relevant to General Settings

Expression	Function
`#{generalSettings.userTimeZone}`	Returns the time zone the current user has selected in application preferences.
`#{generalSettings.preferredDateStyle}`	Returns the date format the current user has selected in application preferences.
`#{generalSettings.preferredDatePattern}`	Returns the current user's preferred date format pattern if it has been set, else, returns `null`.
`#{generalSettings.preferredTimeStyle}`	Returns the time format the current user has selected in application preferences.
`#{generalSettings.preferredTimePattern}`	Returns the current user's preferred time format pattern if it has been set, else, returns `null`.
`#{generalSettings.preferredDateTimePattern}`	Returns the current user's preferred date and time format pattern if it has been set, else, returns `null`.
`#{generalSettings.preferredAccessibilityMode}`	Returns the current user's preferred accessibility mode (either `default`, `inaccessible`, or `screenReader`)
`#{generalSettings.preferredSkinName}`	The current user's preferred skin name if one is specified, otherwise the default value.
`#{generalSettings.formattedCurrentDate}`	Returns the current date in the current user's selected locale.
`#{generalSettings.formattedCurrentTime}`	Returns the current time in the current user's selected locale.
`#{generalSettings.formattedCurrentDateTime}`	Returns the current date and time in the current user's selected locale.
`#{requestContext.skinFamily}`	Returns the name of the ADF Faces skin family being used for the current web request, depending on factors such as what has been configured at the application level, the current user's preference setting, and so on.

A.9 ELs Related to Portal Resources

Use the expressions in this section to query for portal resources. Querying for a portal resource through an EL expression is similar to querying for it through an API call. That is, you must set query parameters in the format `.property['value']`, where `property` is the name of the property, for example, `id`, `resourceScope`, and so on, and `value` is the search value for the attribute.

A query can result in single or multiple results. The query designer must decide what is wanted. The query designer determines whether to return one or multiple results by encountering one of the following values in the expression:

- `singleResult`—Returns a single portal resource. When no matching resource is found, null is returned.

- `resultList`—Returns a list of portal resources. When no matching portal resources are found, an empty list is returned.

Note:

Occurrences of `singleResult` or `resultList` in the expression are used internally as the query end point, and, after this, the query is executed immediately. Anything set after the end point can result in unexpected behavior.

The following example returns the first page template portal resource found with a display name that contains `myPage`:

```
#{srmContext.resourceType['siteTemplate'].displayName['myPage'].singleResult}
```

The following example returns a list of page template portal resources residing in the directory `resourceDir`, with a description that contains `sampleDesc`:

```
#{srmContext.resourceType['siteTemplate'].description['sampleDesc'].contentDirectory[
'resourceDir'].resultList}
```

A property of this class includes any attribute of this class. Example properties include: `Id`, `DisplayName`, `iconURI`, `contentDirectory`, and so on.

The property value can be an explicit value or an EL expression that returns that type of value. For example, the following two queries return the same result:

```
#{srmContext.id['resourceId'].singleResult}
```

```
#{srmContext.id['spacesContext.currentSpace.uiMetadata.siteTemplateId'].singleResult}
```

You can use any property of this class in an EL-based query in the format `property['value']` and in any order. For example, the following two queries return the same result:

```
#{srmContext.resourceScope['scopeName'].id['resourceId'].singleResult}
```

```
#{srmContext.id['resourceId'].resourceScope['scopeName'].singleResult}
```

Table A-8 lists EL expressions relevant to portal resources and describes the types of functionality each provides. Many of the expressions in Table A-8 are building blocks for narrowing down the portal resource or resources you want returned. That is, they are not necessarily meant to be used by themselves, but rather in an assembly of ELs to form the desired query.

For example, the following expression uses three portal resource-related ELs to form a single query that returns a particular portal template. It retrieves the template `storesMasterTemplate` from the parent portal `stores`:

```
#{srmContext.resourceScope['stores'].resourceType['siteTemplate'].displayName['stores
MasterTemplate'].singleResult}
```

For information about EL expressions relevant only to WebCenter Portal assets, see ELs Related to Assets.

Table A-8   ELs Relevant to Portal Resources

Table A-8   (Cont.) ELs Relevant to Portal Resources

EL	Function
`#{srmContext.id['`*`resourceGUID`*`']}`	Returns the portal resource with the specified ID
`#{srmContext.displayName['`*`resourceDisplay Name`*`']}`	Returns any portal resources with the specified display name
`#{srmContext.displayNameKey['`*`displayNameK ey`*`']}`	Returns any portal resources with the specified display name key
`#{srmContext.description['`*`resourceDescrip tion`*`'].singleResult}`	Returns one result of a portal resource that contains the specified value in its description To get multiple results, use `resultList` in lieu of `singleResult`.
`#{srmContext.descriptionKey[`*`descriptionKe y`*`].singleResult}`	Returns one result of a portal resource with the specified description key The description key is the key in the `xsrt` file for the portal resource description. To get multiple results, use `resultList` in lieu of `singleResult`.
`#{srmContext.createdDate['`*`resourceCreatio nDate`*`']}`	Returns any portal resource with the specified creation date
`#{srmContext.modifiedBy['`*`resourceLastModi fiedBy`*`']}`	Returns any portal resource that was last modified by the user with the specified user name
`#{srmContext.modifiedDate['`*`resourceLastMo difiedDate`*`']}`	Returns any portal resource that was last modified by the specified date
`#{srmContext.resourceScope['`*`resourceScope`*`']}`	Returns any portal resource that falls within the specified scope
`#{srmContext.category['`*`categoryName`*`']}`	Returns any portal resource that falls within the specified category For example: `#{srmContext.category['siteTempl ates']}`
`#{srmContext.contentDirectory['`*`contentDir ectory`*`']}`	Returns any portal resource that is stored within the specified directory

Table A-8 (Cont.) ELs Relevant to Portal Resources

EL	Function
`#{srmContext.metadataFile['`*`metadataFileLocation`*`']}`	Returns portal resource metadata from the specified metadata file For example, the following expression returns resource metadata from the following file `/home/metadata/data.xml`: `#{srmContext.metadataFile['/home/metadat/data.xml']}`
`#{srmContext.jspx['`*`jspxFileLocation`*`']}`	Returns any `jspx` file in the specified location For example, the following expression returns the `page.jspx` file: `#{srmContext.jspx['/home/web/page.jspx']}`
`#{srmContext.pageDef['`*`pageDefinition`*`']}`	Returns any `jspx` file with the specified page definition
`#{srmContext.iconURI['`*`iconURI`*`']}`	Returns the icon at the specified icon URI
`#{srmContext.excludeFrom['`*`excludeFromScopes`*`']}`	In a larger expression, returns all specified portal resources except those available in the excluded scopes
`#{srmContext.usesCustomSecurity['`*`usesCustomSecurity`*`']}`	In a larger expression, returns any portal resource that either does or does not use custom security Set *usesCustomSecurity* to TRUE or FALSE. For example: `#{srmContext.usesCustomSecurity['TRUE']}`
`#{srmContext.seeded['`*`seeded`*`']}`	In a larger expression, returns any portal resource that is or is not seeded, depending on the provided value Set *seeded* to TRUE or FALSE. For example: `#{srmContext.seeded['TRUE']}`

Table A-8   (Cont.) ELs Relevant to Portal Resources

EL	Function
`#{srmContext.visibleType['visibleType']}`	In a larger expression, returns any portal resource that is of the specified type of visibility
	Set `visibleType` to TRUE, FALSE, or NEVER to indicate that the portal resource is never shown. For example, the following expression returns portal resources that are visible, that is, portal resources that are set to **Show**:
	`#{srmContext.visibleType['TRUE']}`
`#{srmContext.visible['visible']}`	In a larger expression, returns one or multiple portal resources with visibility set to either TRUE or FALSE
`#{srmContext.createdBy['resourceCreator']}`	In a larger expression, returns any portal resource created by the specified user
`#{srmContext.resourceType['resourceType']}`	In a larger expression, returns one or multiple portal resources of the specified type
	For example, the following expression searches for a portal resource of the type SITE_TEMPLATE:
	`#{srmContext.resourceType['SITE_TEMPLATE']}`
	Note that all resource types are listed in Oracle WebCenter Portal Java API Reference, in the `GenericSiteResourceTypes` class.
`#{srmContext.version['version']}`	In a larger expression, returns one or multiple portal resources available in the application of the specified version

Table A-8   (Cont.) ELs Relevant to Portal Resources

EL	Function
#{srmContext.resourceScope['*scopeName*']}	In a larger expression, returns one or multiple portal resources available in the specified scope For example, the following expression searches for portal resources in the scope (in this instance, the portal) MyPortal: #{srmContext.resourceScope['MyPortal']} To search in the default scope, that is, the application scope, use defaultScope.
#{srmContext.searchType['*searchType*']}	In a larger expression, returns one or multiple portal resources that contain or equal the values set by other included expressions Set *searchType* to CONTAINS or EQUALS.

A.10 ELs Related to Navigation

Table A-9 lists EL expressions relevant to application navigation and describes the types of functionality they provide.

Table A-9   EL Expressions Relevant to Navigation

Expression	Function
Navigation Context Expressions	
#{navigationContext.defaultNavigationModel}	Returns default navigation model. It gets the value from the preference bean. The preference bean gets the value based on the preference setting in adf-config.xml, static value. The following example returns the specified default tree model: #{navigationContext.defaultNavigationModel.defaultTreeModel} You can also use defaultMenuModel and defaultListModel (see next listing).

Table A-9   (Cont.) EL Expressions Relevant to Navigation

Expression	Function
`#{navigationContext.currentNavigationModel}`	Returns the navigation model used to navigate to the current view (that is, the current page)
	The current navigation model is set only when the `processAction` is called.
	The following example returns the default tree model for the current navigation model:
	`#{navigationContext.currentNavigationModel.defaultTreeModel}`
	You can also use `defaultMenuModel` and `defaultListModel`.
	See Also: #{navigationContext.processAction}.
`#{navigationContext.navigationModel['`*`model_path`*`']}`	This EL has been deprecated. Use `#{navigationContext.defaultNavigationModel}` or `#{navigationContext.currentNavigationModel}` in its place.

Table A-9   (Cont.) EL Expressions Relevant to Navigation

Expression	Function
`#{navigationContext.processAction}`	Returns the default navigation method for binding to UI component's `actionListener` attribute. This assumes that the target resource to navigate to is passed in through the action UI component's `node` or `path` attribute.

The `node` attribute is typically used for the iterative case; while the `path` attribute is typically used to create a static link. If both are specified, the `node` attribute is used. Note that when using the `path` attribute, you must also specify the `model` attribute to pass in the navigation model object. If the `model` attribute is not specified, the current navigation model is used.

Example using the `node` attribute:

```
<af:forEach var="node" varStatus="vs"

items="#{navigationContext.defaultNavigationModel.rootNode.children}">
    <af:subform id="pt_sfm1">
      <af:commandLink id="pt_cl1"
        text="#{node.title}"
        inlineStyle="font-size:small;color:White;"

actionListener="#{navigationContext.processAction}"
        action="pprnav">
      <f:attribute name="node" value="#{node}"/>
        <af:showPopupBehavior popupId="menuPopup"
          align="afterStart"
          triggerType="mouseOver"/>
    </af:commandLink>
```

Example using the `path` attribute:

```
<af:commandLink id="pt_cl1" text="Prescriptions"
    inlineStyle="font-size:small; color:White;"

actionListener="#{navigationContext.processAction}"
    action="pprnav">
<f:attribute name="path" value="/prescriptions"/>
<f:attribute name="model" value="#{navigationContext.defaultNavigationModel}"/>
</af:commandLink>
```

Table A-9   (Cont.) EL Expressions Relevant to Navigation

Expression	Function
Navigation Model Expressions	

`#{navigationContext.defaultNaviga tionModel.defaultTreeModel}`
 	
`#{navigationContext.defaultNaviga tionModel.defaultMenuModel}`
 	
`#{navigationContext.defaultNaviga tionModel.defaultListModel}`
 	
`#{navigationContext.defaultNaviga tionModel.defaultSiteMap}`
	Returns a tree/menu/list model based on the default settings
	
	The default values for the settings are specified in the next row. For example, the default value for `depth` is `0`.
	
	For `defaultSiteMap`, it returns the XML for the site map of the navigation based on the default settings.
	
	This expression returns a model of type:
	• `ChildPropertyTreeModel`—used in `<af:tree>` component
	• `ChildPropertyMenuModel`—used in `<af:breadcrumbs>` or `<af:menu>` component
	• `ListNavigationResource`—used in `<af:foreach>`

`#{navigationContext.defaultNaviga tionModel.treeModel['parameters'] }`
 	
`#{navigationContext.defaultNaviga tionModel.menuModel['parameters'] }`
 	
`#{navigationContext.defaultNaviga tionModel.listModel['parameters'] }`
 	
`#{navigationContext.defaultNaviga tionModel.siteMap['parameters']}`
	Returns tree/menu/list model based on the specified comma-separated parameters. Available parameters (with default values as examples) are:
	• `startNode=/`—specify the starting node of the model (do not need "/" prefix unless requesting the root node, for example, `home`).
	• `includeStartNode=true`—specify `true` if you want to include the starting node (for example, the root node above) or `false` to start from its children.
	• `depth=0`—defines the initial depth of fetching. "0" means fetch the entire tree, which may take a long time. In which case, use "1" to fetch on demand (when users click the **Expand** icon).
	• `prefetchOnly=false`—by default (`true`), it returns nodes up to the depth level requested initially; deeper level nodes are returned on demand. Use `false` if you want only the initial sets of nodes. In which case, it does not return deeper nodes later on when requested, even when there are deeper nodes.
	For `siteMap`, the available parameters are `startNode` and `includeStartNode`.
	Example:
	
	`#{navigationContext.defaultNavigationMod el.treeModel['startNode=home,includeStar tNode=false,depth=2']}`

Table A-9   (Cont.) EL Expressions Relevant to Navigation

Expression	Function

#{navigationContext.defaultNavigationModel.rootNode}	Returns a root node (of type NavigationResource) of the navigation model.
	
	Example:
	
	<pre><af:commandLink id="pt_cl1"
	text="Prescriptions"
	    inlineStyle="font-size:small;
	color:White;"
	
	actionListener="#{navigationContext.proc
	essAction}"
	    action="pprnav">
	  <f:attribute name="node"
	        value="#{navigationContext.
	        defaultNavigationModel.
	        rootNode.children[2]"/>
	</af:commandLink></pre>

#{navigationContext.defaultNavigationModel.node['*path*']}	Returns a node (of type NavigationResource) based on the path specified (you do not need "/" prefix unless you are requesting the root node, for example, '/') Example: <pre>#{navigationContext.defaultNavigationModel.node['home/page1']}</pre>
#{navigationContext.defaultNavigationModel.currentSelection}	Returns currently selected navigation portal resource.

Table A-9   (Cont.) EL Expressions Relevant to Navigation

Expression	Function
`#{navigationContext.defaultNavigationModel.newCurrentSelection['path']}`	Sets the current selection to a node specified by the given path (you do not need "/" prefix unless you are requesting the root node, for example, '/'). If successful, returns the newly selected node of type `NavigationResource`.
	The user must have the ability to explicitly set the current selection without having to actually navigate to a node.
	This can be used where the user enters a page directly, and no selection is set. It provides a mechanism for the user to control what is the default when there is no current selection.
	Example:
| | ```
<c:set   value="$
{navigationContext.defaultNavigationMode
l.newCurrentSelection['home/page1']}"
  var="currSel" scope="session"/>
<c:if test="${currSel!=null}">
  <af:forEach
items="#{currSel.children}"
    var="cnode" varStatus="cnodestatus">
...</c:if>
``` |
`#{navigationContext.defaultNavigationModel.attributes.Description}` `#{navigationContext.defaultNavigationModel.attributes['Description']}`	Returns the value of the specified attribute of the navigation model.
`#{navigationContext.defaultNavigationModel.properties['propertyName']}`	Returns the value of the specified property of the navigation model, where `propertyName` is either `rootNode` or `currentSelection`.
Navigation Portal Resource Expressions	
`#{node.attributes.Description}` `#{node.attributes['Description']}`	Returns the value of the specified attribute of the navigation portal resource.
`#{node.parameters.StockSymbol}` `#{node.parameters['StockSymbol']}`	Returns the value of the specified parameter of the navigation portal resource.
`#{node.parametersRaw.StockSymbol}` `#{node.parametersRaw['StockSymbol']}`	Returns the raw value of the specified parameter of the navigation portal resource (before it is evaluated).
`#{node.title}`	Returns the title of the navigation portal resource.
`#{node.path}`	Returns the path to the navigation portal resource.

Table A-9   (Cont.) EL Expressions Relevant to Navigation

Expression	Function
`#{node.externalURL}`	Returns the URL for the navigation portal resource of type `External Link`.
`#{node.prettyUrl}`	Returns the identifying path for this navigation portal resource.
`#{node.prettyUrlPath}` `#{node.prettyUrlPath[N]}`	Returns a collection of identifying paths by depth to enable its use as a starting path to drive another navigation view. For example, assuming the `currentSelection` is `home/company/products/applications`, the following EL returns `home/company/products`: `#{node.prettyUrlPath[3]}`

`#{node.goLinkPretty Url}`	Returns the identifying path for this navigation portal resource suitable for `goLink` destination
	
	`<af:goLink id="pt_gl2"`
	`text="#{node.title}"`
	
	`destination="#{node.goLinkPrettyUrl}"`
	
	`targetFrame="#{node.attributes['Target']}"`
	`        inlineStyle="font-size:small;#{node.selected ?`
	`        'font-weight:bold;' : ''}"/>`

`#{node.separator ? ... : ...}`	Returns whether this portal resource is a separator item.
`#{node.navigable ? ... : ...}`	Returns whether it is possible to navigate to this portal resource.
`#{node.selected ? ... : ...}`	Returns whether this portal resource is currently selected.
`#{node.currentlySelected ? ... : ...}`	Returns whether the portal resource is the currently selected portal resource and the model is the currently selected model.

Table A-9   (Cont.) EL Expressions Relevant to Navigation

Expression	Function
`#{`*node*`.onSelectedPath ? ... : ...}`	Returns whether this node lies on the selected path. This is useful for highlighting the selected tab, for example.
	For example, assuming the currently selected node is `home/company/products/applications`, the following EL highlights the `home` tab.

	<pre><c:set value="$
	{navigationContext.defaultNavigationMode
	l.node['home']}"
	  var="home" scope="session"/>
	    <af:commandImageLink id="cil1"
	        icon="#{(home.onSelectedPath) ?
	        '/images/caremark/nav/
	HomeSelected.png' :
	        '/images/caremark/nav/
	HomeIcon.png'}"
	
	actionListener="#{navigationContext.proc
	essAction}"
	        action="pprnav">
	    <f:attribute name="node"
	value="#{home}"/></af:commandImageLink></pre>

`#{`*node*`.leaf ? ... : ...}`	Returns whether this resource is a leaf node.
`#{`*node*`.parent}`	Returns the parent node (of type `NavigationResource`) of this node. For the root node, `null` is returned.
`#{`*node*`.ancestors}`	Returns the hierarchy list of ancestors of this node (of type NavigationResource) starting with the root node.
	For example, assuming the current node is `home/company/products/applications`, the following ELs return the following values:
	• `#{node.ancestors[1]}` returns the node for `home` • `#{node.ancestors[3]}` returns the node for home/company/products
`#{`*node*`.depth}`	Returns the depth of this node from the root node. Root node has a depth of zero.
`#{`*node*`.siblings}`	Returns the list of siblings (of type `NavigationResource`) of this node (inclusive).

Table A-9   (Cont.) EL Expressions Relevant to Navigation

Expression	Function
`#{`*node*`.nextSibling}`	Returns the next sibling in the list (of type `NavigationResource`) of this node. For example, the following code ensures that you do not output two separators in a row: ``` <c:if test="${(node.separator) && (! node.nextSibling.separator}"> <af:separator id="s166"/> </c:if> ```
`#{`*node*`.previousSibling}`	Returns the previous sibling in the list (of type `NavigationResource`) of this node.
`#{`*node*`.index}`	Returns the zero-relative index of this node relative to its siblings.

`#{`*node*`.children}`
`#{`*node*`.children[`*N*`].title}`
`#{`*node*`.children[`*N*`].children[`*N*`].title}`
	Returns a collection of child resources.

`#{`*node*`.childCount}`	Returns the number of children of this node.
`#{`*node*`.childByIndex['index']}`	Returns the child node (of type `NavigationResource`) specified by the zero-relative index. If not found, this returns `null`.
`#{`*node*`.childByPath['admin']}`	Returns the child node (of type `NavigationResource`) specified by the path relative to this node. The path can be deeper than one level and does not need the `'/'` prefix. If not found, this returns `null`.
`#{`*node*`.properties['propertyName']}`	Returns the value of the specified property of the navigation portal resource, where `propertyName` is one of the following: `separator`, `title`, `prettyUrl`, `prettyUrlPath`, `depth`, `leaf`, `childCount`, `navigable`, `selected`, or `currentlySelected`.

A.11 ELs Related to Tools and Services

ELs Related to Tools and Services lists EL expressions relevant to tools and services and describes the types of functionality they provide. All listed ELs apply to both portals built with WebCenter Portal and Portal Framework applications built with JDeveloper.

Table A-10   EL Expressions Relevant to Tools and Services

Table A-10   (Cont.) EL Expressions Relevant to Tools and Services

Expression	Function
`#{webcenterService['serviceId']}`	Returns an implementation of `oracle.webcenter.framework.service.Service` object representing the WebCenter Portal tool or service represented by the service ID `serviceId`.In the example, object of class `oracle.webcenter.doclib.internal.spaces.DoclibService` is returned: `#{webcenterService['oracle.webcenter.doclib']}` For service IDs, see Table A-11.
`#{webcenterService['serviceId'].configured}`	Returns a Boolean value that indicates whether the WebCenter Portal tool or service represented by the service ID `serviceId` is configured for use in the current WebCenter Portal instance. For example, the following EL returns `true` if discussions can be used in the application, `false` otherwise: `#{webcenterService['oracle.webcenter.collab.forum'].configured}` For service IDs, see Table A-11.
`#{sessionContext['oracle.webcenter.collab.forum'].groupInfo['portalName'].forumId}`	Returns the forum ID of the specified portal discussion forum. Enter the portal name in lieu of `portalName`.
`#{sessionContext['oracle.webcenter.collab.forum'].groupInfo['portalName'].categoryId}`	Returns the category ID of the specified WebCenter Portal discussion forums. Enter the portal name in lieu of `portalName`.

Table A-11 lists service IDs associated with WebCenter Portal tool and services.

Table A-11   Service and Tool IDs

Service/Tool	ID
Announcements	oracle.webcenter.collab.announcement
Discussions	oracle.webcenter.collab.forum
Documents and Wikis	oracle.webcenter.doclib
Events	oracle.webcenter.collab.calendar.community

Table A-11   (Cont.) Service and Tool IDs

Service/Tool	ID
Instant Messaging and Presence (IMP)	oracle.webcenter.collab.rtc
Links	oracle.webcenter.relationship
Lists	oracle.webcenter.list
Mail	oracle.webcenter.collab.mail
Notifications	oracle.webcenter.notification
Page	oracle.webcenter.page
People Connections: Activity Stream	oracle.webcenter.activitystreaming
People Connections: Connections	oracle.webcenter.peopleconnections.connections
People Connections: Feedback	oracle.webcenter.peopleconnections.kudos
People Connections: Message Board	oracle.webcenter.peopleconnections.wall
People Connections: Profile	oracle.webcenter.peopleconnections.profile
RSS	oracle.webcenter.rss
Search	oracle.webcenter.search
Tags	oracle.webcenter.tagging
Blogs	oracle.webcenter.blog
Worklist	oracle.webcenter.worklist

A.12 ELs Related to Documents

Table A-12 lists EL expressions relevant to documents and describes the types of functionality they provide.

Table A-12   EL Expressions Relevant to Documents

Expression	Function
`#{documentsService.defaultConnectionName}`	Gets the default content repository connection name. Returns `null` if no connection name is defined.
`#{documentsService.configured}`	Checks whether the documents feature is configured. Returns `true` if the documents feature is configured, otherwise `false`.

A.13 ELs Related to People Connections

Table A-13 lists EL expressions relevant to the people connections Profile feature and describes the types of functionality they provide.

> **Note:**
>
> The entry `securityContext.userName`, included in every Profile expression, returns the name of the current user. Note also that information is returned only if it is present in the user's profile. If the information is not included in the profile, a null value is returned. Information is also returned only if the current user is allowed to see it.

Table A-13   EL Expressions Relevant to People Connections (Profile)

Expression	Function

`#{webCenterProfile[`*userName*`].managerDisplayName}`
 	
`#{webCenterProfile[securityContext.userName].managerDisplayName}`
	The display name of the user's manager.

`#{webCenterProfile[`*userName*`].employeeNumber}`
 	
`#{webCenterProfile[securityContext.userName].employeeNumber}`
	The user's employee number.

`#{webCenterProfile[`*userName*`].businessPOBox}`
 	
`#{webCenterProfile[securityContext.userName].businessPOBox}`
	The post office box number associated with the user.

`#{webCenterProfile[`*userName*`].timeZone}`
 	
`#{webCenterProfile[securityContext.userName].timeZone}`
	The time zone in which the user's home office is located.

`#{webCenterProfile[`*userName*`].description}`
 	
`#{webCenterProfile[securityContext.userName].description}`
	A description of the user (from Profile "About Me").

`#{webCenterProfile[`*userName*`].department}`
 	
`#{webCenterProfile[securityContext.userName].department}`
	The department to which the user belongs.

`#{webCenterProfile[`*userName*`].businessPager}`
 	
`#{webCenterProfile[securityContext.userName].businessPager}`
	The user's business pager number.

`#{webCenterProfile[`*userName*`].businessCity}`
 	
`#{webCenterProfile[securityContext.userName].businessCity}`
	The city in which the user is located.

`#{webCenterProfile[`*userName*`].maidenName}`
 	
`#{webCenterProfile[securityContext.userName].maidenName}`
	The user's surname or last name before marriage.

`#{webCenterProfile[`*userName*`].businessFax}`
 	
`#{webCenterProfile[securityContext.userName].businessFax}`
	The user's business fax number.

Table A-13   (Cont.) EL Expressions Relevant to People Connections (Profile)

Expression	Function
`#{webCenterProfile[`*userName*`].dateofHire}` `#{webCenterProfile[securityContext.userName].dateofHire}`	The user's date of hire.
`#{webCenterProfile[`*userName*`].nameSuffix}` `#{webCenterProfile[securityContext.userName].nameSuffix}`	Additional information appended to the user's name, such as `Esq.`, `Jr.`, and so on.
`#{webCenterProfile[`*userName*`].middleName}` `#{webCenterProfile[securityContext.userName].middleName}`	The user's middle name.
`#{webCenterProfile[`*userName*`].homePhone}` `#{webCenterProfile[securityContext.userName].homePhone}`	The user's home phone number.
`#{webCenterProfile[`*userName*`].employeeType}` `#{webCenterProfile[securityContext.userName].employeeType}`	The user's employee type classification, for example, `IC4`.
`#{webCenterProfile[`*userName*`].lastName}` `#{webCenterProfile[securityContext.userName].lastName}`	The user's surname or last name.
`#{webCenterProfile[`*userName*`].dateofBirth}` `#{webCenterProfile[securityContext.userName].dateofBirth}`	The user's birthday.
`#{webCenterProfile[`*userName*`].businessState}` `#{webCenterProfile[securityContext.userName].businessState}`	The state in which the user's home office is located.
`#{webCenterProfile[`*userName*`].homeAddress}` `#{webCenterProfile[securityContext.userName].homeAddress}`	The user's home street address.
`#{webCenterProfile[`*userName*`].businessStreet}` `#{webCenterProfile[securityContext.userName].businessStreet}`	The street on which the user's home office is located.
`#{webCenterProfile[`*userName*`].businessPostalCode}` `#{webCenterProfile[securityContext.userName].businessPostalCode}`	The user's postal or ZIP code.
`#{webCenterProfile[`*userName*`].initials}` `#{webCenterProfile[securityContext.userName].initials}`	The user's initials.

Table A-13   (Cont.) EL Expressions Relevant to People Connections (Profile)

Expression	Function
`#{webCenterProfile[`*`userName`*`].firstName}` `#{webCenterProfile[securityContext.userName].firstName}`	The user's first name.
`#{webCenterProfile[`*`userName`*`].organizationalUnit}` `#{webCenterProfile[securityContext.userName].organizationalUnit}`	The organizational unit to which the user belongs, for example, `D10`,.
`#{webCenterProfile[`*`userName`*`].wirelessAcctNumber}` `#{webCenterProfile[securityContext.userName].wirelessAcctNumber}`	The user's wireless account number.
`#{webCenterProfile[`*`userName`*`].businessPhone}` `#{webCenterProfile[securityContext.userName].businessPhone}`	The user's business telephone number.
`#{webCenterProfile[`*`userName`*`].businessCountry}` `#{webCenterProfile[securityContext.userName].businessCountry}`	The country in to which the user is assigned.
`#{webCenterProfile[`*`userName`*`].preferredLanguage}` `#{webCenterProfile[securityContext.userName].preferredLanguage}`	The user's preferred language.
`#{webCenterProfile[`*`userName`*`].displayName}` `#{webCenterProfile[securityContext.userName].displayName}`	The user's display name.
`#{webCenterProfile[`*`userName`*`].userName}` `#{webCenterProfile[securityContext.userName].userName}`	The user's actual name.
`#{webCenterProfile[`*`userName`*`].title}` `#{webCenterProfile[securityContext.userName].title}`	The user's job title.
`#{webCenterProfile[`*`userName`*`].businessEmail}` `#{webCenterProfile[securityContext.userName].businessEmail}`	The user's business email address.
`#{webCenterProfile[`*`userName`*`].organization}` `#{webCenterProfile[securityContext.userName].organization}`	The organization to which the user belongs, for example, `Sales`,.

Table A-13 (Cont.) EL Expressions Relevant to People Connections (Profile)

Expression	Function

`#{webCenterProfile[`*`userName`*`].businessMobile}`
 	
`#{webCenterProfile[securityContext.userName].businessMobile}`
	The user's cell or mobile phone number.

`#{webCenterProfile[`*`userName`*`].expertise}`
 	
`#{webCenterProfile[securityContext.userName].expertise}`
	The user's business expertise.

A.14 ELs Related to Impersonation

Table A-14 lists EL expressions relevant to WebCenter Portal Impersonation and describes the types of functionality they provide. All listed ELs apply to both portals built with WebCenter Portal and Portal Framework applications built with JDeveloper.

For more information about using these ELs, see Customizing WebCenter Portal Impersonation Security.

Table A-14 EL Expressions Relevant to Impersonation

Expression	Function
`#{WCSecurityContext.impersonationConfigured}`	Returns whether or not impersonation has been enabled for the current domain. This EL can be useful when determining if an error was caused by an impersonation session ending prematurely, or to provide an additional indicator that a session has ended.
`#{WCSecurityContext.userInImpersonationSession}`	Returns whether the current user is in an impersonation session or not. You can use this EL to protect content and render it inaccessible during an impersonation session. For example, you could map the rendered attribute of an administration taskflow on a page to this EL only rendering the taskflow if the user is not viewing the taskflow in an impersonation session.
`#{WCSecurityContext.currentImpersonator}`	This EL could be used to modify the page template to display the impersonator or render content accessible only to a particular impersonator.

A.15 EL Expressions Related to the Page Editor

Table A-15 lists EL expressions relevant to the WebCenter Portal page editor. These EL expressions can be used both in portals built with WebCenter Portal and in asset applications built with Oracle JDeveloper.

Table A-15   EL Expressions Relevant to Page Editor

Expression	Function

`#{composerContext.inEditMode}`	Determines whether the current page is in View mode or Edit mode.
	For example:
	
	``<af:outputText id="ot1"
	    value="#{composerContext.inEditMode ne
	    true ? 'View mode' : 'Edit mode'} />```
	
	This example shows an ADF Faces `outputText` component that returns `View mode` when the page is in View mode and `Edit mode` when the page is in Edit mode.

`#{changeModeBean.inEditMode}`	The `childCreation` attribute determines whether the children of popup dialogs can be viewed in Composer. By default, this attribute is set to `deferred` and users are unable to see the children of popup dialogs. Setting the value to `immediate` allows users to view the children of popup dialogs. For example: ```<af:popup childCreation="#{changeModeBean.inEditMode ? 'immediate' : 'deferred'}" />``` There is a performance impact by setting the `childrenCreation` attribute to `immediate`. If you do not need to use the popup on the page, using the default setting of `deferred` will avoid the penalty of CPU and memory usage when it is not needed. Note that the popup will be shown non-modally, thus, the application must be able to handle that fact without error.

A.15.1 Example: Using EL Expressions for the Page Editor

This section provides an example of using an EL expression to display different pages in View mode and Edit mode. Consider that you want one of your pages, `Page1`, to display another page, `Page2`, in View mode. However, while editing `Page1`, instead of `Page2`, you want to display a website, `www.example.com`. You can use the EL of the following format :

```
#{ composerContext.inEditMode ? 'http://www.example.com' : 'url_of_Page2' }
```

This EL displays the specified website when your page is in Edit mode, and shows `Page2` when in View mode.

The following steps demonstrate how to display different pages depending on the page mode.

1. Open a page named `Page1` in the page editor.

2. In Design view, in the resource catalog, click **Web Development**.

3. Click **Add** displayed next to **Web Page**.

4. Click the Edit icon for the newly added Web Page.

5. In the Component Properties dialog, on the **Display Options** tab, in the **Source** field, enter the value as per the following format:

```
#{ composerContext.inEditMode ? 'website_url' : 'url_of_Page2' }
```

6. Click **OK**.

In Composer, `Page1` should now display the website you specified. Save and close the page. `Page1` should now show `Page2`.

A.16 EL Expressions Related to Device Settings

This section lists ELs for obtaining information about device settings. For information about device settings, see Administering Device Settings in *Administering Oracle WebCenter Portal*.

You can use EL expressions to set assets (skins and page templates) targeted to specific devices. For information on using EL with device settings, see Administering Device Settings in *Administering Oracle WebCenter Portal*.

- Table A-16 lists EL expressions for obtaining information about device group mappings.

- Table A-17 lists ELs for obtaining information about devices.

- Table A-18 lists ELs for obtaining information about skins and page templates associated with device groups.

- Sample Task Flow Code for Discovering Device Attributes presents sample code that can be used in a task flow for discovering the attributes of the device used to access the portal.

For example, a page designer can use the device's attribute to control the width of the content area of a page by using the following EL:

```
#{DeviceAgent.device.attributes['display_resolution_width']}
```

Table A-16   EL Expressions Related to Device Agents

Expression	Related Java API	Description
`#{DeviceAgent}`	`DeviceAgent.getInstance()`	The root entry point to all device settings EL.
`#{DeviceAgent.device}`	`DeviceAgent.getInstance().getDevice()`	The device to which the current request is mapped.
`#{DeviceAgent.deviceGroups}`	`DeviceAgent.getInstance().getDeviceGroups()`	A list of device groups to which the current device maps. This operation returns all device groups, whether they are marked as `enabled` or not.

Table A-16   (Cont.) EL Expressions Related to Device Agents

Expression	Related Java API	Description
`#{DeviceAgent.currentScopeDeviceGroup}`	`DeviceAgent.getInstance().getCurrentScopeDeviceGroup()`	Gets the device group that the incoming request was mapped to for the given portal. This EL can help the administrator diagnose issues about why a page variant is not getting displayed for a particular device.

Table A-17   EL Expressions Relevant to Devices

Expression	Related Java API	Description
`#{DeviceAgent.device}`	`DeviceAgent.getInstance().getDevice()`	The device to which the current request is mapped.
`#{DeviceAgent.device.name}`	`DeviceAgent.getInstance().getDevice().getName()`	The name of the device. This is a system friendly identifier that can be used for internal linking.
`#{DeviceAgent.device.displayName}`	`DeviceAgent.getInstance().getDevice().getDisplayName()`	The display name of the device.
`#{DeviceAgent.device.attributes}`	`DeviceAgent.getInstance().getDevice().getAttributes()`	A map of attributes associated with a device - like the screen size, OS version, and so on. For example, the following EL returns the display resolution width that was set when the device was created: `#{DeviceAgent.device.attributes['display_resolution_width']}`

Table A-18   EL Expressions Relevant to Device Groups

Table A-18   (Cont.) EL Expressions Relevant to Device Groups

Expression	Related Java API	Descriptions
`#{DeviceAgent.currentScopeDeviceGroup}`	`DeviceAgent.getInstance().getCurrentScopeDeviceGroup()`	The device group to which the current request has been mapped. The typical flow for identifying the appropriate device group is as follows:
		1. Identify the device to which the current request belongs.
		2. Get all the device groups that have been enabled for the current portal.
		3. Identify the first device group (from the above list) to which the device belongs.
`#{DeviceAgent.currentScopeDeviceGroup.name}`	`DeviceAgent.getInstance().getCurrentScopeDeviceGroup().getName()`	The name of the device group. For example: `#{DeviceAgent.currentScopeDeviceGroup.name=='androidTablets'}`
`#{DeviceAgent.currentScopeDeviceGroup.portalPageTemplate}`	`DeviceAgent.getInstance().getCurrentScopeDeviceGroup().getPortalPageTemplate()`	Returns the GUID of the portal page template associated with this device group object. If the method returns NULL, the default page template setting for the portal is in effect. **Note:** For information on the resource ELs used to work with the obtained GUID, see ELs Related to Portal Resources.
`#{DeviceAgent.currentScopeDeviceGroup.portalSkin}`	`DeviceAgent.getInstance().getCurrentScopeDeviceGroup().getPortalSkin()`	Returns the GUID of the portal skin associated with this device group. If the method returns NULL, the default skin setting for the portal is in effect. **Note:** For information on the resource ELs used to work with the obtained GUID, see ELs Related to Portal Resources.

Table A-18 (Cont.) EL Expressions Relevant to Device Groups

Expression	Related Java API	Descriptions
`#{DeviceAgent.currentS copeDeviceGroup.defaul t}`	`DeviceAgent.getInstanc e().getCurrentScopeDev iceGroup().isDefault()`	Returns true if the device group is the default for the current device. Returns false if the device group is not the default for the device. For example, if the default device group is "Desktop Browsers" and you access the portal on an iPad, then the device group will be iPad, but it is not the default device group. In this case, the method returns false.
`#{DeviceAgent.currentS copeDeviceGroup.enable d}`	`DeviceAgent.getInstanc e().getCurrentScopeDev iceGroup().isEnabled()`	Identifies whether or not this device group has been marked as enabled for the current portal. **Note:** In this invocation, the device group will always state `true`. This EL is useful when working with all the device groups as obtained from the EL `#{DeviceAgent.deviceG roups}`

A.16.1 Sample Task Flow Code for Discovering Device Attributes

The example below lists a JSF page fragment that can be used to create a task flow for discovering device attributes. EL is used to return the attributes. The task flow may be useful for troubleshooting situations where the portal is not rendering properly on a device. By discovering the device attributes, you can take action to correct the rendering problem. Figure A-6 shows output from a task flow created with this sample code.

Example: Task Flow Code for Discovering Device Attributes

```
<?xml version='1.0' encoding='UTF-8'?>
<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">
 <af:panelGroupLayout layout="scroll" id="pgl1">
 <cust:panelCustomizable id="pc1"/>
 <f:facet name="separator">
 <af:separator id="s1"/>
 </f:facet>
 <af:panelFormLayout id="pfl1">
 <af:group id="g1">
 <af:panelLabelAndMessage id="it0" label="Current Device" labelStyle="color:#0000ff;font-
weight:bold;"/>
 <af:panelLabelAndMessage id="it1plm" label="Internal Name">
 <af:outputText id="it1" value="#{DeviceAgent.device.name}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="it2plm" label="Display Name">
```

```
 <af:outputText id="it2" value="#{DeviceAgent.device.displayName}"/>
 </af:panelLabelAndMessage>
 </af:group>
 <af:group>
 <af:panelLabelAndMessage id="it01" label="Current Device Group" labelStyle="color:#0000ff;font-
weight:bold;"/>
 <af:panelLabelAndMessage id="it3plm" label="Internal Name">
 <af:outputText id="it3" value="#{DeviceAgent.currentScopeDeviceGroup.name}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="it4plm" label="Display Name">
 <af:outputText id="it4" value="#{DeviceAgent.currentScopeDeviceGroup.displayName}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="it5plm" label="PageTemplate">
 <af:outputText id="it5"
value="#{srmContext.id[DeviceAgent.currentScopeDeviceGroup.portalPageTemplate].singleResult.displayNam
e}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="it6plm" label="Skin">
 <af:outputText id="it6"
value="#{srmContext.id[DeviceAgent.currentScopeDeviceGroup.portalSkin].singleResult.displayName}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="it7plm" label="Is Default">
 <af:outputText id="it7" value="#{DeviceAgent.currentScopeDeviceGroup.default}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="it8plm" label="Is Enabled">
 <af:outputText id="it8" value="#{DeviceAgent.currentScopeDeviceGroup.enabled}"/>
 </af:panelLabelAndMessage>
 </af:group>
 <af:group>
 <af:panelLabelAndMessage id="it9plm" label="Current Browser User Agent"
labelStyle="color:#0000ff;font-weight:bold;">
 <af:outputText id="it9" value="#{DeviceAgent.userAgent}"/>
 </af:panelLabelAndMessage>
 </af:group>
 <af:group>
 <af:panelLabelAndMessage id="pt0" label="Page Template Info" labelStyle="color:#0000ff;font-
weight:bold;"/>
 <af:panelLabelAndMessage id="it010plm" label="Expected PageTemplate:GUID">
 <af:outputText id="it010" value="#{DeviceAgent.currentScopeDeviceGroup.portalPageTemplate}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="it0010plm" label="Expected PageTemplate:Name">
 <af:outputText id="it0010"
value="#{srmContext.id[DeviceAgent.currentScopeDeviceGroup.portalPageTemplate].singleResult.displayNam
e}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="it10plm" label="Current PageTemplate:GUID">
 <af:outputText id="it10"
value="#{WCAppContext.application.siteTemplatesManager.currentSiteTemplateId}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="it10aplm" label="Current PageTemplate:Name">
 <af:outputText id="it10a"
value="#{srmContext.id[WCAppContext.application.siteTemplatesManager.currentSiteTemplateId].singleResu
lt.displayName}"/>
 </af:panelLabelAndMessage>
 </af:group>
 <af:group>
 <af:panelLabelAndMessage id="sk0" label="Skin Info" labelStyle="color:#0000ff;font-weight:bold;"/>
 <af:panelLabelAndMessage id="it0111plm" label="Expected Skin:GUID">
 <af:outputText id="it0111" value="#{DeviceAgent.currentScopeDeviceGroup.portalSkin}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="it0112plm" label="Expected Skin:Name">
```

```
 <af:outputText id="it0112"
value="#{srmContext.id[DeviceAgent.currentScopeDeviceGroup.portalSkin].singleResult.displayName}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="it111plm" label="Current Skin:GUID">
 <af:outputText id="it111" value=""/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="it112plm" label="Current Skin:Name">
 <af:outputText id="it112"
value="#{requestScope['oracle.webcenter.webcenterapp.view.shell.WebCenterShellManager.SHELLHANDLER'].a
dfFacesSkin}"/>
 </af:panelLabelAndMessage>
 </af:group>
 <af:group>
 <af:panelLabelAndMessage id="pg0" label="Page Info" labelStyle="color:#0000ff;font-weight:bold;"/>
 <af:panelLabelAndMessage id="pg10" label="Page Path">
 <af:outputText id="pg1" value="#{pageDocBean.pagePath}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="pg20" label="Page Style">
 <af:outputText id="pg2" value=""/>
 </af:panelLabelAndMessage>
 </af:group>
 <af:group>
 <af:panelLabelAndMessage id="ap90" label="Optional Attributes" labelStyle="color:#0000ff;font-
weight:bold;"/>
 <af:panelLabelAndMessage id="ap1opta" label="brand-name">
 <af:outputText id="ap1" value="#{DeviceAgent.device.attributes['brand-name']}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="ap2opta" label="device-os">
 <af:outputText id="ap2" value="#{DeviceAgent.device.attributes['device-os']}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="ap3opta" label="device-type">
 <af:outputText id="ap3" value="#{DeviceAgent.device.attributes['device-type']}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="ap4opta" label="device_default_aspect_ratio">
 <af:outputText id="ap4" value="#{DeviceAgent.device.attributes['device_default_aspect_ratio']}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="ap5opta" label="device_os-version">
 <af:outputText id="ap5" value="#{DeviceAgent.device.attributes['device_os-version']}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="ap6opta" label="device_preview_css">
 <af:outputText id="ap6" value="#{DeviceAgent.device.attributes['device_preview_css']}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="ap7opta" label="device_preview_horizontal_css">
 <af:outputText id="ap7" value="#{DeviceAgent.device.attributes['device_preview_horizontal_css']}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="ap8opta" label="device_preview_viewport_css">
 <af:outputText id="ap8" value="#{DeviceAgent.device.attributes['device_preview_viewport_css']}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="ap9opta" label="device_preview_viewport_horizontal_css">
 <af:outputText id="ap9"
value="#{DeviceAgent.device.attributes['device_preview_viewport_horizontal_css']}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="ap10opta" label="device_streaming_flv">
 <af:outputText id="ap10" value="#{DeviceAgent.device.attributes['device_streaming_flv']}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="ap11opta" label="device_streaming_preferred_http_protocol">
 <af:outputText id="ap11"
value="#{DeviceAgent.device.attributes['device_streaming_preferred_http_protocol']}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="ap12opta" label="device_streaming_preferred_protocol">
 <af:outputText id="ap12"
```

```
value="#{DeviceAgent.device.attributes['device_streaming_preferred_protocol']}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="ap13opta" label="device_video_streaming">
 <af:outputText id="ap13" value="#{DeviceAgent.device.attributes['device_video_streaming']}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="ap14opta" label="display_orientation_support">
 <af:outputText id="ap14" value="#{DeviceAgent.device.attributes['display_orientation_support']}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="ap15opta" label="display_resolution_height">
 <af:outputText id="ap15" value="#{DeviceAgent.device.attributes['display_resolution_height']}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="ap16opta" label="display_resolution_width">
 <af:outputText id="ap16" value="#{DeviceAgent.device.attributes['display_resolution_width']}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="ap17opta" label="is-wireless_device">
 <af:outputText id="ap17" value="#{DeviceAgent.device.attributes['is-wireless_device']}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="ap18opta" label="marketing-name">
 <af:outputText id="ap18" value="#{DeviceAgent.device.attributes['marketing-name']}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="ap19opta" label="model-name">
 <af:outputText id="ap19" value="#{DeviceAgent.device.attributes['model-name']}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="ap20opta" label="model-version">
 <af:outputText id="ap20" value="#{DeviceAgent.device.attributes['model-version']}"/>
 </af:panelLabelAndMessage>
 <af:panelLabelAndMessage id="ap21opta" label="icon">
 <af:outputText id="ap21" value="#{DeviceAgent.device.attributes['icon']}"/>
 </af:panelLabelAndMessage>
 </af:group>
 </af:panelFormLayout>
 </af:panelGroupLayout>
</jsp:root>
```

Figure A-6   Output from Sample Task Flow

Mobile EL Taskflow	
Current Device	
Internal Name	iPad3
Display Name	iPad (3rd Gen)
Current Device Group	
Internal Name	iOSTablets
Display Name	iOS Tablets
PageTemplate	
Skin	
Is Default	false
Is Enabled	true
Current Browser User Agent	Mozilla/5.0 (iPad; CPU OS 5_0 like Mac OS X) AppleWebKit/534.46 (KHTML, like Gecko) Version/5.1 Mobile/9A334 Safari/7534.48.3
Page Template Info	
Expected PageTemplate:GUID	
Expected PageTemplate:Name	
Current PageTemplate:GUID	gsr1402fc8c_a13d_44c5_af83_c1c6864b3196
Current PageTemplate:Name	Skyros Top Navigation
Skin Info	
Expected Skin:GUID	
Expected Skin:Name	
Current Skin:GUID	
Current Skin:Name	webcenter-skyros
Page Info	
Page Path	/oracle/webcenter/page/scopedMD/sfc98e7f3_0a0b_415f_891a_e7e47def858b/PortalHome.jspx
Page Style	
Optional Attributes	
brand-name	Apple
device-os	iOS
device-type	tablet
device_default_aspect_ratio	4:3

A.17 Utilitarian EL Expressions

Table A-19 lists utilitarian EL expressions and describes the types of functionality they provide.

Table A-19   Utilitarian EL Expressions

Expression	Function
`#{userPreferences.currentDate}`	Returns the current date in the format specified in the current user's preferences.
`#{WCTruncator['text']['numberOfChars']}`	Returns a truncation of the string specified as `text` to the number of characters specified as `numberOfChars`, followed by a trailing ellipsis, for example: `#{WCTruncator['abracadabra']['5']}` evaluates to *abrac...*
`#{facesContext.viewRoot.locale}` `#{facesContext.externalContext.requestLocale}`	Both of these expressions return the request locale (that is, the browser locale setting).
`#{adfFacesContext.skinFamily}`	Returns the current application skin family. Returns the same value as `#{requestContext.skinFamily}`.

A.18 Built-In Expressions in the Expression Editor

In the Expression Editor, when you select the option **Choose a value** you can populate the text box in the dialog with a built-in EL expression. First, select an expression category, and then select the type of information you want the expression to return. Your selection's associated EL appears in the text box at the bottom of the dialog (Figure A-7).

Figure A-7   Options Under Choose a value in the Expression Editor

This section is broken into subsections for each category. Each subsection lists and describes the ELs associated with that category.

This section includes the following subsections:

- Application Info Built-In ELs

- Asset Info Built-In ELs

- Page Info Built-In ELs

- Page Parameter Built-In ELs

- Portal Info Built-In ELs

- Portal Page Info Built-In Paths

- System Built-In ELs

- User Info Built-In ELs

- WebCenter Events Built-In ELs

A.18.1 Application Info Built-In ELs

Application Info built-in EL expressions return values related to Oracle WebCenter Portal. For example, the expression `#{WCAppContext.application.applicationConfig.title}`, returns the application name specified on the General page in WebCenter Portal administration.

See Also:

For additional EL expressions related to the application, see ELs Related to WebCenter Portal Information.

Table A-20 lists the built-in EL expressions available under the category Application Info and provides an example of the types of values they return.

Table A-20   Built-In EL Expressions Under Application Info

Option	Expression	Example of Returned Value
Current Scope	`#{serviceCtx.scope}`	`Scope[name=standards, guid=sa78185d3_9d65_49ab_ac1d_ 4809380d0b30]`
Current Page Template Scope	`#{WCAppContext.currentScope }`	`Scope[name=standards, guid=sa78185d3_9d65_49ab_ac1d_ 4809380d0b30]`
Previously Set Page Template	`#{WCAppContext.previouslySe tPageTemplate}`	`Fusion Side Navigation`
Application	`#{WCAppContext.application}`	`oracle.webcenter.webcenterapp. internal.view.shell.WCApplicat ion`
Application URL	`#{WCAppContext.applicationU RL}`	`http://host:port/webcenter`
Current WebCenter Portal URI	`#{WCAppContext.currentWebCe nterURI}`	`/oracle/webcenter/page/ scopedMD/GUID/ businessRolePages/Page3.jspx? wc.contextURL=%2Fspaces`
Application Config	`#{WCAppContext.application. applicationConfig}`	`oracle.webcenter.webcenterapp. beans.WebCenterMetadataImpl@12 3db9d`
Application Config Title	`#{WCAppContext.application. applicationConfig.title}`	`WebCenter`
Application Config Logo	`#{WCAppContext.application. applicationConfig.logo}`	`/oracle/webcenter/ webcenterapp/metadata/images/ logo.gif`
Application Config HelpPage	`#{WCAppContext.application. applicationConfig.helpPage}`	`/webcenterhelp/spaces? topic=welcome_main`
Application Config CopyrightMessage CustomValue	`#{WCAppContext.application. applicationConfig.copyright Message.customValue}`	`http://www.example.com/html/ copyright.html`
Application Config Privacy Policy URL	`#{WCAppContext.application. applicationConfig.privacyPo licyUrl}`	`http://www.example.com/html/ privacy.html`
Application Config Skin	`#{WCAppContext.application. applicationConfig.skin}`	`webcenterfx`

For example, consider that you want a link to appear in all pages of a portal that lets you copy and share the current page's URL. You can add an HTML Markup item to a page template, and add the EL that returns page URL.

These steps demonstrate how to use ELs to retrieve page URLs:

1. Open a custom page template in Edit mode.

2. In Design view, in the resource catalog, click **Web Development**.

3. Click **Add** next to **HTML Markup**.

4. Click the Edit icon for the newly added HTML Markup.

5. In the Component Properties dialog, on the **Display Options** tab, in the **Value** field, enter the following EL:

```
<a href="javascript:{
 var returnvalue =prompt( '#{pageDocBean.title}
URL','#{WCAppContext.applicationURL}/faces#{WCAppContext.currentWebCenterURI}');
}">Share Link</a>
```

6. Click **OK**.

7. Apply the page template to your portal.

A.18.2 Asset Info Built-In ELs

Asset Info built-in EL expressions return values related to WebCenter Portal resources, such as templates, styles, resource catalogs, and so on. For example, the expression `#{srmContext.id['ResourceID'].singleResult}`, returns the asset with the specified GUID.

See Also:

For additional asset-related EL expressions, see ELs Related to Assets.

Table A-21 lists the built-in EL expressions available under the category Asset Info and provides an example of the types of values each returns.

Table A-21   Built-In EL Expressions Under Asset Info

Table A-21   (Cont.) Built-In EL Expressions Under Asset Info

Option	Expression	Example of Returned Value
Asset ID	`#{srmContext[wCond['id']['like']['ResourceID']].singleResult}`	`id:gsr3396194a_3a72_44d6_90b4_57fd6efe4ff7 resourceScope:Scope[name=defaultScope, guid=s8bba98ff_4cbb_40b8_beee_296c916a23ed] resourceType:siteTemplate serviceId:oracle.webcenter.siteresources.sitetemplate category: displayName:Fusion Side Navigation displayNameKey:SITE_TEMPLATE_FUS_SIDE_STRETCH_NAV_DN description:Stretching Page Layout with Side Navigation. Use Fusion FX Skin. descriptionKey:SITE_TEMPLATE_FUS_SIDE_STRETCH_NAV_DESC contentDirectory:/oracle/webcenter/siteresources/shared metadataFile: jspx:/oracle/webcenter/webcenterapp/view/templates/WCSiteTemplateSideNavStretch.jspx pageDef:/oracle/webcenter/webcenterapp/bindings/pageDefs/oracle_webcenter_webcenterapp_view_templates_WCSiteTemplateSideNavStretchPageDef.xml iconURI: excludeFrom: usesCustomSecurity:false visible:TRUE seeded:true createdBy:system createdDate:Fri Jul 02 05:44:19 PDT 2010 modifiedBy:system resourceBundle:oracle.webcenter.sitetemplate.view.resource.SiteTemplateBundle modifiedDate:Fri Jul 02 05:44:19 PDT 2010 logoUrl:/adf/webcenter/srm_dflt_logo_qualifier.png attributes:[]`
Display Name	`#{srmContext[wCond['displayName']['like']['DisplayName']].singleResult}`	See Asset ID for an example of a returned value.
Description	`#{srmContext[wCond['description']['like']['Description']].singleResult}`	See Asset ID for an example of a returned value.

Table A-21 (Cont.) Built-In EL Expressions Under Asset Info

Option	Expression	Example of Returned Value
Creation Date	`#{srmContext[wCond['createdDate']['like']['CreationDate']].singleResult}`	See Asset ID for an example of a returned value.
Modified By	`#{srmContext[wCond['modifiedBy']['like']['UserName']].singleResult}`	See Asset ID for an example of a returned value.
Modified Date	`#{srmContext[wCond['modifiedDate']['like']['ModifiedDate']].singleResult}`	See Asset ID for an example of a returned value.
Asset Scope	`#{srmContext[wCond['resourceScope']['like']['ResourceScope']].singleResult}`	See Asset ID for an example of a returned value.

A.18.3 Page Info Built-In ELs

Page Info built-in EL expressions return values related to WebCenter Portal pages. For example, the expression `#{pageDocBean.createdBy}`, returns the user name of the person who created the current page.

Table A-22 lists the built-in EL expressions available under the category Page Info and provides an example of the types of values each returns.

Table A-22 Built-In EL Expressions Under Page Info

Option	Expression	Example of Returned Value
Title	`#{pageDocBean.title}`	`Welcome Page (BRP)`
Created By	`#{pageDocBean.createdBy}`	`j.doe`
Create Date String	`#{pageDocBean.createDateString}`	`2010-11-17T13:27:52`
Last Updated By	`#{pageDocBean.lastUpdatedBy}`	`j.smith`
Last Update Date String	`#{pageDocBean.lastUpdateDateString}`	`2010-11-17T13:27:52`
Page Path	`#{pageDocBean.pagePath}`	`/oracle/webcenter/page/scopedMD/GUID/businessRolePages/Page3.jspx`
Name	`#{pageDocBean.name}`	`Page3.jspx`
UI CSS Style	`#{pageDocBean.UICSSStyle}`	`WCSchemeCustom`

Table A-22   (Cont.) Built-In EL Expressions Under Page Info

Option	Expression	Example of Returned Value
UI Scheme BG Image	`#{pageDocBean.UISchemeBGImage}`	`#{facesContext.externalContext.requestContextPath}/adf/oracle/webcenter/page/images/clouds.png`
UI Scheme BG Color String	`#{pageDocBean.UISchemeBGColorString}`	`null`
Page Permission	`#{pageDocBean.pagePermission}`	`oracle.webcenter.page.model.security.CustomPagePermission`
Page Security Target	`#{pageDocBean.pageSecurityTarget}`	`oracle_webcenter_page_scopedMD_sa78185d3_9d65_49ab_ac1d_4809380d0b30_COIHomePageDef`

A.18.4 Page Parameter Built-In ELs

Page Parameter built-in EL expressions return values related to WebCenter Portal page parameters. The editor lists all parameters that are defined for the current page. If no parameters are defined, the Page Parameter category is not available.

For example, the built-in page style **Blog** provides five built-in page parameters (see Table A-23). The `width` parameters describe the percentage of total width allotted to each column. The `show` parameters specify whether a page header and footer are rendered. Any page may have its own custom page parameters.

Whether built-in or custom, available page parameters are exposed in the Expression Editor under the Page Parameter category.

> **See Also:**
>
> For more information about creating and using page parameters, see Adding or Modifying Page Parameters in *Building Portals with Oracle WebCenter Portal*.

Table A-23 lists the EL expressions available under the Page Parameter category for pages based on the Blog page style. It provides an example of the types of values each parameter returns.

Table A-23   Built-In EL Expressions Under Page Parameter

Option	Expression	Example of Returned Value
leftWidth	`#{bindings.leftWidth}`	25%
centerWidth	`#{bindings.centerWidth}`	50%
rightWidth	`#{bindings.rightWidth}`	25%
showHeader	`#{bindings.showHeader}`	false

Table A-23   (Cont.) Built-In EL Expressions Under Page Parameter

Option	Expression	Example of Returned Value
showFooter	`#{bindings.showFooter}`	`false`

A.18.5 Portal Info Built-In ELs

Portal Info built-in EL expressions return values related to portals (other than the Home portal). For example, the expression `#{spaceContext.space}`, returns the name of the current portal.

> **See Also:**
>
> For additional portal-related EL expressions, see ELs Related to Specific Portals.

Table A-24 lists the built-in EL expressions available under the category Portal Info and provides an example of the types of values each returns.

> **Note:**
>
> Use the EL expressions listed in Table A-24 only in the context of portals other than the Home portal.

Table A-24   Built-In EL Expressions Under Portal Info

Option	Expression	Example of Returned Value
NamedPortal	`#{spaceContext.space['portalName']}`	`oracle.webcenter.spaces.internal.model.SpaceImpl@15fc839`
CurrentPortal	`#{spaceContext.currentSpace}`	`oracle.webcenter.spaces.internal.model.SpaceImpl@d5e197`
CurrentPortal Name	`#{spaceContext.currentSpaceName}`	`standards`
NamedPortal Metadata Path	`#{spaceContext.space['portalName'].metadataPath}`	`/oracle/webcenter/space/metadata/spaces/portalName/space.xml`
CurrentPortal Metadata Path	`#{spaceContext.currentSpace.metadataPath}`	`/oracle/webcenter/space/metadata/spaces/standards/space.xml`
NamedPortal Metadata	`#{spaceContext.space['portalName'].metadata}`	`oracle.webcenter.spaces.beans.SpaceMetadataImpl@1d7320a`

Table A-24   (Cont.) Built-In EL Expressions Under Portal Info

Option	Expression	Example of Returned Value
CurrentPortal Metadata	`#{spaceContext.currentSpace.metadata}`	`oracle.webcenter.spaces.beans.SpaceMetadataImpl@1096c64`
NamedPortal Metadata DisplayName	`#{spaceContext.space['portalName'].metadata.displayName}`	`Standards`
CurrentPortal Metadata DisplayName	`#{spaceContext.currentSpace.GSMetadata.displayName}`	`Sales`
NamedPortal Metadata Icon	`#{WCPrepareImageURL[spaceContext.space['portalName'].metadata.icon]['']}`	`/webcenter-spaces-resources/oracle/webcenter/space/metadata/spacetemplate/PortalSite/images/portal_icon.png`
CurrentPortal Metadata Icon	`#{WCPrepareImageURL[spaceContext.currentSpace.metadata.icon]['']}`	`/webcenter-spaces-resources/oracle/webcenter/space/metadata/spaces/Test_Space/images/wc_prj_icon.png`
NamedPortal Metadata Description	`#{spaceContext.space['portalName'].metadata.description}`	`Central point for all standards, templates, and stylesheets for company collateral`
CurrentPortal Metadata Description	`#{spaceContext.currentSpace.GSMetadata.description}`	`Sales Force Team Site`
NamedPortal Metadata CreationDate	`#{spaceContext.space['portalName'].metadata.creationDate}`	`java.util.GregorianCalendar[time=1290103432242,areFieldsSet=true,areAllFieldsSet=false,lenient=false,zone=java.util.SimpleTimeZone[id=,offset=-28800000,dstSavings=3600000,useDaylight=false,startYear=0, startMode=0,startMonth=0,startDay=0,startDayOfWeek=0,startTime=0,startTimeMode=0,endMode=0,endMonth=0,endDay=0,endDayOfWeek=0,endTime=0,endTimeMode=0],firstDayOfWeek=1,minimalDaysInFirstWeek=1,ERA=1, YEAR=2010,MONTH=10,WEEK_OF_YEAR=?,WEEK_OF_MONTH=?,DAY_OF_MONTH=18,DAY_OF_YEAR=?,DAY_OF_WEEK=?,DAY_OF_WEEK_IN_MONTH=?,AM_PM=0,HOUR=10,HOUR_OF_DAY=10,MINUTE=3,SECOND=52,MILLISECOND=242,ZONE_OFFSET=?,DST_OFFSET=?]`

Table A-24   (Cont.) Built-In EL Expressions Under Portal Info

Option	Expression	Example of Returned Value
CurrentPortal Metadata CreationDate	#{spaceContext.currentSpace.metadata.creationDate}	`java.util.GregorianCalendar[time=1290103432242,areFieldsSet=true,areAllFieldsSet=false,lenient=false,zone=java.util.SimpleTimeZone[id=,offset=-28800000,dstSavings=3600000,useDaylight=false,startYear=0,startMode=0,startMonth=0,startDay=0,startDayOfWeek=0,startTime=0,startTimeMode=0,endMode=0,endMonth=0,endDay=0,endDayOfWeek=0,endTime=0,endTimeMode=0],firstDayOfWeek=1,minimalDaysInFirstWeek=1,ERA=1,YEAR=2010,MONTH=10,WEEK_OF_YEAR=?,WEEK_OF_MONTH=?,DAY_OF_MONTH=18,DAY_OF_YEAR=?,DAY_OF_WEEK=?,DAY_OF_WEEK_IN_MONTH=?,AM_PM=0,HOUR=10,HOUR_OF_DAY=10,MINUTE=3,SECOND=52,MILLISECOND=242,ZONE_OFFSET=?,DST_OFFSET=?]`
NamedPortal Metadata CreatedBy	`#{spaceContext.space['`*portalName*`'].metadata.createdBy}`	`j.doe`
NamedPortal Metadata Keywords	`#{spaceContext.space['`*portalName*`'].metadata.keywords}`	`standards templates style sheets collateral`
CurrentPortal Metadata Keywords	`#{spaceContext.currentSpace.metadata.keywords}`	`sales salesReports salesNews salesTeam`
NamedPortal Metadata Offline	`#{spaceContext.space['`*portalName*`'].metadata.offline}`	`false` or `true`, depending on whether the named portal is offline
CurrentPortal Metadata Offline	`#{spaceContext.currentSpace.metadata.offline}`	`false` or `true`, depending on whether the current portal is offline
NamedPortal Metadata Closed	`#{spaceContext.space['`*portalName*`'].metadata.offline}`	`false` or `true`, depending on whether the named portal is closed
CurrentPortal Metadata Closed	`#{spaceContext.currentSpace.metadata.closed}`	`false` or `true`, depending on whether the current portal is closed
NamedPortal Metadata SelfRegistration	`#{spaceContext.space['`*portalName*`'].metadata.selfRegistration}`	`false` or `true`, depending on whether self registration is enabled for the named portal
CurrentPortal Metadata SelfRegistration	`#{spaceContext.currentSpace.metadata.selfRegistration}`	`false` or `true`, depending on whether self registration is enabled for the current portal

Table A-24 (Cont.) Built-In EL Expressions Under Portal Info

Option	Expression	Example of Returned Value
NamedPortal Metadata Discoverable	`#{spaceContext.space['port alName'].metadata.discover able}`	`false` or `true`, depending on whether the named portal is discoverable
CurrentPortal Metadata Discoverable	`#{spaceContext.currentSpac e.metadata.discoverable}`	`false` or `true`, depending on whether the current portal is discoverable
NamedPortal Metadata PublishRSS	`#{spaceContext.space['port alName'].metadata.publishR SS}`	`false` or `true`, depending on whether the named portal can be published in an RSS reader
CurrentPortal Metadata PublishRSS	`#{spaceContext.currentSpac e.metadata.publishRSS}`	`false` or `true`, depending on whether the current portal can be published in an RSS reader
NamedPortal Distribution List	`#{spaceContext.space['port alName'].distributionList}`	standardsGroup@myCompany.c om
CurrentPortal Distribution List	`#{spaceContext.currentSpac e.distributionList}`	salesTeam@myCompany.com
NamedPortal Metadata CustomAttribut es	`#{spaceContext.space['port alName'].metadata.customAt tributes['attributeName']}`	Returns the value provided for the named custom attribute **Note:** If you use this EL to provide a value for a field in WebCenter Portal Administration, clicking the **Test** button may return a blank value. However, the value will resolve correctly in the running portal.
CurrentPortal Metadata CustomAttribut es	`#{spaceContext.currentSpac e.metadata.customAttribute s['attributeName']}`	Returns the value provided for the named custom attribute **Note:** If you use this EL to provide a value for a field in WebCenter Portal Administration, clicking the **Test** button may return a blank value. However, the value will resolve correctly in the running portal.
CurrentPortal Metadata Logo	`#{WCPrepareImageURL[spaceC ontext.currentSpace.metada ta.logo]['']}`	/webcenter-spaces- resources/oracle/ webcenter/space/metadata/ spacetemplate/PortalSite/ images/portal_logo.png
CurrentPortal Role - Portal Manager	`#{WCSecurityContext.userIn ScopedRole['Moderator']}`	`false` or `true`, depending whether any users in the current scope are assigned the role `Portal Manager`

Table A-24   (Cont.) Built-In EL Expressions Under Portal Info

Option	Expression	Example of Returned Value
CurrentPortal Role - Participant	#{WCSecurityContext.userInScoped Role['Participant']}	false or true, depending whether any users in the current scope are assigned the role Participant
CurrentPortal Role - Viewer	#{WCSecurityContext.userInScoped Role['Viewer']}	false or true, depending whether any users in the current scope are assigned the role Viewer
CurrentPortal Role - Custom	`#{WCSecurityContext.userIn ScopedRole['`*CustomRole*`']}`	`false` or `true`, depending on if any users in the current scope are assigned the named custom role
NamedPortal Children	`#{spaceContext.space['`*port alName*`'].children}`	`[oracle.webcenter.spaces.i nternal.model.SpaceImpl@57 05f4, oracle.webcenter.spaces.in ternal.model.SpaceImpl@147 d658]` **Note:** The braces ([]) denote a set.
CurrentPortal Children	`#{spaceContext.currentSpac e.children}`	`[oracle.webcenter.spaces.i nternal.model.SpaceImpl@57 05f4, oracle.webcenter.spaces.in ternal.model.SpaceImpl@147 d658]` **Note:** The braces ([]) denote a set.
NamedPortal Parent	`#{spaceContext.space['`*port alName*`'].parent}`	`oracle.webcenter.spaces.in ternal.model.SpaceImpl@15f c839`
CurrentPortal Parent	`#{spaceContext.currentSpac e.parent}`	`oracle.webcenter.spaces.in ternal.model.SpaceImpl@15f c839`
NamedPortal Security Parent	`#{spaceContext.space['`*port alName*`'].securityParent}`	`oracle.webcenter.spaces.in ternal.model.SpaceImpl@139 2793`
CurrentPortal Security Parent	`#{spaceContext.currentSpac e.securityParent}`	`oracle.webcenter.spaces.in ternal.model.SpaceImpl@139 2793`
All visible root level spaces	`#{spaceContext.spacesQuery .unionOf['ALL_QUERIABLE,DI SCOVERABLE,PUBLIC_ACCESSIB LE,ALLUSER_ACCESSIBLE,USER _JOINED,USER_MODERATED'].s hape['ROOT_LEVEL']}`	`oracle.webcenter.spaces.qu ery.SpacesQueryParameters@ ae34c8`

Table A-24   (Cont.) Built-In EL Expressions Under Portal Info

Option	Expression	Example of Returned Value
All children and descendent portals for specified Parent Portal GUID	`#{spaceContext.spacesQuery .parentSpaceGuid['`*`portalGu`* *`id`*`'].shape['RECURSIVE_FLAT TENED']}`	oracle.webcenter.spaces.query.Spac esQueryParameters@4da22f
Immediate Subportals for specified Parent Portal name	`#{spaceContext.spacesQuery .parentSpaceName['`*`portalGu`* *`id`*`'].shape['ROOT_LEVEL']}`	oracle.webcenter.spaces.query.Spac esQueryParameters@1089971
List of Portals matching given Portal Names	`#{spaceContext.spacesQuery .containingNames['`*`portal1,`* *`portal2,portal3`*`']}`	`oracle.webcenter.spaces.qu ery.SpacesQueryParameters@ 68e467`
List of Portals matching given Portal GUID	`#{spaceContext.spacesQuery .containingGuids['`*`portalGu`* *`id1, portalGuid2`*`']}`	`oracle.webcenter.spaces.qu ery.SpacesQueryParameters@ 1c56009`
List of Portals sorted based on criteria	`#{spaceContext.spacesQuery .unionOf.ALL_QUERIABLE.sor t['sp.lastUpdateDate'] ['desc']}`	`oracle.webcenter.spaces.qu ery.SpacesQueryParameters@ 6c3def`
List of Portals based on search pattern	`#{spaceContext.spacesQuery .unionOf.ALL_QUERIABLE.sea rch['`*`searchKeyword`*`']}`	`oracle.webcenter.spaces.qu ery.SpacesQueryParameters@ 19b2cf3`
List of Portals with limited page size	`#{spaceContext.spacesQuery .unionOf.USER_JOINED.items PerPage[`*`n`*`].pageNumber[`*`n`*`]}`	`oracle.webcenter.spaces.qu ery.SpacesQueryParameters@ 4f8818`
List of portals matching the where clause	`#{spaceContext.spacesQuery .unionOf.USER_JOINED.wher e[wCond['sp.createdBy'] ['=']['userName']]}`	`oracle.webcenter.spaces.qu ery.SpacesQueryParameters@ 1ba816d`

A.18.6 Portal Page Info Built-In Paths

Rather than EL expressions, the selections under Portal Page Info are directory paths to different application pages. Use them, for example, to provide navigation information to a specified page.

Table A-25 lists the built-in directory paths available under the category Portal Page Info and provides an example of the types of values each returns. The number of values in the dropdown list depends on the template used and the pages created by users.

Table A-25   Built-In Paths Under Portal Page Info

Table A-25   (Cont.) Built-In Paths Under Portal Page Info

Option	Path	Destination
PortalSiteHome.jspx	`/faces/oracle/webcenter/page/scopedMD/se2dff4fb_202a_4f4c_b56d_c7cf59ad1c04/PortalSiteHome.jspx`	Path to the Home page of the current portal
GroupSpaceDocLibMainView.jspx	`/faces/oracle/webcenter/page/scopedMD/s8bba98ff_4cbb_40b8_beee_296c916a23ed/businessRolePages/GroupSpaceDocLibMainView.jspx`	Path to the Documents page in the current portal
ForumMainView.jspx	`/faces/oracle/webcenter/page/scopedMD/s8bba98ff_4cbb_40b8_beee_296c916a23ed/businessRolePages/ForumMainView.jspx`	Path to Discussions page in the current portal
AnnouncementsMainView.jspx	`/faces/oracle/webcenter/page/scopedMD/s8bba98ff_4cbb_40b8_beee_296c916a23ed/businessRolePages/AnnouncementsMainView.jspx`	Path to Announcements page in current portal
ListsMainView.jspx	`/faces/oracle/webcenter/page/scopedMD/s8bba98ff_4cbb_40b8_beee_296c916a23ed/businessRolePages/ListsMainView.jspx`	Path to Lists page in current portal
EventsMainView.jspx	`/faces/oracle/webcenter/page/scopedMD/s8bba98ff_4cbb_40b8_beee_296c916a23ed/businessRolePages/EventsMainView.jspx`	Path to Events page in current portal
GroupSpaceActivityStreamMainView.jspx	`/faces/oracle/webcenter/page/scopedMD/s8bba98ff_4cbb_40b8_beee_296c916a23ed/businessRolePages/GroupSpaceActivityStreamMainView.jspx`	Path to Activity Stream page in current portal

A.18.7 System Built-In ELs

System built-in EL expressions return values related to system settings, such as the user name of the current user or the current user's specified locale. For example, the expression `#{securityContext.userName}`, returns the name of the currently

logged in user. For example, if user j.doe is viewing a page where this expression is used, the returned value in j.doe's view is `j.doe`. If user j.smith is looking at the same page, the value returned in j.smith's view is `j.smith`.

> **See Also:**
>
> For additional utilitarian EL expressions, see Utilitarian EL Expressions.

Table A-26 lists the built-in EL expressions available under the category System and provides an example of the types of values each returns.

Table A-26   Built-In EL Expressions Under System

Option	Expression	Example of Returned Value
User	`#{securityContext.userName }`	The current user's user name
Locale	`#{facesContext.externalCon text.requestLocale}`	The current user's specified locale This may be taken from application or portal-level settings or user Preferences, whichever is taking precedence. For example: `en_US`

A.18.8 User Info Built-In ELs

User Info built-in EL expressions return values related to a given user's Profile. For example, the expression `#{webCenterProfile[securityContext.userName].description}`, returns the content of the About Me attribute of the current user's Profile.

> **See Also:**
>
> For additional user-related expressions, see ELs Related to People Connections.

Table A-27 lists the built-in EL expressions available under the category User Info and provides an example of the types of values each returns.

Table A-27   Built-In EL Expressions Under User Info

Option	Expression	Example of Returned Value
About Me	`#{webCenterProfile[securit yContext.userName].descrip tion}`	Returns the content of the About Me attribute of the current user's Profile
Username	`#{webCenterProfile[securit yContext.userName].userNam e}`	The user name specified in the current user's Profile

Table A-27   (Cont.) Built-In EL Expressions Under User Info

Option	Expression	Example of Returned Value
First Name	`#{webCenterProfile[securityContext.userName].firstName}`	The current user's first name
Middle Name	`#{webCenterProfile[securityContext.userName].middleName}`	The current user's middle name
Last Name	`#{webCenterProfile[securityContext.userName].lastName}`	The current user's last name
Initials used	`#{webCenterProfile[securityContext.userName].initials}`	The current user's initials
Display Name	`#{webCenterProfile[securityContext.userName].displayName}`	The current user's display name
Email	`#{webCenterProfile[securityContext.userName].businessEmail}`	The current user's business email address
Department	`#{webCenterProfile[securityContext.userName].department}`	The current user's department
Job Title/ Designation	`#{webCenterProfile[securityContext.userName].title}`	The current user's job title or designation
Manager	`#{webCenterProfile[securityContext.userName].managerDisplayName}`	The current user's manager's display name
Time zone	`#{webCenterProfile[securityContext.userName].timeZone}`	The time zone associated with the current user
Employee type	`#{webCenterProfile[securityContext.userName].employeeType}`	The current user's employee type
Employee Number	`#{webCenterProfile[securityContext.userName].employeeNumber}`	The current user's employee number
Preferred Language Code	`#{webCenterProfile[securityContext.userName].preferredLanguage}`	The current user's preferred language
Organization	`#{webCenterProfile[securityContext.userName].organization}`	The company organization to which the current user belongs

Table A-27   (Cont.) Built-In EL Expressions Under User Info

Option	Expression	Example of Returned Value
Organizational Unit	#{webCenterProfile[securityContext.userName].organizationalUnit}	The unit of the company organization to which the current user belongs
Expertise	#{webCenterProfile[securityContext.userName].expertise}	The current user's expertise; for example, Application, system, and database administration
Business Fax	#{webCenterProfile[securityContext.userName].businessFax}	The current user's business fax number
Business Mobile Number	#{webCenterProfile[securityContext.userName].businessMobile}	The current user's business mobile or cell phone number
Business Phone	#{webCenterProfile[securityContext.userName].businessPhone}	The current user's business phone number
Business Pager Number	#{webCenterProfile[securityContext.userName].businessPager}	The current user's business pager number
Business Street	#{webCenterProfile[securityContext.userName].businessStreet}	The street address of the current user's business office
Business City	#{webCenterProfile[securityContext.userName].businessCity}	The city where the current user's business office is located
Business State	#{webCenterProfile[securityContext.userName].businessState}	The state where the current user's business office is located
Business PO Box	#{webCenterProfile[securityContext.userName].businessPOBox}	The current user's business Post Office Box number
Business Postal Code	#{webCenterProfile[securityContext.userName].businessPostalCode}	The postal or ZIP code of the current user's business office
Business Country	#{webCenterProfile[securityContext.userName].businessCountry}	The country where the current user's business office is located
Home Address	#{webCenterProfile[securityContext.userName].homeAddress}	The current user's home address
Home phone	#{webCenterProfile[securityContext.userName].homePhone}	The current user's home phone number

Table A-27   (Cont.) Built-In EL Expressions Under User Info

Option	Expression	Example of Returned Value
Date of Birth	`#{webCenterProfile[securit yContext.userName].dateofB irth}`	The current user's date of birth
Maiden Name	`#{webCenterProfile[securit yContext.userName].maidenN ame}`	The current user's maiden name
Date of hire	`#{webCenterProfile[securit yContext.userName].dateofH ire}`	The date the current user was hired by the company
Name Suffix	`#{webCenterProfile[securit yContext.userName].nameSuf fix}`	Titles or credentials appended to the current user's name, for example, `MD`, `Esq`, and so on
Wireless Account Number	`#{webCenterProfile[securit yContext.userName].wireles sAcctNumber}`	The account number for the current user's wireless internet account

A.18.9 WebCenter Events Built-In ELs

WebCenter Events built-in EL expressions return values related to document-related events, such as the name of the selected document, the document's creator, the date the document was last modified, and so on. For example, the expression `#{wcEventContext.events.WebCenterResourceSelected.name}`, returns the name of the currently selected document.

Table A-28 lists the built-in EL expressions available under the category WebCenter Events and provides an example of the types of values each returns.

Table A-28   Built-In EL Expressions Under WebCenter Events

Option	Expression	Example of Returned Value
Selected User Name	`#{wcEventContext.events.We bCenterUserSelected.UserNa me}`	The user name of the currently selected user
Selected Document Name	`#{wcEventContext.events.We bCenterResourceSelected.na me}`	The name of the currently selected document
Selected Document Creator	`#{wcEventContext.events.We bCenterResourceSelected.cr eatedBy}`	The name of the user who created the currently selected document
Selected Document Last Modified By	`#{wcEventContext.events.We bCenterResourceSelected.la stModifiedBy}`	The date the selected document was last modified

A.19 Desupport of Freeform JPQL WHERE and SORT Clauses in Portal Queries

In WebCenter Portal, if you have introduced an EL expression for querying portals based on a JPQL `WHERE` clause, you should know that the freeform JPQL `WHERE` clause is desupported. Instead, use a structured `WHERE` clause. Freeform `SORT` clauses are also desupported. Instead, specify sort criteria on a portal query in a structured way.

For example, suppose that you have used the following syntax to form a `WHERE` clause on a portal query:

```
#{spaceContext.spacesQuery.unionOf['ALL_QUERIABLE'].whereClause['sp.keywords like
\'forquery%\' and not
(sp.discoverable is null or sp.discoverable =\'N\')'].listSpaces}
```

Going forward, you must use the following syntax to form a `WHERE` clause:

```
#{spaceContext.spacesQuery.unionOf['ALL_QUERIABLE'].where[wCond['sp.keywords']
['like']['forquery']['and'][wCond['sp.discoverable']
['is']['null']['or'][wCond['sp.discoverable']['=']['N']]['not']]].listSpaces}
```

Where:

- `wCond` is an atomic condition that can be expressed as `wCond[p][op][v]`.

  Where:

  - `p` is a portal attribute (a field in `WcSpaceHeader`, see the following bullet list).

  - `op` is one among a restricted set of JPQL comparison operators (`'='`, `'like'`, `'!='`, `'<'`, `'>'`, `'<='`, `'>='`, `'is'`).

  - `v` is a string whose interpretation depends on the operator `op`.

    If the operator is `is`, `v` can be the string `'null'` or `'not null'`, and the condition is understood as the attribute being checked for null or not null respectively in JPQL. If the operator is anything else, `v` is interpreted as a literal, and the condition is understood to be the attribute being compared with the literal value.

  You can use any field defined in `WcSpaceHeader` in the `WHERE` clause, including:

  - `closed`

  - `createDate`

  - `createdBy`

  - `description`

  - `discoverable`

  - `displayName`

  - `keywords`

  - `icon`

  - `logo`

- – lastUpdateDate

- – selfSubEnabled

- – rssEnabled

- – spaceGuid

- – spaceId

- – spaceOffline

- – spacePublic

- – spacesType

- – spacesUser

- – updatedBy

- – version

- – parentGuid

- – securityParentGuid

- – ancestorPath

- – seeded

- – groupSpaceMemCount

- sp represents the JPA entity for a portal, `WcSpaceHeader`.

  For example, you can use the following condition to create a filter for all portals created by users having user names starting with `monty` on which self subscription is enabled:

  ```
  sp.createdBy like 'monty%' and not (sp.selfSubEnabled is null or
  sp.selfSubEnabled ='N')
  ```

- and, or, not (along with `wCond`) are types of conditions.

  Where:

  - – and represents a conjunction of two conditions. You can express a conjunction condition as `cond1['and'][cond2]` where `cond1` and `cond2` are the expressions for the two conditions being conjoined.

  - – or represents a disjunction condition. You can express a disjunction condition as `cond1['or'][cond2]` where `cond1` and `cond2` are the expressions for the two conditions being disjoined.

  - – not represents a negation condition. You can express a negation condition as `cond['not']` where `cond` is the expression for the condition being conjoined.

For example, the following EL returns a list of all portals (to which the current user has access) that are created by users having names starting with `monty` and on which self subscription is enabled:

```
#{spaceContext.spacesQuery.unionOf.ALL_QUERIABLE.where[wCond['sp.createdBy']
['like']['monty%'] ['and'] [wCond['sp.selfSubEnabled']['is']['null'] ['or']
[wCond['sp.selfSubEnabled']['=']['N']] ['not'] ]].listSpaces}
```

Now imagine that you have used the following syntax to specify SORT criteria on a portal query:

```
#{spaceContext.spacesQuery.unionOf['ALL_QUERIABLE'].sortCriteria
['sp.discoverabledesc, sp.keywords'].listSpaces}
```

Going forward, you must use the following syntax to specify SORT criteria on a portal query:

```
#{spaceContext.spacesQuery.unionOf['ALL_QUERIABLE'].sort['sp.discoverable']
['desc'].sort['sp.keywords']['asc'].listSpaces}
```

Where:

- sp represents the JPA entity for a portal, WcSpaceHeader (see previous bullet list for fields defined in WcSpaceHeader).

- desc and asc define sort order, descending and ascending, respectively.

For example, the following query returns a list of portals to which the user has access and sorts the result set on the fields discoverable and lastUpdateDate.

```
#{spaceContext.spacesQuery.unionOf['ALL_QUERIABLE'].sort['sp.discoverable']
['asc'].sort['sp.lastUpdateDate']['desc'].listSpaces}
```

Oracle *strongly* recommends that you convert all existing freeform JPQL WHERE and SORT clauses to the new syntax.

If for any reason you must retain the existing syntax, your WebCenter Portal administrator must explicitly enable support for freeform JPQL in the webcenter-config.xml.xml file by setting the following global flag:

```
oracle.webcenter.spaces.query.DIRECT_JPQL_CLAUSE_SUPPORTED
```

> **Note:**
>
> The webcenter-config.xml.xml file is the customization document for webcenter-config.xml.

Set this flag to true for continued support of legacy freeform WHERE and SORT clauses. When this option is set to true, WebCenter Portal log files will contain warnings.

To set this flag in webcenter-config.xml.xml:

1. Export the latest webcenter-config.xml.xml from MDS.

   For example:

   ```
   exportMetadata(application='webcenter', server='WC_Portal', toLocation='/tmp/
   mydata',docs='/oracle/webcenter/webcenterapp/metadata/mdssys/cust/site/webcenter/
   webcenter-config.xml.xml')
   ```

2. Open the file in a text editor and modify settings, as required.

   For example:

   ```
   <mds:insert parent="webcenter(xmlns(webcenter=http://
        xmlns.oracle.com/webcenter/webcenterapp))/webcenter:custom-attributes"
        position="last">              <attribute name=
                   "oracle.webcenter.spaces.query.DIRECT_JPQL_CLAUSE_SUPPORTED"
   ```

```
                    xmlns="http://xmlns.oracle.com/webcenter/webcenterapp">
                        <description/>
                        <type>java.lang.String</type>
                        <value>true</value>
                        <visible/>
            </attribute>
    </mds:insert>
```

3. Save and close `webcenter-config.xml.xml`.

4. Import the updated `webcenter-config.xml.xml` file to WebCenter Portal.

   For example:

```
importMetadata(application='webcenter', server='WC_Portal', fromLocation='/tmp/
mydata', docs='/oracle/webcenter/webcenterapp/metadata/mdssys/cust/site/
webcenter/webcenter-config.xml.xml')
```

WebCenter Portal Accessibility Features

This appendix describes the accessibility features of WebCenter Portal.

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 Portal. For general information about creating accessible ADF Faces pages, see Developing Accessible ADF Faces Components and Pages in *Developing Web User Interfaces with Oracle ADF Faces*. For information about accessibility features in JDeveloper, select the JDeveloper Accessibility node under JDeveloper Basics in the online help table of contents.

This appendix includes the following topics:

- Generating Accessible HTML

- Accessibility Features at Runtime

- Accessibility Considerations for Portlets

B.1 Generating Accessible HTML

WebCenter Portal 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 the attributes listed in Table B-1.

Table B-1   Accessibility Attributes for WebCenter Portal's 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.

Table B-1   (Cont.) Accessibility Attributes for WebCenter Portal's Composer Components

Component	Accessibility Attributes
`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.

B.2 Accessibility Features at Runtime

When you give users the ability to customize a page at runtime, you must ensure that any customizations are 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 Portal components, see Table B-1. For a list of accessibility-related attributes for other components, see Developing Accessible ADF Faces Components and Pages in *Developing Web User Interfaces with Oracle ADF Faces.*

B.3 Accessibility Considerations for Portlets

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

Note:

Portlets created using the Oracle JSF Portlet Bridge have the `requireIFrame` container runtime option set to `true` because, due to JavaScript issues, these portlets are too complex to render directly on Oracle ADF pages.

WebCenter Portal provides an optional attribute in the `adf:portlet` tag called `renderPortletInIFrame`. You can set this attribute to `false` to avoid ever using IFrames.

C

Using the WebCenter Portal REST APIs

This chapter contains the following topics:

- Introduction to REST
- Understanding the Username-Based Security Token Encryption
- Benefits of Using REST
- Introduction to the WebCenter Portal REST APIs
- Understanding the Link Model
- Understanding Items Hypermedia
- Navigating Hypermedia Using HTTP
- Security Considerations for WebCenter Portal REST APIs
- Security Considerations for CMIS REST APIs
- Understanding Common Types
- Managing Caches
- Configuring a Proxy Server
- WebCenter Portal REST API Examples
- Using the People Connections REST APIs
- Content Management REST API
- Using the Events REST API
- Using the Tags REST APIs
- Using the Discussions REST API
- Using the Lists REST API
- Using the Activity Graph REST APIs
- Using the Search REST APIs

C.1 Introduction to REST

REST (REpresentational State Transfer) is an architectural style for making distributed resources available through a uniform interface that includes uniform resource identifiers (URIs), well-defined operations, hypermedia links, and a constrained set of

media types. Typically, these operations include reading, writing, editing, and removing, and media types include JSON and XML/ATOM.

REST commands use standard HTTP methods as requests to point to the resource being used. Every request returns a response, indicating the status of the operation. If the request results in an object being retrieved, created, or updated, the response includes a standard representation of that object.

REST supports multiple clients, both from client machines and other servers, and it can be used from just about any client or development technology, including Java, JavaScript, Ruby on Rails, PHP, .Net, and so on.

> **Tip:**
>
> For a good general introduction to REST, see the Wikipedia article Representational State Transfer at `http://en.wikipedia.org/wiki/Representational_State_Transfer`.

REST is typically used with Rich Internet Applications (RIA) that are client-side scripted and require the ability to interact with data from a server-side application. For example, the WebCenter iPhone App uses WebCenter Portal REST APIs to interact with a WebCenter Portal application. The native iPhone client is written in Objective-C, and the REST APIs enable the client to send and retrieve application data.

C.2 Understanding the Username-Based Security Token Encryption

To provide additional security, every URI, for both `href` and `template` attributes, includes a security token parameter that is based on the authenticated username (a utoken).

The security generation algorithm uses a randomly generated "salt" along with the username. The salt ensures that if any of the parameters used to perform encryption become compromised, existing tokens can be invalidated and new ones generated. Most importantly, because the username is used as part of the user token generation algorithm, the salt prevents having to change all user names in the event of a compromise.

The salt is stored in the Credential Store Framework (CSF) in the map `o.webcenter.jf.csf.map` against key `user.token.salt`. You can change the value of this key (and other keys used for token encryption) by accessing CSF in Oracle Enterprise Manager.

> **Caution:**
>
> If you change the encryption key values, all existing username based security tokens will immediately become invalid. Only change these values under extraordinary circumstances, like an algorithm parameter compromise.

C.3 Benefits of Using REST

Many excellent articles have been published about REST and the benefits of the RESTful style of software architecture. Some of these benefits include:

- Using "Hypermedia As The Engine Of Application State" (HATEOAS) links to access the REST API helps promote API robustness. Since the clients only use URIs

returned from the server, if the server changes the URI format, the client will continue to function properly. See also Understanding the Link Model.

- Using the standard HTTP protocol allows network infrastructure to cache REST requests where appropriate, reducing load on both the client and server. See also Managing Caches .

- Stateless REST requests allow each request to be served by any number of different servers, helping with scalability.

- Using the standard HTTP protocol allows a wide variety of clients to interact with the REST APIs without requiring specialized libraries.

C.4 Introduction to the WebCenter Portal REST APIs

In addition to enabling mobile access, WebCenter Portal REST APIs allow you to take advantage of Web 2.0 technologies like Ajax, JavaScript, and JSON to create rich, interactive browser-based user interfaces and to access and modify WebCenter Portal data. In general, the WebCenter REST commands provide a more natural and easy-to-use alternative to a SOAP-style Web services approach. For more information, see REST API for Oracle WebCenter Portal.

Table C-1 describes the Oracle WebCenter Portal REST APIs provided for WebCenter Portal: Services features.

Table C-1   Summary of WebCenter Tools and Services Features Supported by REST APIs

REST API	Description	Section
Discussions	Enable a client to post, read, update, and delete discussion forums, topics, and messages.	Using the Discussions REST API
Lists	Enables a client to browse all the lists associated with a named portal; search list columns given a search term; create new lists; add, update, and remove list rows; and similar sorts of list-related tasks	Using the Lists REST API
People Connections	Enable a client to view profile data; manage connection lists, feedback, and messages; create new activities and view activities for users, lists, and portals.	Using the People Connections REST APIs
WebCenter Portal	Enable a client to retrieve portal metadata and view, create, update, and delete portal lists and list items. You can also retrieve portal membership information.	Introduction to the WebCenter Portal REST APIs
Content Management	Uses the CMIS (Content Management Interoperability Services) RESTful server binding to provide access to the CM VCR (Content Management Virtual Content Repository).	Content Management REST API
Activity Graph	Enables you to retrieve recommendations for connections, portals, and items using the underlying Activity Graph engine.	Using the Activity Graph REST APIs

Table C–1   (Cont.) Summary of WebCenter Tools and Services Features Supported by REST APIs

REST API	Description	Section
Events	Lets you access calendar events associated with a named portal.	Using the Events REST API
Feedback	Enables a client to create, read, and delete feedback in a social networking application.	See Using the Lists REST API
Search	Lets you post, read, update, and delete searches. You can specify keywords and the scope of the search; for example, the iPhone could search for "smith" in all portals, documents and wiki pages.	Using the Search REST APIs
Tags	Enables a client to read, post, update, and delete tags and tagged items.	Using the Tags REST APIs
Navigation	Use the navigation REST APIs to create your own interface for displaying navigations. **Note:** The navigation REST APIs do not share the resource index described in this chapter.	Using the Tags REST APIs

---

Note:

XSD files for the following URNs:

```
urn:oracle:webcenter:activities:stream -> activitystream.xsd
urn:oracle:webcenter:messageBoard -> wall.xsd
urn:oracle:webcenter:people -> people.xsd
urn:oracle:webcenter:people:invitations -> people.xsd
urn:oracle:webcenter:people:person -> people.xsd
urn:oracle:webcenter:search:results -> search.xsd
urn:oracle:webcenter:spaces -> spaces.xsd
```

can be found in your `<WCP_ORACLE_HOME>/webcenter/schemas/` directory.

C.5 Understanding the Link Model

Hypermedia is at the core of two of the most successful Web-based formats: HTML and ATOM. HTML and ATOM allow consumers to navigate to other hypermedia documents through links–for example, clicking on a link to go to a news article.

Hypermedia drives the RESTful application state (known as HATEOAS: Hypermedia As The Engine Of Application State).

> **Note:**
>
> HATEOAS analogy to define application state:
>
> Suppose you are completing your taxes in your favorite browser. You finish entering your W-2 data and move on to deductions when the browser crashes. The state you lost—the fact that you were on deductions and still needed to enter data—is the application state; not the W-2 data entered (that is, change states from the current state).
>
> HATEOAS dictates that *this* state—the application state—be captured wholly in hypermedia. Application state is where you are in the application, not what data you've entered into the application. One of the benefits of this approach is that it simplifies the client and server, because they do not need to be aware of the state they are in. The link contains all the state information necessary to process the request, so, when the browser restarts and returns to the link, the user will be at the same place in the tax process.

Given a set of top-level URI entry points to a RESTful service, all interactions beyond those entry points are driven by hypermedia links returned in response representations. This link-centered approach helps keep the client from becoming too tightly coupled to the server URLs. The client is using URLs given to it by the server, therefore the client code does not break if the server URLs change format.

Understanding this link model helps you understand how to use the data the service returns to navigate the REST APIs.

This section describes the hypermedia link model used by WebCenter's RESTful services. It includes the following topics:

- Using the Resource Index
- Anatomy of a Link

C.5.1 Using the Resource Index

In WebCenter, the *Resource Index* is your starting point for all authenticated access. The Resource Index provides access to the set of top-level URI entry points. It provides the way in to all the available WebCenter RESTful services. The Resource Index URI is the only URI that you need to know.

> **Tip:**
>
> A REST client is helpful for generating custom REST requests. For example, a Firefox RESTClient add-on is available at:
>
> ```
> http://restclient.net/
> ```
>
> Other similar REST clients can also be easily obtained.

The WebCenter Resource Index URI is:

```
http://host:port/rest/api/resourceIndex
```

> **Note:**
>
> Access to the Resource Index always requires authentication; however, you can (optionally) access the CMIS resource entry point anonymously using the following URI:
>
> ```
> http://host:port/rest/api/cmis/repository
> ```
>
> See also Security Considerations for CMIS REST APIs and Security Considerations for WebCenter Portal REST APIs

The first step in using the WebCenter Portal REST APIs is to send a GET request to the Resource Index. The response varies depending on the services available and the media type of the request. The example below shows how the response might look if you made an Ajax request using JavaScript (and possibly a client-side scripting library, such as Dojo) to retrieve the JSON data for the Resource Index. Note that this is an abridged sample response and does not include all of the links actually present in a real response.

Example: Response to a GET on the Resource Index

```
{
  "resourceType": "urn:oracle:webcenter:resourceindex",
  "links": [
    {
      "template": "opaque-template-uri",
      "resourceType": "urn:oracle:webcenter:messageBoard",
      "href": "opaque-uri",
      "capabilities": "urn:oracle:webcenter:read"
    },
    {
      "resourceType": "urn:oracle:webCenter:cmis",
      "href": "opaque-uri",
      "capabilities": "urn:oracle:webcenter:read"
    },
    {
      "template": "opaque-template-uri",
      "resourceType": "urn:oracle:webcenter:discussions:forums",
      "href": "opaque-uri",
      "capabilities": "urn:oracle:webcenter:read"
    },
    {
      "resourceType": "urn:oracle:webcenter:resourceindex",
      "rel": "self",
      "href": "http://host:port/rest/api/resourceIndex",
      "capabilities": "urn:oracle:webcenter:read"
    },
    {
      "template": "opaque-template-uri",
      "resourceType": "urn:oracle:webcenter:activities:stream",
      "href": "opaque-uri",
      "capabilities": "urn:oracle:webcenter:read"
    },
    {
      "template": "opaque-template-uri",
      "resourceType": "urn:oracle:webcenter:people:person",
      "capabilities": "urn:oracle:webcenter:read"
    },
    {
      "template": "opaque-template-uri",
```

```
      "resourceType": "urn:oracle:webcenter:feedback",
      "href": "opaque-uri",
      "capabilities": "urn:oracle:webcenter:read"
    },
    {
      "template": "opaque-template-uri",
      "resourceType": "urn:oracle:webcenter:spaces",
      "href": "opaque-uri",
      "capabilities": "urn:oracle:webcenter:read"
    },
    {
      "template": "opaque-template-uri",
      "resourceType": "urn:oracle:webcenter:people",
      "href": "opaque-uri",
      "capabilities": "urn:oracle:webcenter:read"
    }
  ]
}
```

By interpreting the links returned in the Resource Index data, you can retrieve the URI entry point for an individual service by locating the URI for the resource type you want to use. You can then continue navigating through the hypermedia until you can perform the required operation. The following example shows a method that locates a URI given the Resource Index JSON data.

Example: Locating the URI for a Particular Service in the Resource Index

```
/* Parse the resourceIndex to find the specified URL and
 * return it.
 *
 * @Param jsonData the JSON data retrieved from calling
 *        the /rest/api/resourceIndex URL.
 * @Param strResourceType the resource type of the URL
 *        you want to retrieve from the resourceIndex data.
 *        E.g., 'urn:oracle:webcenter:activities:stream'
 */
function getResourceURL(jsonData, strResourceType)
{
  // Using the HATEOAS model, we browse the returned links
  // looking for the one with the correct resource type.
  for (var i = 0; i < data.links.length; i++) {
    if (data.links[i].resourceType == strResourceType) {
      return data.links[i].href;
    }
  }
}
```

C.5.2 Anatomy of a Link

The `resourceType`, `rel`, and `capabilities` attributes of the hypermedia link provide metadata that enable clients to determine which URI (`href` or `template`) to use, without having to parse the URIs directly. The URIs are opaque—the metadata determines which link is useful in a given circumstance.

The following examples show the anatomy of a hypermedia link as XML and in JSON document fragments, respectively.

Example: Link in an XML Document Fragment

```
<links>
  <link href="opaque-URI"
        template="opaque-template-URI (optional)"
```

```
            rel="rel-name"
            title="human-readable-title (optional)"
            type="media-type (optional)"
            resourceType="resource-type"
            capabilities="operation"/>
  ...repeat as needed...
</links>
```

Example: Link in a JSON Document Fragment

```
"links": [
  {
    "href":"opaque-URI",
    "template":"opaque-template-URI (optional)",
    "rel":"rel-name",
    "title":"human-readable-title (optional)",
    "type":"media-type (optional)",
    "resourceType":"resource-type",
    "capabilities":"operation"
  },
  ...repeat as needed...
]
```

Multiword field, element, and attribute names are formatted in camel case, unless the representation is attempting to conform to a specification not under the service author's direct control. Acronyms are treated as normal words with their case adjusted accordingly (for example, `fooXml` or `xmlFoo`) as shown in the following examples:

Example: XML Naming Convention

```
<myElement>text</myElement>
```

Example: JSON Naming Convention

```
{"myElement": "text"}
```

This section includes the following topics that describe the different attributes of the hypermedia link:

- Resource Type

- Relationship

- Capabilities

- Media Type

- Templates

C.5.2.1 Resource Type

The `resourceType` link attribute indicates the type of resource to which the link points. Clients should use the `resourceType` to determine the expected response bodies for `GET` and `POST` and allowable request bodies for `POST` and `PUT`.

For more information, see Navigating Hypermedia Using HTTP.

C.5.2.2 Relationship

The `rel` link attribute indicates the relationship of the linked object to the current object (that is, the object that contains the list of links). The value of this attribute is a space-separated list of the following currently supported values:

- `self` - The linked object is the current object

- `related` - The linked object is related to the current object

- `via` - The linked object is the source of the information for the current object

- `alternate` - The linked object is a substitute for the current object (typically, the same object in another format, such as an HTML page that displays the current object)

- `urn:oracle:webcenter:parent` - The linked object is the parent of the current object. That is, the linked object owns the current object

---

Note:

Some REST APIs for some WebCenter features may contain additional `rel` link attributes. See the REST API documentation for each specific feature for more information.

C.5.2.3 Capabilities

The `capabilities` link attribute indicates which methods are supported by the linked resource.

Links are returned only if a client is allowed to access that resource. User authorization can affect the capabilities a client has with the links returned in a response representation. In general, services only return the capabilities that the current authorized user has permission to execute and that the resource supports.

If there is no link, then the client cannot access the resource. If a link has no capabilities, then it is not returned to the client, meaning that the client does not have permission to do anything with that link (even read it).

Capability-based expression of hypermedia links communicates the range of operations that the client can expect to succeed, which allows the client to dynamically configure any associated UI to provide the best overall user experience.

The value of this attribute is a space-separated list of the following values:

- `urn:oracle:webcenter:create` - This maps to the HTTP verb `POST`

- `urn:oracle:webcenter:read` - This maps to the HTTP verb `GET`

- `urn:oracle:webcenter:update` - This maps to the HTTP verb `PUT`

- `urn:oracle:webcenter:delete` - This maps to the HTTP verb `DELETE`

Note:

The top-level `resourceIndex` links only returns the `read` capability, even if the user is authorized with additional capabilities.

Note:

Querying a resource for the allowed HTTP verbs using `OPTIONS` returns the verbs that the resource can support in general, and does not take a user's access into account. The capabilities attribute in a link describes exactly what the current user can do with the current resource. `OPTIONS` may return more HTTP verbs than the current user is allowed.

C.5.2.4 Media Type

The `type` link attribute indicates the media types supported by the linked object.

All REST services, except for CMIS, support both XML (`application/xml`) and JSON (`application/json`) media type. CMIS currently supports only ATOM. For more information about the CMIS REST API, see Content Management REST API.

C.5.2.5 Templates

The `template` link attribute indicates that the client can use a URI template, instead of the `href` URI, to provide parameterized values for the linked object. Links must include at least an `href` or a `template` URI, but can include both.

Some hypermedia links support request query parameters that allow the client to configure the link in different ways. Rather than force the client to know the URI format and manually build the URI, URI templates are used. These templates allow client code to easily insert data into a URI without having to understand exactly how the URI works. This maintains the opacity of hypermedia URIs and protects the client from changes to the URI format.

The example below shows a URI template including several request query parameters.

Example: URI Template

```
http://host:port/.../lists?
startIndex={startIndex}&itemsPerPage={itemsPerPage}&q={searchTerms}&projection={proje
ction]
```

WebCenter Portal REST APIs use a simple slot replacement syntax that follows many industry URI template schemes.

For example, using the template above, to see 10 list items (default) on the first page, the client would provide a value of 1 for the `startIndex` parameter and a value of 10 for the `itemsPerPage` parameter, as shown below.

Example: URI Template with Parameter Values

```
http://host:port.../lists?startIndex=1&itemsPerPage=10
```

> **Note:**
>
> All unused parameters must be removed from a template before it can be used. Clients may not submit unprocessed templates to the service that produced it; doing so results in undefined behavior, generally returning a status code of 500.
>
> Clients must process templates into valid URI form before submitting to the server. Clients must replace slots with appropriate values, taking care to properly URI encode any value replacing the slot token. If a client does not have a suitable value for one or more of the slots in the template, then it must replace the slot token with an empty string.

You must URL-encode special or reserved character in parameter values. For example, to search lists for a person named Günter, you must URL-encode the ü as shown below.

Example: Encoding Special Characters in URI Templates

```
http://host:port.../lists?q=G%FCnter
```

C.5.2.5.1 Common Request Query Parameters

Many resources support a common set of request query parameters. For example, when retrieving a collection of entities, it is common to change the shape of the results set by limiting the quantity or details of the results. The REST framework uses the following request parameters to scope results and provide security:

- `startIndex` - Specifies the index of the first matching result that should be included in the result set (0-*n* ... zero based). This is used for pagination.

- `itemsPerPage` - Specifies the maximum number of results to return in the response (1-*n*). This is used for pagination.

- `q` - Specifies implementation-specific searching. Searches may be specified using the following format (square brackets [] denote optional values):

  ```
  [[field1:[operand]][:]value1[;field2:operand:value2]]
  ```

  For example:

  ```
  &q=login:equals:monty
  &q=title:contains:issues
  &q=creator:equals:monty;description:contains:Urgent
  ```

  While each resource uses the same format for the `q` parameter, the way search is implemented is different depending on the resource being searched. For more information about how each resource implements search, see the topic for the specific service.

- `projection` - Reserved for implementation-specific projection of model representations, such as variable recursion depth, field or attribute filtering. Valid values are `summary` or `details`.

  For example, requesting a projection of `summary` for a collection of lists, returns only the title, description, and hypermedia links. Requesting a projection of `details` results in the server sending back a collection of lists that includes all the column metadata for each list. This may require additional processing time or database queries on the server.

The following example request results in the response entity containing a deeper object graph.

```
http://host:port/...lists&projection=details
```

- `data` - This parameter accepts a comma separated list of data sets and items. This parameter lets clients specify what data they would like to receive. For example, a mobile device application might use this parameter to limit the amount of XML data returned. If both the `projection` and `data` query string parameters are present, the `data` parameter will be used to determine which data to return. If you specify the constant `'data'` as the data parameter, all the standard information will be returned for the resource.

For information about how these parameters are supported by specific resources, see the topic for the appropriate service.

C.6 Understanding Items Hypermedia

A collection of `items` makes up the actual content of responses. This is at the same level as the `links` section described previously. Each item (including the top-level tag in each response) has one common attribute (`resourceType`) in addition to resource-specific content and format. For details beyond the `resourceType`, see the topic for the specific service.

C.7 Navigating Hypermedia Using HTTP

You can navigate REST service hypermedia in a similar way to that used to browse and interact with HTML or an ATOM feed. Interactions are performed on the resources identified by links using HTTP methods. The REST services return response codes and response bodies to the client, and the client uses the hypermedia in the response to drive further interactions.

Table C-2 describes the general pattern followed when constructing opaque resource URIs. The `resourceType` differentiates whether the HTTP method operates on a collection of resources or an individual resource.

Table C-2   HTTP Methods

HTTP Method	Response for a Collection of Resources	Response for an Individual Resource
GET	Returns resource collection container (200 HTTP response code)	Returns resource (200 HTTP response code)
PUT	Cannot update a collection of resources (405 HTTP response code)	Updates and returns resource (200 HTTP response code)
POST	Creates and returns a new resource within the collection. **Note:** Can return a 201 or 204 HTTP response code. The returned code depends on whether the newly created object is directly addressable or not. For instance, activities cannot be addressed individually, so they return a 204 no-content response code.	Cannot create a resource within an individual resource (405 HTTP response code)

Table C-2 (Cont.) HTTP Methods

HTTP Method	Response for a Collection of Resources	Response for an Individual Resource
DELETE	Cannot delete a collection of resources (405 HTTP response code)	Deletes resource (204 HTTP response code)

Collection resources generally support reading a collection (GET) and creating a subordinate to that collection (POST). Individual resources generally support reading a resource (GET), updating a resource (PUT), and deleting a resource (DELETE).

C.7.1 HTTP Response Status Codes

Table C-3 describes the potential response status codes.

Table C-3 HTTP Response Status Codes

HTTP Response Status Code	Description
200	OK. Upon successful completion of a GET or PUT. This is accompanied by a resource representation as a response body.
201	Created. Upon successful completion of a POST. This has a location header to the newly-created resource.
204	No content, or any request that does not return content. For example, creating an object that cannot be linked. Upon successful completion of a DELETE.
400	Bad Request. The URI was malformed or could not be processed; for example, the IDs were not formatted correctly, or the ID was supplied in the URI on a POST.
401	Unauthorized. Client may retry by submitting credentials. This may be accompanied with a fault response body to help diagnose the issue.
403	Forbidden. Client does not have permission to perform a particular action, such as creating or deleting a resource. Re-authenticating as the same user does not help. This may be accompanied with a fault response body to help diagnose the issue.
404	Not Found. Referencing a specific resource with an ID, but that resource does not exist.
405	Method Not Allowed. This includes a list of valid methods for the requested resource.
406	Not Acceptable. The Accept header media type(s) sent by the client are not supported for the requested operation. This may be accompanied with a fault response body to help diagnose the issue.

Table C-3   (Cont.) HTTP Response Status Codes

HTTP Response Status Code	Description
409	Conflict. Possibly the resource ID is in use, or an entity has been modified by another process during an update.This may be accompanied with a fault response body to help diagnose the issue.
422	Bad entity body, the data in the body, although syntactically correct, was not valid, or could not be processed; for example, invalid data when updating a row.
500	Internal server error. The server encountered an unexpected condition that prevented it from fulfilling the request.
501	Not Implemented. The server does not support the functionality required to fulfill the request. This is the appropriate response when the server does not recognize the request method and is not capable of supporting it for any resource.

C.8 Security Considerations for WebCenter Portal REST APIs

All of the WebCenter REST URIs reference protected resources (similar to protected web pages) and require authentication for access.

> **Note:**
>
> The one exception to the authenticated access rule is access to the CMIS resources. CMIS resources can be accessed anonymously through the CMIS URI entry point:
>
> `http://host/port/rest/api/cmis/repository`
>
> See also Security Considerations for CMIS REST APIs.

You can pass this authentication in with the request using basic authentication, or you can configure the client and the WebCenter REST service to use single sign-on. For more information about single sign-on, see Configuring SSL in *Administering Oracle WebCenter Portal*.

Basic authentication sends the user's password in plain text. If you use this type of authentication, you should consider securing the connection using SSL. For more information, see Configuring SSL in *Administering Oracle WebCenter Portal*.

To provide additional security, every URI, for both `href` and `template` attributes, includes a security token parameter. The security token is user-scoped. This means that it is based on and scoped to an authenticated user and can be bookmarked or cached across that user's sessions. These security tokens help prevent Cross-Site Request Forgery (CSRF) attacks.

For example:

```
<link
  template="opaque-template-uri/@me?startIndex={startIndex}&itemsPerPage={itemsPerPage}
    &token=generated-token" resourceType="urn:oracle:webcenter:messageBoard"
```

```
    href="opaque-uri/@me?token=generated-token" capabilities="urn:oracle:webcenter:read"
/>
```

> **Note:**
>
> The security token is not used for authentication or identity propagation.

WebCenter Portal REST APIs operate under the identity of the authenticated user. For example, the portal REST APIs only return information for, and allow changes to, portals to which the user has access.

C.9 Security Considerations for CMIS REST APIs

The CMIS REST APIs do not use the same authentication scheme as the other WebCenter Portal REST APIs. Whereas other WebCenter Portal REST APIs do not allow unauthenticated access and prompt the user for authentication before allowing access, the CMIS REST APIs do allow unauthenticated access.

If a document requires authentication information and does not receive that information (because it is being accessed by an unauthenticated user), a 404 error is returned. This does not necessarily mean that the document cannot be found, rather that the (unauthenticated) user does not have the appropriate permissions to access the document. For the request to succeed, it should include basic authentication headers to identify the current user.

CMIS stands for Content Management Interoperability Services, a standard REST interface for Enterprise Content Management Systems. For more information, see Content Management REST API.

C.10 Understanding Common Types

This section describes the common types that are shared by multiple WebCenter Portal REST APIs.

- Common Types

- Portable Contact Types

C.10.1 Common Types

Common types provide a consistent way to reference objects used in the WebCenter Portal REST APIs.

This section includes the following topics:

- personReference

- groupSpaceReference

C.10.1.1 personReference

This is a generic data type that represents a user in the system. It is used by several APIs, for example to identify the author of a message board or feedback message, or a user's manager or direct reports. It is made up of the following elements:

- `guid` - The GUID of the user

- `id` - The login ID of the user

- `displayName` - The display name of the user

The `personReference` also includes a link to the user's profile icon. You can control the size of the icon by providing values: `small`, `medium`, or `large`.

Depending on where the `personReference` type is included, it can also return links to the associated REST APIs of the generating response. For example, if the `personReference` type is included as the author in a message board response, it includes links to message board services.

C.10.1.2 groupSpaceReference

This is a generic data type that represents a portal. It is used by several APIs, for example in the activity stream, to identify a portal in which a particular activity occurred. It is made up on the following elements:

- `guid` - The GUID of the portal

- `name` - The name of the portal

- `displayName` - The display name of the portal

The `groupSpaceReference` also includes an html link, rest link, and icon link.

C.10.2 Portable Contact Types

Portable Contact types provide users with a standard way to access their address books and friends lists over the Web. Portable Contact types are used by the Profile component of the People Connections service.

> **Note:**
>
> These types are based on the Portal Contact Types in WebCenter. They may include additional data.

This section includes the following topics:

- name Portable Contact Type

- address Portable Contact Type

- organization Portable Contact Type

- value Portable Contact Type

C.10.2.1 name Portable Contact Type

This is a portable contact type that provides information about the user's name. It is made up of the following elements:

- `formatted` - The formatted version of the full name of the user, for example, Michael David Jones Ph.D.

- `familyName` - The family name, or last name, of the user, for example Jones

- `givenName` - The given name, or first name, of the user, for example Michael

- `honorificSuffix` - The honorific suffix of the user, for example, Esq. or Ph.D.

- `initials` - The first initials of the user, for example, M. D.

- `maidenName` - The maiden name of the user

Some of the elements may not be present depending on the user repository configuration and data.

C.10.2.2 address Portable Contact Type

This is a portable contact type that provides information about the user's address. It is made up of the following elements:

- `formatted` - The formatted version of the full address

- `type` - The type of the address, for example, Home, Work

- `streetAddress` - The street address

- `poBox` - The post office box number

- `locality` - The city or locality

- `region` - The state or region

- `postalCode` - The zip code or postal code

- `country` - The country

Some of the elements may not be present depending on the user repository configuration and data.

C.10.2.3 organization Portable Contact Type

This is a portable contact type that provides information about the user's organizational affiliation. It is made up of:

- `name` - The name of the organization

- `employeeNumber` - The employee number of the user

- `employeeType` - The employee type of the user.

- `department` - The department within the organization to which the user belongs

- `defaultGroup` - The default group to which the user belongs

- `title` - The job title of the user within the organization

- `description` - A textual description of the user's role within the organization

- `expertise` - The expertise of the user within the organization

- `startDate` - The date when the user joined the organization

Some of the elements may not be present depending on the user repository configuration and data.

C.10.2.4 value Portable Contact Type

This is a generic object that contains data for a wide variety of contact information. It is made up of the following elements:

- `primary` - A boolean value that identifies whether this is the primary piece of information of this type for this person. The primary element may not be present and is only relevant if there are multiple values for the same type of data.

- `value` - The value for this type

- `type` - The type of information. Valid types are:

  - `standard`: with valid values of `work`, `home`, `other`

  - `phoneNumber`: with valid values of `work`, `home`, `fax`, `pager`, `mobile`

  - `photos`: with a valid value of `thumbnail`

C.11 Managing Caches

Client-side developers need to know how to handle HTTP cache headers in both requests and responses. Individual resources that have a "last modified" date are also return entity tags. The entity tags can be used to make retrieval of a specific entity more efficient. To learn more about the use of entity tags in caching, refer to the Hypertext Transfer Protocol specification:

`http://www.w3.org/Protocols/`

C.12 Configuring a Proxy Server

This section explains how to set up a simple, response-rewriting reverse HTTP proxy on an Apache server. A proxy server is typically employed to avoid cross-domain request problems associated with making XMLHttpRequest (XHR) calls from a browser client. These calls are typically associated with the Ajax development technique for creating rich, interactive client-side interfaces. REST APIs are typically used within this kind of client-side development scenario.

> **Note:**
>
> This section illustrates a simple example of setting up a proxy server on Apache. For more detailed information, refer to the Apache Server documentation available at `http://httpd.apache.org/docs`. You can also use Oracle HTTP Server (OHS) for your proxy server. For more information, see *Administrator's Guide for Oracle HTTP Server*.

The basic steps for setting up a proxy server on Apache are:

1. Obtain access to an Apache server. Oracle recommends Apache 2.2.7 or a later version.

2. Make sure the server has the `mod_substitute` and `mod_proxy` modules installed. Note that Apache versions 2.2.7 and later include `mod_substitute` by default. It is also possible to use `mod_sed` or `mod_line_edit`, however these configurations are not supported by Oracle.

3. Open the `httpd.conf` or the virtual host configuration file, and add the following lines, substituting your server name/information where appropriate:

```
ProxyRequests          Off
LoadModule             substitute_module        modules/mod_substitute.so
SetOutputFilter        SUBSTITUTE
```

```
ProxyPass               /rest/api/             http://myhost:8888/rest/api/
ProxyPassReverse        /rest/api/             http://myhost:8888/rest/api/
Substitute              s|myhost|yourhost|n

ProxyPass               /pathname/rest/api/      http://myhost:8888/rest/api/
ProxyPassReverse        /pathname/rest/api/      http://myhost:8888/rest/api/
Substitute              s|myhost:8888/rest/api|yourhost/pathname/rest/api|n
```

> **Note:**
>
> Two servers are being proxied in this example scenario. Note that the following two calls are actually talking to these two different servers, but they appear to clients to be the same server host:
>
> ```
> http://myhost/rest/api/resourceIndex
> ```
>
> ```
> http://myhost/pathname/rest/api/resourceIndex
> ```

4. Restart the Apache server. For example, on Linux, you could do this:

```
sudo /etc/init.d/httpd restart
```

Note that on some configurations of Linux, proxying with Apache in this fashion requires you tell selinux to allow outbound connections from httpd. You can accomplish this by enabling the `httpd_can_network_connect flag` in selinux's GUI or through the command line.

> **Developer Tip:**
>
> Set the `UserDir` permissions in `httpd.conf` to allow users to drop these files in their own `public_html` directory. For example, you might hit `http://host/~yourname/sample.html` to access your sample application, and then have the sample application make XHR calls to `http://host/rest/api/resourceIndex`.

C.13 WebCenter Portal REST API Examples

This section includes some examples illustrating how to use the WebCenter Portal REST APIs. It includes the following topics:

- Navigating the Message Board Hypermedia
- Displaying Activity Stream Data
- Updating User Status

C.13.1 Navigating the Message Board Hypermedia

This section includes examples to illustrate how to navigate the REST service hypermedia. The examples show how to read messages on a message board, post messages to another user's message board, delete unwanted messages, and filter message board messages.

This section includes the following topics:

- [Accessing the Resource Index](#)
- [Reading Messages](#)
- [Creating a New Message](#)
- [Updating a Message](#)
- [Deleting a Message](#)
- [Filtering Messages](#)

C.13.1.1 Accessing the Resource Index

The first step is always to access the Resource Index as shown in the example below.

Example: Accessing the Resource Index

```
GET /resourceIndex
```

This request returns a list of the top-level URI entry points to the RESTful services, including the entry point for the message board:

Example: Response to Accessing the Resource Index

```
200 OK
Accept: application/json;charset=UTF-8

{
  "resourceType": "urn:oracle:webcenter:resourceindex",
  "links": [
    {
      "resourceType": "urn:oracle:webcenter:resourceindex",
      "capabilities": "urn:oracle:webcenter:read",
      "rel": "self",
      "href": "http://host:port/rest/api/resourceIndex"
    },
    {
      "resourceType": "urn:oracle:webcenter:messageBoard",
      "capabilities": "urn:oracle:webcenter:read",
      "href": "opaque-messageBoard-URI"
    },
  ...repeating for other services...
}
```

You can examine this list to find the URI that you require to access your message board. You should look for the link with a `resourceType` of `urn:oracle:webcenter:messageBoard`. The `href` for this link is the one that you require to access your message board.

For other resources `rel`, `type`, and `template` also help find the correct link.

C.13.1.2 Reading Messages

Once you have determined the correct URI for your message board, you can send a GET request to that URI to read your messages.

To read messages on a message board, you must be logged in.

Example: Retrieving Messages from Your Message Board (GET)

```
GET /opaque-messageBoard-URI
```

The response provides information about all the messages on your message board:

Example: Response to Retrieving Messages from Your Message Board

```
200 OK
Accept: application/json;charset=UTF-8

{
  "resourceType": "urn:oracle:webcenter:messageBoard",
  "links": [
    {
      "resourceType": "urn:oracle:webcenter:messageBoard",
      "capabilities": "urn:oracle:webcenter:read urn:oracle:webcenter:create",
      "rel": "self",
      "href": "opaque-messageBoard-URI"
    }
  ]
  "items": [
    {
      "resourceType": "urn:oracle:webcenter:messageBoard:message",
      "links": [
        {
          "resourceType": "urn:oracle:webcenter:messageBoard:message",
          "capabilities": "urn:oracle:webcenter:read urn:oracle:webcenter:delete",
          "rel": "self",
          "href": "opaque-message-URI"
        }
      ]
      "id": "89add57c-7a35-4d35-b24f-ea9259612eb8",
      "body": "What's up?  It's been a while.  Some of us are going to Conner
O'Neal's after work.  Want to go?",
      "created": "2009-09-10T11:18:46.696-0700",
      "author": {
        "id": "carl",
        "displayName": "carl",
        "guid": "649A27F09D5C11DEBFAA799CBD41D9B8",
        "links": [
          {
            "resourceType": "urn:oracle:webcenter:people:person",
            "capabilities": "urn:oracle:webcenter:read",
            "rel": "via",
            "href": "opaque-person-URI"
          },
          {
            "resourceType": "urn:oracle:webcenter:messageBoard",
            "capabilities": "urn:oracle:webcenter:read urn:oracle:webcenter:create",
            "href": "opaque-messageBoard-URI-for-Carl"
          },
          {
            "type": "text/html",
            "resourceType": "urn:oracle:webcenter:spaces:profile",
            "capabilities": "urn:oracle:webcenter:read",
            "rel": "alternate",
            "href": "opaque-profile-URI"
          }
        ]
      },
    }
  ],
  "startIndex": 0,
  "itemsPerPage": 1,
}
```

From the response you can see that you have `read` and `create` capabilities on your message board. So you can read its contents and post new messages.

In addition, the response also includes a collection of items (in this case the collection consists of just a single item). These items, with a `resourceType` of `urn:oracle:webcenter:messageBoard:message`, are the messages on your message board. The `capabilities` attribute for the message indicates that, for this particular message, you can read it or delete it from your message board.

For each message, the response provides the following information:

- `id` - the identifier of the message

- `body` - the text of the message

- `author` - the author of the message. The author element is also made up of several other elements:

  - `id` - the identifier, or user name, of the author of the message

  - `displayName` - the name of the author, formatted for display

  - `guid` - the globally unique identifier of the author

Within the `author` element there is also a collection of three links. The `resourceType` of these links are:

- `urn:oracle:webcenter:people:person` - enables you to view information about the author of the message

- `urn:oracle:webcenter:messageBoard` - enables you to read or create a message on the author's message board

- `urn:oracle:webcenter:spaces:profile` - enables you to read a `text/html` document of the author's profile

C.13.1.3 Creating a New Message

Now that you have read the message on your message board, you probably want to reply to Carl on his message board. To do this you should send a `POST` request to the URI for Carl's message board.

To find the correct URI, use the `href` from the `author` link with `resourceType` of `urn:oracle:webcenter:messageBoard`.

A `POST` request creates a subordinate resource of the resource to which you post it. In this case, we are posting to the `messageBoard`, so we should post its subordinate resource: `message`:

Example: Creating a Message on Another User's Message Board (POST)

```
POST opaque-messageBoard-URI-for-Carl
Accept: application/json;charset=UTF-8
Content-Type: application/json;charset=UTF-8


{
    "body": "sure; see you guys at 6."
}
```

The response shows that your message was successfully created on Carl's message board:

Example: Response to Creating a Message on Another User's Message Board

```
201 Created
Content-Type: application/json;charset=UTF-8

{
  "id": "36b8464f-afda-44b5-90ad-8ecedcb040a3",
  "body": "sure; see you guys at 6.",
  "created": "2009-09-10T12:21:09.785-0700",
  "resourceType": "urn:oracle:webcenter:messageBoard:message",
  "links": [
    {
      "resourceType": "urn:oracle:webcenter:messageBoard:message",
      "capabilities": "urn:oracle:webcenter:read urn:oracle:webcenter:update
urn:oracle:webcenter:delete",
      "rel": "self",
      "href": "opaque-message-URI"
    },
  "author": {
    "id": "mike",
    "displayName": "mike",
    "guid": "649657609D5C11DEBFAA799CBD41D9B8",
    "links": [
      {
        "resourceType": "urn:oracle:webcenter:people:person",
        "capabilities": "urn:oracle:webcenter:read",
        "rel": "self",
        "href": "opaque-person-URI"
      },
      {
        "resourceType": "urn:oracle:webcenter:messageBoard",
        "capabilities": "urn:oracle:webcenter:read urn:oracle:webcenter:create",
        "href": "opaque-messageBoard-URI"
      },
      {
        "type": "text/html",
        "resourceType": "urn:oracle:webcenter:spaces:profile",
        "capabilities": "urn:oracle:webcenter:read",
        "rel": "alternate",
        "href": "opaque:profile:URI"
      }
    ]
  }
  ]
}
```

C.13.1.4 Updating a Message

A PUT request is very similar to a POST request, except that it is performed on the resource being edited, instead of on the parent resource.

From the response to your earlier POST request, when you created your message on Carl's message board, you can see that you have read, update, and delete capabilities on the message. You can also see that the href provides the URI for your message. Something came up at work and you must stay a bit later. Using the URI for your message, you can now send a PUT request to update the message and let Carl know that you are going to be late.

Example: Updating a Message (PUT)

```
PUT opaque:message:URI
Accept: application/json;charset=UTF-8
```

```
Content-Type: application/json;charset=UTF-8

{
    "body": "working late; see you guys at 7."
}
```

The response is nearly identical to that of POST, except that the body contains your updated message:

Example: Updating a Message

```
200 OK
Content-Type: application/json;charset=UTF-8

{
  "id": "36b8464f-afda-44b5-90ad-8ecedcb040a3",
  "body": "working late; see you guys at 7.",

...deleted for brevity...

}
```

C.13.1.5 Deleting a Message

Performing a DELETE request on a resource deletes it, if you have the delete capability on the resource. The link to your message on Carl's message board supports delete.

You decide to delete the message that you left on Carl's message board:

Example: Deleting a Message (DELETE)

```
DELETE opaque:message:URI
```

The response is simply a status code of 204:

Example: Response to Deleting a Message

```
204 NO CONTENT
```

> **Note:**
>
> DELETE is idempotent, meaning that it can be sent multiple times with the same result. Therefore if you try to delete the same object twice, you still receive the same 204 response even though it has previously been deleted.

C.13.1.6 Filtering Messages

Messages can be filtered (using the HTTP verbs GET, POST, and PUT) based on visibility criteria. Messages posted on message boards falls into the following visibility categories:

- Public

- Private

- Hidden

- Public and hidden

- Private and hidden

C.13.1.6.1 Public vs Private Messages

The owner of a message board can mark any message as private. When a message is marked as private, that message is not visible to anyone other than the owner of the message. By default, all messages are public.

Messages can also be sent as private messages. If Mike, for example, is viewing Carl's message board, only public messages (including those sent or received as private) will be visible to user Mike.

C.13.1.6.2 Hidden vs Non-hidden Messages

The owner of a message board can mark any message as hidden. When a message is marked as hidden, that message will not be visible to the message board's owner, but will remain visible to anyone else viewing that person's message board.

The following examples show how to retrieve,

Example: Retrieving Filtered Messages (GET)

- me

```
all
  rest/api/messageBoards/person/@me
private
  rest/api/messageBoards/person/@me/private
public
  rest/api/messageBoards/person/@me/public
hidden and public
  rest/api/messageBoards/person/@me/hidden
private and hidden
  rest/api/messageBoards/person/@me/private_hidden
```

- person

```
rest/api/messageBoards/person/<GUID>
```

  Note that the GUID of the logged in user for whom to retrieve messages is required.

- space-guid

```
rest/api/messageBoards/space/<GUID>
```

  Note that visibility-based filtering is not available for `space-guid`.

Example: Using Filtering for New Messages (POST)

- me

```
all
  rest/api/messageBoards/person/@me
private
  {"body" : "<BODY_CONTENT>","visibilityType" : "private"}
public
  {"body" : "<BODY_CONTENT>","visibilityType" : "public"}
hidden and public
 {"body" : "<BODY_CONTENT>","visibilityType" : "hidden"}
private and hidden
  {"body" : "<BODY_CONTENT>","visibilityType" : "private_hidden"}
```

- person

  ```
  rest/api/messageBoards/person/<GUID>
  ```

  For a GUID other than that of "me":

  ```
  public
    {"body" : "<BODY_CONTENT>","visibilityType" : "public"}
  private
    {"body" : "<BODY_CONTENT>","visibilityType" : "private"}
  ```

- space-guid

  ```
  rest/api/messageBoards/space/<GUID>
  ```

  Note that visibility-based filtering is not available for `space-guid`.

Example: Using Filtering for Modified Messages (PUT)

- me

  ```
  all
    rest/api/messageBoards/person/@me/messages/<msg guid>
  private
    {"body" : "<BODY_CONTENT>","visibilityType" : "private"}
  public
    {"body" : "<BODY_CONTENT>","visibilityType" : "public"}
  hidden and public
   {"body" : "<BODY_CONTENT>","visibilityType" : "hidden"}
  private and hidden
    {"body" : "<BODY_CONTENT>","visibilityType" : "private_hidden"}
  ```

- person

  ```
  rest/api/messageBoards/person/<GUID>/messages/<msg guid>
  ```

  For a GUID other than that of "me":

  ```
  public
    {"body" : "<BODY_CONTENT>","visibilityType" : "public"}
  private
    {"body" : "<BODY_CONTENT>","visibilityType" : "private"}
  ```

- space-guid

  ```
  rest/api/messageBoards/space/<GUID>/messages/<msg guid>
  ```

  Note that visibility-based filtering is not available for `space-guid`.

C.13.2 Displaying Activity Stream Data

To properly display Activity Stream data, you must:

- Retrieve the Activity Stream URI entry point

- Retrieve the Activity Stream data

- Process the Activity Stream data for display

More On OTN

This sample is available on the Oracle WebCenter Portal Demonstrations and Samples page on the Oracle Technology Network (OTN) at:

http://www.oracle.com/technetwork/middleware/webcenter/ps3-samples-176806.html

In the following example:

- The getResourceURL method shows you how to retrieve the URI entry point for the Activity Stream service by retrieving the JSON data for the main Resource Index (/rest/api/resourceIndex) and locating the URI for the Activity Stream resource in that data.

- The formatMessage method processes the Activity Stream data into a displayable format. This involves locating an individual message and replacing any template parameters in the message with the name of the object or user that corresponds to that parameter. The parameters also include links to display the object or user. They may contain links to REST services for those objects or users, if available.

> **Note:**
>
> To get the Resource Index and Activity Stream JSON data, make Ajax requests using Javascript (and possibly a client-side scripting library like Dojo) and pass the resulting data into the appropriate methods.

Example: Displaying Activity Stream Data

```
/* Parse the resourceIndex to find the specified URL and
 * return it.
 *
 * @Param jsonData the JSON data retrieved from calling
 *        the /rest/api/resourceIndex URL.
 * @Param strResourceType the resource type of the URL
 *        you want to retrieve from the resourceIndex data.
 *        E.g., 'urn:oracle:webcenter:activities:stream'
 */
function getResourceURL(jsonData, strResourceType)
{
  // Using the HATEOAS model, we browse the returned links
  // looking for the one with the correct resource type.

  for (var i = 0; i < data.links.length; i++) {
    if (data.links[i].resourceType == strResourceType) {
      return data.links[i].href;
    }
  }
}

/* Parse the resourceIndex to find the activity stream URL and
 * then load it.
 *
 * @Param jsonData the JSON data retrieved from calling
 *        the /rest/api/resourceIndex URL.
 */
function getActivitiesURL(jsonData)
{
  // Parse the JSON data to get the activities URI entry point.

  var strHref = getResourceURL(jsonData,
```

```
                              'urn:oracle:webcenter:activities:stream');

        // INSERT CODE HERE: Implement getting the JSON data from the
        // strHref URL with the Accept header set to "application/json",
        // and use the formatMessage(index, jsonData) function to get
        // the displayable activity message for each activity.

     }

     /* Replace activity message parameters.
      *
      * @Param index the index of the activity to process
      * @Param jsonData the JSON data retrieved from calling
      *          the /rest/api/resourceIndex URL.
      */
     function formatMessage(index, jsonData)
     {
       var activity = jsonData.items[index];
       var strMessage = activity.message;

       // Look for activity parameters and replace them in the message.

       if (activity.templateParams && activity.templateParams.items) {
         for (var i = 0; i < activity.templateParams.items.length; i++) {
           var param = activity.templateParams.items[i];

           // Each parameter also has a set of links which at least
           // includes an HTML link and possibly a REST API link.

           strMessage = strMessage.replace(param.key, param.displayName);
         }
       }
       if (activity.detail) {
         strMessage = strMessage + "<br><font size='1'>" +
           activity.detail + "</font>";
       }
       return strMessage;
     }
```

C.13.3 Updating User Status

The following example shows how to use REST APIs to update a user's WebCenter Portal profile status.

Note:

You must host the sample on a web server (for example, Apache or Oracle HTTP Server) or an application server. To avoid cross site scripting errors, you should proxy URL access to the REST service. On Apache or OHS, the conf commands would look like (change *myspaceshost* and *port* accordingly):

```
ProxyPass /webcenter/ http://myspaceshost:port/webcenter/
ProxyPassReverse /webcenter/ http://myspaceshost:port/webcenter/
ProxyPass /rest/ http://myspaceshost:port/rest/
ProxyPassReverse /rest/ http://myspaceshost:port/rest/
ProxyPreserveHost on
```

This sample is available on the Oracle WebCenter Portal Demonstrations and Samples page on the Oracle Technology Network (OTN) at:

http://www.oracle.com/technetwork/middleware/webcenter/ps3-samples-176806.html

Updating user status involves making a sequence of asynchronous AJAX calls to get the URL for the status object:

```
urn:oracle:webcenter:resourceindex
   urn:oracle:webcenter:people
      urn:oracle:webcenter:people:person:status
```

The HTML page for updating user status includes an input field where users can enter the new status message (statusMessage). Clicking the **Update Status** button (or pressing Enter) calls the updateStatus method to make the initial call to the Resource Index.

Example: HTML Body

```
<body>
<div>New status message: <input id="statusMessage" type="text"
   onkeyup="{if (event.keyCode==13) updateStatus();}" maxlength="250"
   size="60" /></div>
<div>
<button id="button1" onclick="updateStatus();">Update Status</button>
</div>
<div id="statusResults"></div>
</body>
```

The following example shows the HTML page.

Figure C-1   New Status Message HTML Page

The code in the following example retrieves the Resource Index (resourceIndexURL variable set to /rest/api/resourceIndex). The Resource Index is returned as an AJAX request object (resourceIndexRequest).

Example: Retrieving the Resource Index

```
function updateStatus() {
   // Set the UI to busy state. This is cleared in the success and error callbacks
   setUIBusy("Updating status...");
   //get the Resource Index
   resourceIndexRequest.get(
      resourceIndexURL,
      resourceIndexCallback,
      clearUIBusy);
}
```

The getResourceURL method shown in the following example traverses the links in the data returned by a REST call to find a specified string (URN) that identifies the required resource type.

Example: Traversing the Links

```
function getResourceURL(data, strResourceType) {
   for (var i = 0; i < data.links.length; i++) {
      if (data.links[i].resourceType == strResourceType) {
         return data.links[i].href;
      }
   }
   return null;
}
```

The following example uses `getResourceURL` to parse the Resource Index to find the user profile URL. The data is returned as an AJAX request object (`profileRequest`).

Example: Retrieving the User Profile

```
function resourceIndexCallback(data) {
   //get my user profile
   profileRequest.get(
      getResourceURL(data, 'urn:oracle:webcenter:people'),
      profileCallback,
      clearUIBusy);
}
```

The following example takes the new status message provided by the user in the HTML page (`statusMessage`) and uses `request.put` to update the status object (retrieved by again calling `getResourceURL`).

Example: Retrieving the Status Object

```
function profileCallback(data) {
   profile = data;
   // get the URL for the status object
   var url = getResourceURL(data, 'urn:oracle:webcenter:people:person:status');
   // get the new status and escape double quotes
   var newStatus = document.getElementById
      ('statusMessage').value.replace(/\"/g. "\\\"");
   //send a JSON string representing the new status object
   var statusMessage = '{"note": "' + newStatus + '"}';
   statusRequest.put(
      getResourceURL(data, 'urn:oracle:webcenter:people:person:status'),
         statusMessage, renderStatusPutResults, clearUIBusy);
}
```

The `renderStatusPutResults` method, shown in the example below, renders the new status object.

Example: Rendering the New Status Object

```
function renderStatusPutResults(data) {
   var name = profile.displayName;
   // get the URL for my profile HTML page in WebCenter Portal
   var profileURL = getResourceURL(profile,
      'urn:oracle:webcenter:spaces:profile');
   var html = 'Status for <a href="' + profileURL + '" target="_blank">'
      + name + '</a> is: ' + data.note;
   // clear the UI busy state and print the new status message
   clearUIBusy(html);
}
```

The following example shows the code for disabling and reenabling the UI.

Example: Disabling the UI

```
function setUIBusy(message) {
   document.getElementById('statusResults').innerHTML = message;
   document.getElementById('button1').disabled = true;
   document.getElementById('statusMessage').disabled = true;
}

function clearUIBusy(message) {
   document.getElementById('statusResults').innerHTML = message;
   document.getElementById('button1').disabled = false;
   document.getElementById('statusMessage').disabled = false;
}
```

The example below shows the library that assists in AJAX calls. Most of this is not WebCenter specific and works for any XHR requests to a service that returns JSON.

The library includes a reusable XHR object. This object supports HTTP GET, POST, PUT, and DELETE. All functions are asynchronous and take a URL and two callback functions, one for success, and one for failure. Success calls are made with a JavaScript object containing the return data. Failure calls are made with an error string. POST and PUT also take a data argument which must be a JSON string.

Example: AJAX Library

```
function AjaxRequest() {
   // constructor to create an object oriented AJAX request
   var xhr = null;

   // get XHR object. Should work for IE 6+, Safari, and Mozilla based browsers
   if (window.XMLHttpRequest) {
      xhr = new window.XMLHTTPRequest;
   } else {
      try {
         xhr = new ActiveXObject('MSXML2.XMLHTTP.3.0');
      } catch (ex) {
         xhr = null;
      }
   }
   if (xhr == null) {
      alert("Your browser does not support AJAX");
   }

   this.get = function(url, callback, errorCallback) {
      xhr.open('GET', url, true);
      // the REST APIs return XML by default. Please return JSON
      xhr.setRequestHeader('Accept', 'application/json; charset=utf-8');
      sendRequest(null, callback, errorCallback);
   };

   this.post = function(url, data, callback, errorCallback) {
      xhr.open('POST', url, true);
      // set headers to send and receive JSON
      xhr.setRequestHeader('Accept', 'application/json; charset=utf-8');
      xhr.setRequestHeader('Content-Type', 'application/json; charset=utf-8');
      sendRequest(data, callback, errorCallback);
   };

   this.put = function(url, data, callback, errorCallback) {
      xhr.open('PUT', url, true);
      // set headesr to send and receive JSON
      xhr.setRequestHeader('Accept', 'application/json; charset=utf-8');
      xhr.setRequestHeader('Content-Type', 'application/json; charset=utf-8');
```

```
            sendRequest(data, callback, errorCallback);
        };

        this.deleteResource = function(url, callback, errorCallback) {
            xhr.open('DELETE', url, true);
            sendRequest(null, callback, errorCallback);
        };

        // set the callbacks and send the request with data, if any
        function sendRequest(data, callback, errorCallback) {
            xhr.onreadystatechange = function () {
                processResponse(xhr, callback, errorCallback);
            };
            xhr.send(data);
        }

        function processResponse(xhr, callback, errorCallback) {
            var data = null;
            // can get called here many times. 4 means really done
            if (xhr.readyState == 4) {
                // let's call any HTTP codes in the 200s success
                if (xhr.status >= 200 && xhr.status <300) {
                    // convert response text into a JSON object. This is insecure.
                    // data should not be blindly evaluated from the server return.
                    // consider using json2.js http://www.JSON.org/js.html
                    data = eval('(' + xhr.responseText + ')');
                    callback(data);
                } else {
                    // we got an error. Format an error string
                    data = 'Error: ' + xhr.status + ' ' + xhr.statusText + '<br />'
                        + xhr.responseText;
                    errorCallback(data);
                }
            }
        }
    }
```

C.14 Using the People Connections REST APIs

This section includes the following topics:

- Activity Stream REST API

- Connections and Profile REST API

- Feedback REST API

- **Message Board REST API**

- Creating an Invitation

C.14.1 Activity Stream REST API

Use the Activity Stream REST API to browse user application activities in an activity stream. This section provides information about the REST API methods you can use to perform this action.

This section includes the following topics:

- Activity Stream Entry Point

- Activity Stream Resource Type Taxonomy

- Activity Stream Security Considerations

- Activity Stream Resource Types

C.14.1.1 Activity Stream Entry Point

Each REST service has a link element within the Resource Index that provides the entry point for that service. For People Connections, each feature has its own link element. For example, to find the entry point for the People Connections' Activity Stream feature, find the link elements with a `resourceType` of:

`urn:oracle:webcenter:activities:stream`

The corresponding `href` or `template` element provides the URI entry point, which retrieves application activities for the current user from the Activity Stream. The client sends HTTP requests to this entry point to work with the People Connections' Activity Stream feature.

> **See Also:**
>
> For more information about the Resource Index, see Using the Resource Index.
>
> For more information about resource types, see Resource Type.

C.14.1.2 Activity Stream Resource Type Taxonomy

When the client has identified the entry point, it can then navigate through the resource type taxonomy to perform the required operations. For more information about the individual resource types, see the appropriate section in Activity Stream Resource Types.

The resource type taxonomy for the People Connections' Activity Stream feature is:

```
urn:oracle:webcenter:activities:stream
urn:oracle:webcenter:activities:activity
```

C.14.1.3 Activity Stream Security Considerations

You must be logged into the REST service to access any of the People Connections REST APIs. After that, the underlying service handles permission checking and the like.

> **See Also:**
>
> For general security considerations, see Security Considerations for WebCenter Portal REST APIs.

C.14.1.4 Activity Stream Resource Types

This section provides you with all the information you need to know about each resource type. It includes the following topics:

- urn:oracle:webcenter:activities:stream

- Navigation Paths to stream

C.14.1.4.1 urn:oracle:webcenter:activities:stream

The `stream` response contains URIs for use in retrieving activities from the Activity Stream.

You can retrieve the activities from a user's stream or activities from a user's connections' streams. To have even greater control over which activities to retrieve, use the activity stream query filter. With the query filter, you can:

- Specify the user to query

- Include the user's connections' activities in the results

- Include activities from portals, including the Home portal, in the results

- Restrict the results to activities from specific services

The options available to you depend on the path you take to get to the `stream` resource and the `rel` of the link that you use. For example, the activity stream query filter is available only from links with a `rel` attribute of `urn:oracle:webcenter:activities:stream`. If you access the activity stream query filter from the `person` resource, the `personGuid` parameter is prefilled.

Table C-4 shows the activities returned depending on the `rel` element of the link.

Table C-4   Activities Returned by stream

rel	Returns
`urn:oracle:webcenter:activities:stream:person`	Activities from the user's stream (*GUID*/`@self`)[1]
`urn:oracle:webcenter:activities:stream:connections`	Activities from the user's connections' streams (*GUID*/`@connections`)
`urn:oracle:webcenter:activities:stream`	Activities determined by the activity stream query filter
`urn:oracle:webcenter:activities:stream:space`	Activities from the portal activity stream
`urn:oracle:webcenter:activities:stream:list`	Activities from the portal activity list

[1] *GUID* can be any valid user GUID or @me.

C.14.1.4.2 Navigation Paths to stream

This section shows how the client can navigate through the hypermedia to access this resource:

```
resourceIndex
    stream (rel="urn:oracle:webcenter:activities:stream:person" or
                "urn:oracle:webcenter:activities:stream")

resourceIndex
    person
        stream (rel="urn:oracle:webcenter:activities:stream:person" or
                    "urn:oracle:webcenter:activities:stream:connections" or
                    "urn:oracle:webcenter:activities:stream")

resourceIndex
    person
```

```
      list
          stream

resourceIndex
   space
       stream

stream
   activity
       param
          stream (rel="urn:oracle:webcenter:activities:stream:space")

personReference
   stream
```

C.14.1.4.3 Supported Methods for stream

The following method is supported by the `stream` resource:

- `GET`

  - **request—Parameters:**

    ◆ `startIndex, itemsPerPage`

See Also:

For information about REST API parameters, such as `startIndex` and `itemsPerPage`, see Common Request Query Parameters.

The following additional parameters are available for the query filter URI:

◆ `fromDate` – Specifies activities start date (`yyyy-mm-dd`).

◆ `toDate` – Specifies activities end date, (`yyyy-mm-dd`).

◆ `data` – Returns specified data only (for more information, see Common Request Query Parameters). For the stream resource, if you specify the constant `'data'` as the data parameter, all of the basic information about the resource is returned except "comments" and "likes" summaries. If you want to return comments or likes, specify the `data` parameter value `'commentsSummary'` or `'likesSummary'`.

Note:

You can specify multiple data values as a comma separated list. For example, `data=data, commentsSummary`.

◆ `personGuid` – (Required) Retrieves activities from the stream for the specified user. **Valid values:** any valid user GUID or `@me`.

◆ `serviceIds` – Retrieves activities only for the specified services. **Valid values:** An asterisk (*) returns all services. If null or empty, uses the user preference settings for service filters (from the settings link in the top bar).

◆ `personal` – Includes the specified user's activities in the Home portal. **Valid values:** `true` or `false`. **Default value:** `false`.

◆ `connections` – Includes activities from the streams of the specified user's connections. **Valid values:** `true` or `false`. **Default value:** `false`.

◆ `groupSpaces` – Includes activities from all portals of which the specified user is a member. **Valid values:** `true` or `false`. **Default value:** `false`.

◆ `connectionListIds` – A comma separated list of connection list names that specifies the connection lists used to show activities.

◆ `groupSpaceGuids` – A comma separated list of portal GUIDs used to show activities.

◆ `userGroupSpaceActivities` – Specifies whether or not to show the user's activities in their portal. **Valid values:** `true` or `false`. **Default value:** `false`.

◆ `followedObjects` – Specifies whether or not to show all activities for followed objects that both the current user and the specified user follow. **Valid values:** `true` or `false`. **Default value:** `false`.

◆ `followedObjectsUserActivities` – Specifies whether or not to show the specified user's activities for followed objects that both the current user and the specified user follow. **Valid values:** `true` or `false`. **Default value:** `false`.

◆ `advancedQuery` – Specifies filters against streamed activities. Create filters for user names, service IDs, and object details, such as a document's display name.

Note:

You must plug actual values into the `advancedQuery` parameter. You cannot pass EL expressions directly into the parameter. Typically, the REST API client handles an EL transformation manually and inserts the actual object data value into the REST URL. See the example below.

For example, the following URI returns activities from the current user's activity stream, for all services that the user has configured in their personal preference settings for service filters. It returns activities from the user's Home portal and from the streams of the user's connections:

```
http://host:port/rest/api/activities?
personal=true&connections=true&personGuid=@me&token=utoken
```

The following URI returns only the user's personal profile and connections activities:

```
http://host:port/rest/api/activities?serviceIds=oracle.webcenter.peopleconn
ections.profile,oracle.webcenter.peopleconnections.connections&personal=
true&personGuid=@me&token=utoken
```

This next example illustrates how to use the `advancedQuery` parameter. As explained previously, you cannot pass an EL expression to `advancedQuery`. The REST API client must first obtain the actual data object value, and that value can then be passed to `advancedQuery`. For example, to filter activities for a particular portal, you can pass the GUID of the portal's scope to `advancedQuery`:

```
http://host:port/rest/api/activities?personGuid=@me&advancedQuery=AE.SCOPE_ID
%20%3D%20\%27s8bba98ff_4cbb_40b8_beee_296c916a23ed\%27&ttoken=token
```

where `s8bba98ff_4cbb_40b8_beee_296c916a23ed` is the GUID of the portal. Note that the query string must be encoded with the appropriate escape codes.

– **response—Body:** 0 or more activities

> **Note:**
>
> Because the `stream` resource includes activity items, the response may also return resource links for the objects referenced in the activity.

C.14.1.4.4 Resource Types Linked to From stream

Table C-5 lists the resource types that the client can link to from the `stream` resource.

Table C-5   Related Resource Types for stream

rel	resourceType
self urn:oracle:webcenter:activities:stream:person	urn:oracle:webcenter:activities:stream
	urn:oracle:webcenter:activities:activity

C.14.1.4.5 urn:oracle:webcenter:activities:activity

The `activity` response contains the data for activities and URIs for use in retrieving all the data you need from an activity that is included in an Activity Stream.

C.14.1.4.6 Navigation Paths to activity

This section shows how the client can navigate through the hypermedia to access the `activity` resource:

```
resourceIndex
   stream
      activity

resourceIndex
   person
      stream
         activity

resourceIndex
   space
      stream
         activity
```

C.14.1.4.7 Supported Methods for activity

No methods are supported for `activity`. Activities are currently available only from the `stream` resource type.

C.14.1.4.8 Resource Types Linked to from activity

Table C-6 lists the resource types that the client can link to from this resource.

Table C-6   Related Resource Types for urn:oracle:webcenter:activities:activity

rel	resourceType
self	`urn:oracle:webcenter:activities:activity`
icon	urn:oracle:webcenter:activities:activity:icon

C.14.1.4.9 Read-only Elements for activity

Table C-7 lists the read-only elements for the `activity` resource.

> **Note:**
>
> The activity will also return a link to an activity icon in the links section of the response, if an icon is available. See urn:oracle:webcenter:activities:activity:icon.

Table C-7   Read-Only Elements for activity

Element	Type	Description
`activityType`	String	Activity type
		Unique within the service. For example, Discussions can return the `activityType createForum`.
`commentsCount`	String	If you specify `commentsSummary` in the data parameter, then the `commentsCount` parameter will be returned.
`createdDate`	Date (String)[1]	Date the activity was created.
`description`	String	The description of the activity.
`detail`	String	Detail information for the activity, if available.
		For example, this might return the contents of a message, the file name of a document, and the like.
		Similar to `detailURL`. Both, either, or neither can be used. The detailURL will be available when creating wiki and blog in a portal.
`detailURL`	String	detailURL is available when creating wikis and blogs in a portal.In a Web-based WebCenter Portal, the user can click this URL and open a wiki or blog page in a portal.
		Similar to `detail`. Both, either, or neither can be used.

Table C-7   (Cont.) Read-Only Elements for activity

Element	Type	Description
displayDescription	String	A pre-formatted, internationalized description.
displayMessage	String	A pre-formatted, internationalized message (does not include template information).
groupSpace	groupSpaceReference	Information about the portal in which the activity was performed **Note:** This reference is not present for activities that did not happen in a portal (for example, activities that happened in the Home portal). Also, because creation of a portal happens in a Home portal, that activity does not have this element either.
id	String	Unique ID of the message
isSummary	true or false	Indicates whether or not this activity is a summary of other activities.
likesCount	String	If you specify likesSummary in the data parameter, then the likesCount parameter will be returned.
message	String	Localized string for this activity This may contain replacement strings within curly braces ({}).
permission	PRIVATE SHARED PUBLIC	Permission level of this activity
scope	String	Scope of the activity This might return a portal, like the Home portal. For example, for a portal, it returns a string similar to the following: s8bba98ff_4cbb_40b8_beee_296c916a23ed
serviceId	String	Unique ID of the service that created the activity

Table C-7   (Cont.) Read-Only Elements for activity

Element	Type	Description

sharedLink_url	String	Can be used to render a navigation link by the REST client as shown in the example below:
		
		`<param`
		`resourceType="urn:oracle:webcenter:activities:parameter">`
		`<links>`
		`<link href="http://www.google.com" />`
		`</links>`
		`<displayName>http://www.google.com</displayName>`
		`<key>_sharedLink_url</key>`
		`<type>custom</type>`
		`</param> "`
		
		Note that this parameter cannot be used in a message template (for example: {actor[0]} has created the portal {object[0]}).

templateParams	urn:oracle:webcenter:activities:parameter	A list of param elements that capture data related to an activity. A key provided with each templateParam <param> element allows you to plug one or more data items into a parameterized activity message. See Understanding the templateParams Element.
custom	Param	The custom parameter includes a displayName, key, and type, and may or may not have a URL.

[1] Data types, such as DATE and BOOLEAN, are stored in the API as STRING. The DATE data type returns a Java standard date format, for example, 2009-08-21T14:43:11.0013-0700, with 0700 representing the time zone.

C.14.1.4.10 Understanding the templateParams Element

Suppose you want to display the most recent messages for a user named Carl. You want to display information like this: "Carl created the portal Customer Feedback". The templateParams element helps you solve this problem.

The templateParams element is returned in objects of type oracle:webcenter:activities:activity. This element captures a lot of data related to an activity. For example, if a user creates a new portal, the templateParams element for that activity captures information about the user and about the portal. Keys are provided that allow you to perform string substitutions in a parameterized message string related to the activity.

For example, if the user creates a portal, the returned activity object contains a <message> item that is parameterized like this:

```
<message>{actor[0]} has created the portal {object[0]}</message>
```

By parsing the `templateParams` element for the activity, you can find the available keys that allow you to perform string substitutions as well as appropriate data to display, like the name of the user and the activity.

The `templateParams` element also provides links to `comments`, `likes`, `commentsSummary`, and `likesSummary` objects, if they are requested using the `data` parameter. These links let you query for all the comments or likes for an object or post a new comment or like. The summary links returns the comments or likes count and several recent comments or likes. See also Understanding Comments and Likes.

The links returned with the `templateParams` element vary depending on what kind of object is returned, like a user, document, portal, or custom object. For more information, see urn:oracle:webcenter:activities:parameter.

> **Note:**
>
> The `templateParams` element can sometimes contain elements that are not directly referenced in the message.

If a `rel` link is marked "via," this means it is a link to the underlying REST object–for instance, the document not the parameter. If a `rel` link is marked "alternate" and type "text/html," it is a link to the HTML page for that object. Portal objects include an activity stream link for portals. Users have icon and activity stream links. Other objects can have an `alt` link to the tagged item, as well as the related tagged items list, if tagging is enabled for that object. If a regular object supports comments or likes, it can include the comments/commentsSummary and likes/likesSummary, as explained previously.

C.14.1.4.11 Understanding Comments and Likes

Hyperlinks to `comments`, `commentsSummary`, `likes` and `likesSummary` are included by default (you can expand the `commentsSummary` and `likesSummary` by specifying `data=data,commentsSummary,likesSummary`). Comments and likes can be associated with the object referenced by the activity or the activity itself. For example, if you edit a document, then the comments will be associated with the document. If you add comments on an edit document activity on the activity stream page, then comments will be associated with the edit document activity.

The links associated with comments and likes are:

- `comments` – The `comments` link lets you query for all the comments for an object. This link also lets you POST a new comment. For a `comments` POST, the `text` field is required. For example, for body data you would use:

  JSON:

  ```
  {
    "text" : "REST Comment"
  }
  ```

  XML:

  ```
  <text>REST Comment</text>
  ```

- `commentsSummary` – The `commentsSummary` link returns the comments count and several recent comments.

- `likes` – The `likes` link lets you query for all the likes for an object. This link also lets you POST a new like. There are no required fields for a `likes` POST.

- `likesSummary` – The `likesSummary` link returns the likes count and the current user's like (if available).

Note:

`Like` objects themselves only support DELETE, not GET. The `likesCount` and `commentsCount` items return the number of likes or comments for the current object.

The following URLs show examples of `GET` requests for `comments`, `commentSummary`, `likes` and `likesSummary` for a document object:

```
"http://example.com:8892/rest/api/activities/services/oracle.webcenter.doclib/
objectTypes/document/objects/(<document_object_id>)/comments?
startIndex=0&amp;itemsPerPage=10&amp;utoken=FCvY3qQSBh0eAByLdxugV-lUgFO3_w**"
```

```
"http://example.com:8892/rest/api/activities/services/oracle.webcenter.doclib/
objectTypes/document/objects/(<document_object_id>)/commentsSummary?
utoken=FCvY3qQSBh0eAByLdxugV-lUgFO3_w**"
```

```
"http://example.com:8892/rest/api/activities/services/oracle.webcenter.doclib/
objectTypes/document/objects/(<document_object_id>)/likes?
startIndex=0&amp;itemsPerPage=10&amp;utoken=FCvY3qQSBh0eAByLdxugV-lUgFO3_w**"
```

```
"http://example.com:8892/rest/api/activities/services/oracle.webcenter.doclib/
objectTypes/document/objects/(<document_object_id>)/likesSummary?
utoken=FCvY3qQSBh0eAByLdxugV-lUgFO3_w**"
```

Where <document_object_id> is the document object ID.

The following URLs show examples of `GET` requests for `comments`, `commentSummary`, `likes` and `likesSummary` for a create document activity:

```
"http://example.com:8892/rest/api/activities/ffa9a9f0-d02f-4e30-8c58-8a41b7a6e8a3/
comments?startIndex=0&amp;itemsPerPage=10&amp;utoken=FCvY3qQSBh0eAByLdxugV-
lUgFO3_w**"
```

```
"http://example.com:8892/rest/api/activities/ffa9a9f0-d02f-4e30-8c58-8a41b7a6e8a3/
commentsSummary?utoken=FCvY3qQSBh0eAByLdxugV-lUgFO3_w**"
```

```
"http://example.com:8892/rest/api/activities/ffa9a9f0-d02f-4e30-8c58-8a41b7a6e8a3/
likes?startIndex=0&amp;itemsPerPage=10&amp;utoken=FCvY3qQSBh0eAByLdxugV-lUgFO3_w**"
                "http://example.com:8892/rest/api/activities/ffa9a9f0-
d02f-4e30-8c58-8a41b7a6e8a3/likesSummary?utoken=FCvY3qQSBh0eAByLdxugV-lUgFO3_w**"
```

C.14.1.4.12 urn:oracle:webcenter:activities:parameter

The `templateParams` element returns a set of `param` elements. The `param` elements return data specific to the type of activity object returned. The possible types of param elements include:

- `user` – Returns the `displayName`, `guid`, `id`, `key`, `primaryId`, and `type`.

- `document` – Returns the `displayName`, `iconUrl`, `id`, `key`, `primaryId`, and `type`.

- `custom` – Returns `displayName`, `key`, and `type`, and may or may not have a URL.

A param can also be a variable reference or key of the general form `{prefix[index].variable}`.

C.14.1.4.13 urn:oracle:webcenter:activities:activity:icon

Use this resource type to get the icon of the activity, if available (`GET`).

C.14.1.4.14 Navigation Paths to activities:activity:icon

This section shows how the client can navigate through the hypermedia to access this resource:

```
resourceindex
   activities
      activity
         icon
```

C.14.1.4.15 Supported Methods for icon

The following methods are supported by this resource:

- `GET` – Returns the icon used for the named activity.

C.14.1.4.16 Resource Types Linked to from icon

Table C-8 lists the resource types that the client can link to from this resource.

Table C-8   Related Resource Types for urn:oracle:webcenter:activities:activity:icon

rel	resourceType
self	
urn:oracle:webcenter:parent	urn:oracle:webcenter:activities:activity

C.14.2 Connections and Profile REST API

Use the Connections and Profile REST API to browse a profile or a connections list, manage connections lists, add or remove connections, send invitations to connect, and update a profile status message. This section provides information about the REST API methods to use to perform these actions.

This section includes the following topics:

- Connections and Profile Entry Point

- Connections and Profile Resource Type Taxonomy

- Connections and Profile Security Considerations

- Connections and Profile Resource Types

C.14.2.1 Connections and Profile Entry Point

Each REST service has a link element within the Resource Index that provides the entry point for that service. For People Connections, each feature has its own link element. For example, to find the entry points for the People Connections' Connections and Profile features, find the link elements with a `resourceType` of:

`urn:oracle:webcenter:people` (returns the current user profile)

`urn:oracle:webcenter:people:person` (enables you to query for a user)

`urn:oracle:webcenter:people:invitations` (returns invitations sent or received by the current user)

Note:

The `people:person` and `people:invitations` resource types have a `template` but not an `href`.

The corresponding `href` or `template` element provides the URI entry point, which returns a list of people (`people`) or an individual user (`people:person`). The client sends HTTP requests to this entry point to work with the People Connections' Connections and Profile features.

C.14.2.2 Connections and Profile Resource Type Taxonomy

When the client has identified the entry point, it can then navigate through the resource type taxonomy to perform the required operations. For more information about the individual resource types, see the appropriate section in Connections and Profile Resource Types.

The resource type taxonomy for the People Connections' Connections and Profile features is:

```
urn:oracle:webcenter:people
   urn:oracle:webcenter:people:person
   urn:oracle:webcenter:people:icon
   urn:oracle:webcenter:people:person:list
      urn:oracle:webcenter:people:person:listNames
      urn:oracle:webcenter:people:person:listName
       urn:oracle:webcenter:people:person:list:member
      urn:oracle:webcenter:people:person:status
   urn:oracle:webcenter:people:invitations
      urn:oracle:webcenter:people:invitation
```

C.14.2.3 Connections and Profile Security Considerations

You must be logged in to the REST service to access any of the People Connections REST APIs. After that, the underlying service handles permission checking and the like.

C.14.2.4 Connections and Profile Resource Types

This section provides you with all the information you need to know about each resource type. It includes the following topics:

- urn:oracle:webcenter:people

- Navigation Paths to people

- Supported Methods for people

- Predefined Sets for the links Parameter

- Resource Types Linked to from people

- urn:oracle:webcenter:people:icon

- Supported Methods for icon

- Resource Types Linked to from icon

- urn:oracle:webcenter:people:person

- Supported Methods for person

- Read-only Elements for person

C.14.2.4.1 urn:oracle:webcenter:people

The `people` response contains URIs for use in retrieving the profile of one or more users.

C.14.2.4.2 Navigation Paths to people

This section shows how the client can navigate through the hypermedia to access the `people` resource:

```
resourceIndex
   people
```

C.14.2.4.3 Supported Methods for people

The following method is supported by the `people` resource:

- `GET`

  - **request—Parameters:**

    - `startIndex` – See Common Request Query Parameters.

    - `itemsPerPage` – See Common Request Query Parameters.

    - `projection` – See Common Request Query Parameters.

    - `data` – The data parameter is a comma separated list that controls which data will be returned in the response. The predefined set `basic` is equivalent to `data=guid,id,displayName`. The predefined set `data` is a standard set that returns all the data for the user, but does not include `status`, `manager`, `reportees`, or `photos`. The full list of possible values includes the predefined sets `basic` and `data`, as well as the following individual data values: `guid`, `id`, `displayName`, `birthday`, `language`, `timeZone`, `name`, `addresses`, `organizations`, `workEmail`, `phoneNumbers`, `manager`, `reportees`, `photos`, and `status`.

      If you specify the constant `'data'` as the `data` parameter, all the basic information will be returned for the resource. If both the `projection` and `data` query string parameters are present, the `data` parameter will be used to determine which data to return.

      The `data` parameter can also take any of the following values comma-separated values to return the corresponding data: `guid`, `id`, `displayName`, `birthday`, `language`, `timeZone`, `name`, `addresses`, `organizations`, `workEmail`, `phoneNumbers`, `manager`, `reportees`, `photos`, and/or `status`.

> **Note:**
>
> The data parameter can accept a predefined set, a collection of values, or a mix of sets and values. For example, to get the basic data plus the user's birthday, you can specify `data=basic,birthday`.

- ◆ `links` – The links parameter is a comma-separated list that controls which links will be returned in the response. This parameter can accept predefined sets, individual data values, or a combination of predefined sets and individual data values. The predefined sets are `basic`, `data`, `activitiesSet`, `connectionsSet`, and `feedbackSet`. These predefined sets are described in Predefined Sets for the links Parameter.

  The individual values for the `links` parameter are: `person`, `profile`, `icon`, `status`, `messageBoard`, `activities`, `personActivities`, `connectionActivities`, `connections`, `listNames`, `invitation`, `givenFeedback`, `receivedFeedback`, `userGivenFeedback`, `manager`, `reportees`, `member`.

  If both the `projection` and `links` query string parameters are present, the `links` parameter will be used to determine which links to return.

  - – **response**—**Body:** One or more person objects.

C.14.2.4.4 Predefined Sets for the links Parameter

The following items are predefined sets that can be returned in the `links` parameter. For example, if you specify `links=basic`, it is the equivalent of specifying `data=person,profile,icon`. You can also specify additional parameters as needed. For example, you could specify `data=basic,birthday`.

> **Note:**
>
> Links that are not currently available will not be returned even if they are specified in the `links` parameter.

- • `basic` – A standard set that returns the basic information for the profile and includes `person`, `profile`, and `icon`.

- • `data` – A standard set that returns all the basic links for the response plus the connections list, status, and activity stream template.

- • `activitiesSet` – Includes `activities`, `personActivities`, and `connectionActivities`.

- • `connectionsSet` – Includes `connections`, `listNames`, and `invitation`.

- • `feedbackSet` – Includes `givenFeedback`, `receivedFeedback`, and `userGivenFeedback`.

C.14.2.4.5 Resource Types Linked to from people

Table C-9 lists the resource types that the client can link to from the `people` resource.

Table C-9   Related Resource Types for people

Table C-9   (Cont.) Related Resource Types for people

rel	resourceType
urn:oracle:webcenter:people:icon	urn:oracle:webcenter:people:person
urn:oracle:webcenter:people:person:list:connections	urn:oracle:webcenter:people:person:list
urn:oracle:webcenter:activities:stream:person	urn:oracle:webcenter:activities:stream
urn:oracle:webcenter:activities:stream:connections	urn:oracle:webcenter:activities:stream
urn:oracle:webcenter:activities:stream	urn:oracle:webcenter:activities:stream
urn:oracle:webcenter:feedback:all-received	urn:oracle:webcenter:feedback
urn:oracle:webcenter:feedback:all-given	urn:oracle:webcenter:feedback
self	urn:oracle:webcenter:people:person:status
urn:oracle:webcenter:people:person:list:list	urn:oracle:webcenter:people:person:list
self	urn:oracle:webcenter:people:person:listName
urn:oracle:webcenter:activities:stream:list	urn:oracle:webcenter:activities:stream

C.14.2.4.6 urn:oracle:webcenter:people:icon

Use this resource type to get the icon used for the named profile (GET).

Navigation Paths to people:icon

This section shows how the client can navigate through the hypermedia to access this resource:

```
resourceindex
   people
      icon
```

C.14.2.4.7 Supported Methods for icon

The following methods are supported by this resource:

- GET – Returns the icon used for the named profile.

> **Note:**
>
> This resource includes a template that lets you choose the size of the profile icon to use. The size template parameter can be set to small, medium, or large.

C.14.2.4.8 Resource Types Linked to from icon

Table C-10 lists the resource types that the client can link to from this resource.

Table C-10   Related Resource Types for urn:oracle:webcenter:people:icon

Table C-10   (Cont.) Related Resource Types for urn:oracle:webcenter:people:icon

rel	resourceType
self	urn:oracle:webcenter:people:icon
urn:oracle:webcenter:parent	urn:oracle:webcenter:people

C.14.2.4.9 urn:oracle:webcenter:people:person

The `person` response contains profile data and the URIs for use in retrieving a user profile.

C.14.2.4.10 Navigation Paths to person

This section shows how the client can navigate through the hypermedia to access the `person` resource:

```
resourceIndex
   people

resourceIndex
   people
      person

resourceIndex
   person

author
   person

resourceIndex
   activities:stream
      person
```

C.14.2.4.11 Supported Methods for person

The following method is supported by the `person` resource:

- `GET`

  – **request—Parameters:** `q`

    To retrieve a specified person, the format of `q` is:

    ```
    q=[loginid:equals:username]
    Or
    q=[guid:equals:guid]
    Or
    q=[email:equals:email]
    ```

    Note:

    The parameter `q` is only on the `resourceIndex` template for `person`.

  – **response—Body:** `message`

See Also:

For information about the response `message`, see Read-only Elements for person.

C.14.2.4.12 Read-only Elements for person

Table C-11 lists the read-only elements for the `person` resource.

Note:

The elements present in a `person` response depend on how the user repository is configured and the elements it supports. Additionally, several of the pieces of data represented in Table C-11, such as `address`, `emails`, `photos`, `phoneNumbers`, and `organization`, can have multiple instances.

See Also:

Some of the elements listed in Table C-11 can be returned in predefined sets described in Predefined Sets for the links Parameter.

Table C-11   Read-Only Elements for person

Element	Type	Description
guid	String	Unique GUID of the person
id	String	Unique login ID of the person (that is, the user name, for example, pat_coi)
displayName	String	Display name of the person (the user's name, for example, Pat Coi). This may have the same value as `id`, depending on the repository configuration.
birthday	Date (String)[1]	Birth date of the person
connected	Boolean (String)1	Whether or not this person is connected to the current user
language	String	Preferred language of the person
timeZone	String	Time zone of the person
name	name	Name information for the person `name` is a portable contact type. For more information, see name Portable Contact Type.
address	address	Address information for the person `address` is a portable contact type. For more information, see address Portable Contact Type.

Table C-11　(Cont.) Read-Only Elements for person

Element	Type	Description
emails	value	Emails for the person emails is derived from the value portable contact type. For more information, see value Portable Contact Type.
photos	value	Profile photos for the person photos is derived from the value portable contact type. For more information, see value Portable Contact Type.
phoneNumbers	value	Phone numbers for the person phoneNumber is derived from the value portable contact type. For more information, see value Portable Contact Type.
organizations	organization	Organization information for the person organization is a portable contact type. For more information, see organization Portable Contact Type
manager	personReference	Manager of this person
reportees	personReference	Direct reports of this person
status	urn:oracle:webcenter:people:person:status	Person's profile status message

[1]　Data types, such as DATE and BOOLEAN, are stored in the API as STRING.

C.14.2.4.13 Resource Types Linked to from person

Table C-12 lists the resource types that the client can link to from the person resource.

Table C-12　Related Resource Types for person

rel	resourceType
self	urn:oracle:webcenter:people:person
alternate	urn:oracle:webcenter:spaces:profile (HTML)
urn:oracle:webcenter:people:person:list:connections	urn:oracle:webcenter:people:person:list
	urn:oracle:webcenter:people:person:listNames
	urn:oracle:webcenter:people:person:status
urn:oracle:webcenter:activities:stream:person	urn:oracle:webcenter:activities:stream
urn:oracle:webcenter:activities:stream:connections	urn:oracle:webcenter:activities:stream
urn:oracle:webcenter:activities:stream	urn:oracle:webcenter:activities:stream

Table C-12   (Cont.) Related Resource Types for person

rel	resourceType
	urn:oracle:webcenter:messageBoard
urn:oracle:webcenter:feedback:all-given	urn:oracle:webcenter:feedback
urn:oracle:webcenter:feedback:all-received	urn:oracle:webcenter:feedback

C.14.2.4.14 urn:oracle:webcenter:people:person:list

The `list` response contains URIs for use in retrieving all the profiles on a connections list (GET), inviting a user to be a connection or adding a connection to a connections list (POST), and removing a connections list (DELETE).

C.14.2.4.15 Navigation Paths to list

This section shows how the client can navigate through the hypermedia to access the `list` resource:

```
resourceIndex
   person
      listNames
         list

resourceIndex
   person
      list (rel="urn:oracle:webcenter:people:person:list:connections")
```

C.14.2.4.16 Supported Methods for list

The following methods are supported by the `list` resource:

- GET

  – **request—Parameters:** `startIndex, itemsPerPage, projection`

  > **See Also:**
  >
  > For information about REST API parameters, such as `startIndex` and `itemsPerPage`, see Common Request Query Parameters.

  – **response—Body:** 0 or more `person` items

- POST

  – **request—Body:** `member`

  – **response—Body:** `member`

  > **See Also:**
  >
  > For information about `member` in the request and response elements, see urn:oracle:webcenter:people:person:list:member.

- DELETE

– **request**

C.14.2.4.17 Resource Types Linked to from list

Table C-13 lists the resource types that the client can link to from the list resource.

Table C-13   Related Resource Types for list

rel	resourceType
self urn:oracle:webcenter:people:person:list[1]	urn:oracle:webcenter:people:person:list
urn:oracle:webcenter:activities:stream	urn:oracle:webcenter:activities:stream

[1] `self rel` currently includes `"urn:oracle:webcenter:people:person:list:list"` instead of the correct `"urn:oracle:webcenter:people:person:list"`. For the `@connections` default list, it currently includes `"urn:oracle:webcenter:people:person:list:connections"`.

C.14.2.4.18 urn:oracle:webcenter:people:person:listNames

The listNames response contains the names of existing connections lists as well as the URIs for use in retrieving the lists (GET) and creating connections lists (POST).

C.14.2.4.19 Navigation Paths to listNames

This section shows how the client can navigate through the hypermedia to access the listNames resource:

```
resourceIndex
   person
      listNames
```

C.14.2.4.20 Supported Methods for listNames

The following methods are supported by the listNames resource:

- GET

    – **request**

    – **response**: **Body:** 0 or more listName items

- POST

    – **request—Body:** listName

    – **response—Body:** listName

See Also:

For information about listName, see
urn:oracle:webcenter:people:person:listName.

C.14.2.4.21 Resource Types Linked to from listNames

Table C-14 lists the resource types that the client can link to from the listNames resource.

Table C–14   Related Resource Types for listNames

rel	resourceType
self	urn:oracle:webcenter:people:person:listNames

C.14.2.4.22 urn:oracle:webcenter:people:person:listName

The `listName` response contains the names of connections lists and URIs for use in accessing the connections lists. See also urn:oracle:webcenter:people:person:listName.

C.14.2.4.23 Navigation Paths to listName

This section shows how the client can navigate through the hypermedia to access the `listName` resource:

```
resourceIndex
    person
        listNames
            listName
```

C.14.2.4.24 Supported Methods for listName

The following method is supported by the `listName` resource:

- DELETE

    – **request**

C.14.2.4.25 Writable Elements for listName

Table C–15 lists the writable elements for the `listName` resource.

Table C–15   Writable Elements for listName

Element	Type	Required	Constraints	Description
name	String	Yes	1 or more characters	A single list name

C.14.2.4.26 Resource Types Linked to from listName

Table C–16 lists the resource types that the client can link to from the `listName` resource.

Table C–16   Related Resource Types for listName

rel	resourceType
self urn:oracle:webcenter:people:person:list	urn:oracle:webcenter:people:person:list
urn:oracle:webcenter:activities:stream:list	urn:oracle:webcenter:activities:stream

C.14.2.4.27 urn:oracle:webcenter:people:person:list:member

The `member` response contains URIs for use in deleting a connection from a connections list.

C.14.2.4.28 Navigation Paths to member

This section shows how the client can navigate through the hypermedia to access the `member` resource:

```
resourceIndex
    person
        list
            member
```

C.14.2.4.29 Supported Methods for member

The following method is supported by the `member` resource:

- `DELETE`

  – **request**

C.14.2.4.30 Writable Elements for member

Table C-17 lists the writable elements for the `member` resource. Writable elements for `member` are used when you add a connection to a list or invite a user to be a connection. The `member` resource itself is for deleting connections, and does not use writable elements.

Table C-17   Writable Elements for member

Element	Type	Required	Constraints	Description
guid	String	Yes	1 or more characters	GUID of the user
message	String	No	0 or more characters	Invitation message Use this only when inviting users to be connections (that is POSTing to the `@connections` list, not to user-created connections lists).

C.14.2.4.31 urn:oracle:webcenter:people:person:status

The `status` response contains URIs for use in retrieving (`GET`) and updating (`PUT`) the profile status message of a specified user.

C.14.2.4.32 Navigation Paths to status

This section shows how the client can navigate through the hypermedia to access the `status` resource:

```
resourceIndex
    people
        person
            status
```

C.14.2.4.33 Supported Methods for status

The following methods are supported by the `status` resource:

- `GET`

- – **request**

- – **response—Body:** `status`

- PUT

  - – **request—Body:** `status`

  - – **response—Body:** `status`

C.14.2.4.34 Writable Elements for status

Table C-18 lists the writable elements for the `status` resource.

Table C-18   Writable Elements for status

Element	Type	Required	Constraints	Description
note	String	Yes	1 or more characters	Content of status message

C.14.2.4.35 Resource Types Linked to from status

Table C-19 lists the resource types that the client can link to from the `status` resource.

Table C-19   Related Resource Types for status

rel	resourceType
self	urn:oracle:webcenter:people:person:status

C.14.2.4.36 urn:oracle:webcenter:people:invitations

The `invitations` response contains URIs for use in retrieving invitations (`GET`). You can also send an invitation (`POST`) to another user.

C.14.2.4.37 Navigation Paths to invitations

This section shows how the client can navigate through the hypermedia to access the `invitations` resource:

```
resourceIndex
   invitations
```

If you are not already connected to a user, you can also navigate to the invitations resource from that user's profile in order to invite them to connect. This path is only used for POSTing.

```
resourceIndex
   person
      invitations
```

C.14.2.4.38 Supported Methods for invitations

The following methods are supported by the `invitations` resource:

- GET

  - – **Request—Parameters:** `q`

    To retrieve invitations sent to the current user, the format of `q` is:

```
q=[invitee:equals:@me]
```

To retrieve invitations sent from the current user, the format of `q` is:

```
q=[invitor:equals:@me]
```

– **Response—Body:** 1 or more invitations

- POST

  – **Request—Body:** `invitation`

  – **Response—Body:** `invitation`

C.14.2.4.39 Writable Elements for invitations

Table C-20 lists the writable elements for the `invitations` resource.

Table C-20   Writable Elements for invitations

Element	Type	Required	Constraints	Description
message	String	No	1 or more characters	Message attached to the invitation

C.14.2.4.40 Resource Types Linked to from invitations

Table C-21 lists the resource types that the client can link to from the `invitations` resource:

Table C-21   Related Resource Types for invitations

rel	resourceType
self	urn:oracle:webcenter:people:invitation

C.14.2.4.41 urn:oracle:webcenter:people:invitation

The `invitation` response contains URIs for use in deleting (`DELETE`) invitations you have sent, or deleting (`DELETE`) or updating (`PUT`) invitations you have received.

C.14.2.4.42 Navigation Paths to invitation

This section shows how the client can navigate through the hypermedia to access the `invitation` resource:

```
resourceIndex
   invitations
      invitation
```

C.14.2.4.43 Supported Methods for invitation

The following methods are supported by the `invitation` resource:

- PUT

  – **Request—Body:** `invitation`

  – **Response—Body:** `invitation`

- DELETE

– **Request**

C.14.2.4.44 Writable Elements for invitation

Table C-22 lists the writable elements for the `invitation` resource.

Table C-22   Writable Elements for invitation

Element	Type	Required	Constraints	Description
status	String	Yes (PUT)	ACCEPTED IGNORED	The status of the invitation. **Note:** When you accept or ignore an invitation, it is removed from your list of invitations.

C.14.2.4.45 Read-only Elements for invitation

Table C-23 lists the read-only elements for the `invitation` resource.

Table C-23   Read-only Elements for invitation

Element	Type	Description
id	String	Unique ID of the invitation
invitee	personReference	User to whom the invitation is sent
invitor	personReference	User from whom the invitation is sent
sentDate	Date (String)[1]	Date the invitation was sent

[1] Data types, such as `DATE` and `BOOLEAN`, are stored in the API as `STRING`.

C.14.2.4.46 Resource Types Linked to from invitation

Table C-24 lists the resource types that the client can link to from the `invitation` response.

Table C-24   Related Resource Types for invitations

rel	resourceType
self	urn:oracle:webcenter:people:invitation

C.14.3 Feedback REST API

Use the Feedback REST API to read and delete feedback. This section provides information about the REST API methods to use to perform these actions.

This section includes the following topics:

- Feedback Entry Point

- Feedback Resource Type Taxonomy

- Feedback Security Considerations

- Feedback Resource Types

C.14.3.1 Feedback Entry Point

Each REST service has a link element within the Resource Index that provides the entry point for that service. For People Connections, each feature has its own `link` element. To find the entry points for the People Connections' Feedback feature, find the `link` elements with a `resourceType` of:

```
urn:oracle:webcenter:feedback
```

The corresponding `href` or `template` element provides the URI entry point, which returns all received feedback for the current user. The client sends HTTP requests to this entry point to work with the People Connections' Feedback feature.

> **See Also:**
>
> For more information about the Resource Index, see Using the Resource Index.
>
> For more information about resource types, see Resource Type.

C.14.3.2 Feedback Resource Type Taxonomy

When the client has identified the entry point, it can then navigate through the resource type taxonomy to perform the required operations. For more information about the individual resource types, see the appropriate section in Feedback Resource Types.

The resource type taxonomy for the People Connections' Feedback feature is:

```
urn:oracle:webcenter:feedback
   urn:oracle:webcenter:feedback:message
```

C.14.3.3 Feedback Security Considerations

You must be logged into the REST service to access any of the People Connections REST APIs. After that, the underlying service handles permission checking and the like.

> **See Also:**
>
> For general security considerations, see Security Considerations for WebCenter Portal REST APIs.

C.14.3.4 Feedback Resource Types

This section provides you with all the information you need to know about each resource type. It includes the following topics:

- urn:oracle:webcenter:feedback

- Navigation Paths to feedback

- Supported Methods for feedback

- Resource Types Linked to from feedback

- urn:oracle:webcenter:feedback:message

- Navigation Paths to message

- Supported Methods from message

- Read-only Elements for message

- Resource Types Linked to from feedback

C.14.3.4.1 urn:oracle:webcenter:feedback

The `feedback` response contains URIs for use in reading Feedback messages.

C.14.3.4.2 Navigation Paths to feedback

This section shows how the client can navigate through the hypermedia to access the `feedback` resource:

```
resourceIndex
    feedback

resourceIndex
    person
        feedback
```

C.14.3.4.3 Supported Methods for feedback

The following methods are supported by the `feedback` resource:

- `GET`

  – **request—Parameters:** `startIndex`, `itemsPerPage`

    See Also:

    For information about REST API parameters, such as `startIndex` and `itemsPerPage`, see Common Request Query Parameters.

  – **response—Body:** `message`

    See Also:

    For information about `message`, see urn:oracle:webcenter:feedback:message.

- `POST` – If permitted, lets you add feedback for a target user. This method is only available if the current user is connected to and has permission to add feedback for the target user.

  – **request - body**: feedback

```
<message resourceType="urn:oracle:webcenter:feedback:message">
    <body>test from REST API</body>
    <receivedUser>
    <guid>4F16DD80393611DFBF895F177662C511</guid>
    </receivedUser>
</message>
```

C.14.3.4.4 Resource Types Linked to from feedback

Table C-25 lists the resource types that the client can link to from the `feedback` resource.

Table C-25   Related Resource Types for feedback

rel	resourceType
self urn:oracle:webcenter:feedback:all-received	urn:oracle:webcenter:feedback
self urn:oracle:webcenter:feedback:all-given	urn:oracle:webcenter:feedback
NA	urn:oracle:webcenter:feedback:message

C.14.3.4.5 urn:oracle:webcenter:feedback:message

The `message` response contains the feedback message data and URIs for use in deleting a Feedback message.

C.14.3.4.6 Navigation Paths to message

This section shows how the client can navigate through the hypermedia to access the `message` resource:

```
resourceIndex
    feedback
        message

resourceIndex
    person
        feedback
            message
```

C.14.3.4.7 Supported Methods from message

The following method is supported by the `message` resource:

- `DELETE`

    – **request**

C.14.3.4.8 Read-only Elements for message

Table C-26 lists the read-only elements for the `message` resource.

Table C-26   Read-only Elements for message

Element	Type	Description
`body`	String	Message content
`id`	String	Unique ID of the message
`author`	personReference	User who created the message
`created`	Date (String)[1]	Date the message was created

Table C-26   (Cont.) Read-only Elements for message

Element	Type	Description
receivedUser	personReference	A person reference to the user who received the feedback

[1] Data types, such as DATE and BOOLEAN, are stored in the API as STRING.

C.14.3.4.9 Resource Types Linked to from feedback

Table C-27 lists the resource types that the client can link to from the feedback resource.

Table C-27   Related Resource Types for message

rel	resourceType
self	urn:oracle:webcenter:feedback:message

C.14.4 Message Board REST API

Use the Message Board REST API to post, read, and delete messages to a user's or a portal message board. This section provides information about the REST API methods to use to perform these actions.

This section includes the following topics:

- Message Board Entry Point

- Message Board Resource Type Taxonomy

- Message Board Security Considerations

- Message Board Resource Types

- Filtering Messages Based on Visibility

C.14.4.1 Message Board Entry Point

Each REST service has a link element within the Resource Index that provides the entry point for that service. For People Connections, each feature has its own link element. To find the entry points for the People Connections' Message Board feature, find the link elements with a resourceType of:

urn:oracle:webcenter:messageBoard

Note:

As well as an entry point from the Resource Index, to navigate to an individual user's message board, the Message Board feature also has an entry point from a portal response for the portal message board.

The corresponding href or template element provides the URI entry point, which returns the Message Board for the current user. The client sends HTTP requests to this entry point to work with People Connections' Message Board feature.

> **See Also:**
>
> For more information about the Resource Index, see Using the Resource Index.
>
> For more information about resource types, see Resource Type.

C.14.4.2 Message Board Resource Type Taxonomy

When the client has identified the entry point, it can then navigate through the resource type taxonomy to perform the required operations. For more information about the individual resource types, see the appropriate section in Message Board Resource Types.

The resource type taxonomy for the Message Board feature for People Connections is:

```
urn:oracle:webcenter:messageBoard
   urn:oracle:webcenter:messageBoard:message
```

C.14.4.3 Message Board Security Considerations

You must be logged into the REST service to access any of the People Connections REST APIs. After that, the underlying service handles permission checking and the like.

> **See Also:**
>
> For general security considerations, see Security Considerations for WebCenter Portal REST APIs

C.14.4.4 Message Board Resource Types

This section provides you with all the information you need to know about each resource type. It includes the following topics:

- urn:oracle:webcenter:messageBoard
- Navigation Paths to messageBoard
- Supported Methods for messageBoard
- Read-only Elements for messageBoard
- Resource Types Linked to from messageBoard
- urn:oracle:webcenter:messageBoard:message
- Navigation Paths to message
- Supported Methods for message
- Writable Elements for message
- Read-only Elements for message
- Resource Types Linked to from message

C.14.4.4.1 urn:oracle:webcenter:messageBoard

The `messageBoard` response contains URIs for use in reading (`GET`) and posting (`POST`) portal and individual user Message Board messages.

C.14.4.4.2 Navigation Paths to messageBoard

This section shows how the client can navigate through the hypermedia to access the `messageBoard` resource:

```
resourceIndex
   messageBoard

resourceIndex
   person
      messageBoard

resourceIndex
   spaces
      space
         messageBoard
```

C.14.4.4.3 Supported Methods for messageBoard

The following methods are supported by the `messageBoard` resource:

- `GET`

  – **request—Parameters:** `startIndex, itemsPerPage`

  See Also:

  For information about REST API parameters, such as `startIndex` and `itemsPerPage`, see Common Request Query Parameters.

  – **response—Body:** message

  Note:

  The REST `GET` command for reading (retrieving) messages retrieves all shown messages by default. You can also retrieve messages that are private or hidden through the application user interface as described in Filtering Messages Based on Visibility. For information about hiding and showing messages, see "Adding Messages and Feedback to a Portal" in *Building Portals with Oracle WebCenter Portal*.

- `POST`

  – **request—Body:** `message`

  The POST for messages supports including a link URL in the message.

  See Also:

  For information about message, see urn:oracle:webcenter:messageBoard:message.

C.14.4.4.4 Read-only Elements for messageBoard

Table C-28 lists the read-only elements for the `messageBoard` resource.

Table C-28 Read-only Elements for message

Element	Type	Description
messageType	String	Returns `link` if the message has a link. Otherwise, returns `null`.
link	String	Contains link data for messages with links: `name`, `url`, `icon`, `description`, `mimeType`, `objectId`, `objectType`, `serviceId`.

C.14.4.4.5 Resource Types Linked to from messageBoard

Table C-29 lists the resource types that the client can link to from the `messageBoard` resource.

Table C-29 Related Resource Types for messageBoard

rel	resourceType
self	urn:oracle:webcenter:messageBoard
NA	urn:oracle:webcenter:messageBoard:message

C.14.4.4.6 urn:oracle:webcenter:messageBoard:message

The `message` response contains the Message Board message data and URIs for use in reading (`GET`), revising (`PUT`), and deleting (`DELETE`) a portal or individual user Message Board message.

C.14.4.4.7 Navigation Paths to message

This section shows how the client can navigate through the hypermedia to access the `message` resource:

```
resourceIndex
   messageBoard
      message

resourceIndex
   person
      messageBoard
         message

resourceIndex
   spaces
      space
         messageBoard
            message
```

C.14.4.4.8 Supported Methods for message

The following methods are supported by the `message` resource:

- GET

- **request**

- **response—Body:** `message`

- PUT

  - **request—Body:** `message`

  - **response—Body:** `message`

- DELETE

  - **request**

C.14.4.4.9 Writable Elements for message

Table C-30 lists the writable elements for the `message` resource.

Table C-30   Writable Elements for message

Element	Type	Required	Constraints	Description
body	String	Yes	1 or more characters	message content

C.14.4.4.10 Read-only Elements for message

Table C-31 lists the read-only elements for the `message` resource.

Table C-31   Read-only Elements for message

Element	Type	Description
id	String	Unique ID of the message
author	personReference	User who created the message
created	Date (String)[1]	Date the message was created

[1]   Data types, such as DATE and BOOLEAN, are stored in the API as STRING.

C.14.4.4.11 Resource Types Linked to from message

Table C-32 lists the resource types that the client can link to from the `message` resource.

Table C-32   Related Resource Types for message

rel	resourceType
self	urn:oracle:webcenter:messageBoard:message

C.14.4.5 Filtering Messages Based on Visibility

Although the REST GET command for retrieving messages retrieves all shown messages by default, you can also retrieve messages that are private or hidden through the application user interface using a set of visibility filters.

Messages posted on message boards can have the following visibility criteria:

- Public

- Private

- Hidden

- Public and hidden

- Private and hidden

By default, messages are public.

C.14.4.5.1 Private Messages:

A person owning a message board can mark any message as private. When a message is marked as private, that message will not be visible to anyone other than the owner of the message unless the owner chooses to send a private message to someone else. Otherwise, other people viewing a message board with private messages will see only public messages.

C.14.4.5.2 Hidden Messages:

A person owning a message board can mark any message as hidden. When a message is marked as hidden, that message will not be visible to the owner of the message board, but will remain visible to other people viewing the message board.

C.14.4.5.3 REST URLs for Message Boards:

The context for message board URL filters is:

```
rest/api/messageBoards/<BOARD-TYPE>/<GUID>/<FILTER-TYPE>
```

Where:

- `<BOARD-TYPE>` is either `person` or `space`

- `<GUID >` is either `@me`, the `person` `<GUID>` (if `<BOARD-TYPE>` is person) or the `space` `<GUID>` (if `<BOARD-TYPE>` is `portal`)

- `<FILTER-TYPE>` is applicable only for the `person` `<BOARD-TYPE>` and can be one of:

  - `null` - (default) shows all messages

  - `private` - shows private messages

  - `public` - shows public messages

  - `hidden` - shows hidden and public messages

  - `private_hidden` - shows private and hidden messages

Available filters in the context of the applicable HTML methods are shown below.

GET

- `@me`

  - all `rest/api/messageBoards/person/@me`

  - private `rest/api/messageBoards/person/@me/private`

  - public `rest/api/messageBoards/person/@me/public`

– hidden and public `rest/api/messageBoards/person/@me/hidden`

– private and hidden `rest/api/messageBoards/person/@me/private_hidden`

- `person`

  `rest/api/messageBoards/person/<GUID>`

  If the GUID matches that of the logged in user then the same filtering as for @me applies. If the GUID is different, then no filtering is available.

- `space`

  `rest/api/messageBoards/space/<GUID>`

  No filtering based on visibility is available.

POST

- @me

  – all rest/api/messageBoards/person/@me

  – private

    ```
    {"body" : "<BODY_CONTENT>",
     "visibilityType" : "private"}
    ```

  – public

    ```
    {"body" : "<BODY_CONTENT>",
     "visibilityType" : "public"}
    ```

  – hidden and public

    ```
    {"body" : "<BODY_CONTENT>",
     "visibilityType" : "hidden"}
    ```

  – private and hidden

    ```
    {"body" : "<BODY_CONTENT>",
     "visibilityType" : "private_hidden"}
    ```

- `person` `rest/api/messageBoards/person/<GUID>` If the GUID is the same as the logged in user, then the filters for GET apply. If the users are different, then the following filters apply:

  – public

    ```
    {"body" : "<BODY_CONTENT>",
     "visibilityType" : "public"}
    ```

  – private

    ```
    {"body" : "<BODY_CONTENT>",
     "visibilityType" : "private"}
    ```

- `space`

  `rest/api/messageBoards/space/<GUID>`

  No filtering based on visibility is available.

PUT

- @me

  - all est/api/messageBoards/person/@me/messages/<msg guid>

  - private

    ```
    {"body" : "<BODY_CONTENT>",
     "visibilityType" : "private"}
    ```

  - public

    ```
    {"body" : "<BODY_CONTENT>",
     "visibilityType" : "public"}
    ```

  - hidden and public

    ```
    {"body" : "<BODY_CONTENT>",
     "visibilityType" : "hidden"}
    ```

  - private and hidden

    ```
    {"body" : "<BODY_CONTENT>",
     "visibilityType" : "private_hidden"}
    ```

- `person` rest/api/messageBoards/person/<GUID>/messages/<msg guid> If the GUID is the same as the logged in user, then the filters for GET apply. If the users are different, then the following filters apply:

  - public

    ```
    {"body" : "<BODY_CONTENT>",
     "visibilityType" : "public"}
    ```

  - private

    ```
    {"body" : "<BODY_CONTENT>",
     "visibilityType" : "private"}
    ```

- `space`

  `rest/api/messageBoards/space/<GUID>/messages/<msg guid>`

  No filtering based on visibility is available.

C.14.5 Creating an Invitation

This section illustrates how to invite another user to join your connections list using the People Connections Service REST API. After the invitation is made, the invitee is given the option to accept or not. This example also illustrates how to delete an invitation.

This section includes the following topics:

- Creating an Invitation

- Accepting an Invitation

- Deleting an Invitation

C.14.5.1 Creating an Invitation

1. The first step, as always with REST API methods, is to retrieve the resource index:

   ```
   GET http://<host:port>/rest/api/resourceIndex
   ```

2. Next, find the person to whom you want to connect. To do this:

a. In the resource index listing, scan for the link with the following resource type:

```
urn:oracle:webcenter:people:person
```

b. Execute a search for the user who you wish to invite. In this example, the user is named Monty.

```
GET http://<host:port>/rest/api/people?q=loginid:equals:monty&utoken=ASDF
```

c. Locate Monty's GUID in the response, as shown in Figure C-2.

Figure C-2   Response With User's GUID

```
</links>
<connected>false</connected>
<displayName>monty</displayName>
<guid>1AE5AF102E2611E09F062B573E287934</guid>
<id>monty</id>
─ <name>
    <formatted>monty</formatted>
</name>
```

d. Save the GUID so that you can use it to connect to the user.

3. Now that you have the invitee's GUID, you must find the resource to which you would like to add him. In this case, you want to add him to your own `connections` list.

To find your `connections` list, first, scan these People Connections topics for "`connections`." You will discover in Table C-12 that `connections` are linked to from the `person` resource. For convenience, this table is shown in Figure C-3.

Figure C-3   Finding the connections Link

rel	resourceType
self	urn:oracle:webcenter:people:person
alternate	urn:oracle:webcenter:spaces:profile (HTML)
urn:oracle:webcenter:people:person:list:connections	urn:oracle:webcenter:people:person:list
	urn:oracle:webcenter:people:person:listNames
	urn:oracle:webcenter:people:person:status
urn:oracle:webcenter:activities:stream:person	urn:oracle:webcenter:activities:stream
urn:oracle:webcenter:activities:stream:connections	urn:oracle:webcenter:activities:stream
urn:oracle:webcenter:activities:stream	urn:oracle:webcenter:activities:stream
	urn:oracle:webcenter:messageBoard
urn:oracle:webcenter:feedback:all-given	urn:oracle:webcenter:feedback
urn:oracle:webcenter:feedback:all-received	urn:oracle:webcenter:feedback

Now that you know that `connections` are linked to from the `person` resource, you need to find the `person` resource. As the URN indicates, you get to the `person` resource from the `people` resource, as described in the following steps.

4. Return to the resource index (or use a cached version of the `resourceIndex` from your previous visit):

```
GET http://<host:port>/rest/api/resourceIndex
```

5. Scan for the `people` resource, and you will find:

```
urn:oracle:webcenter:people
```

6. Use the link in the `people` resource to access *your* lists:

```
GET http://<host:port>/rest/api/people/@me/lists/@self?utoken=ASDF
```

7. Scan the returned links for the `resourceType` and `rel` listed in Table C-12:

```
resourceType="urn:oracle:webcenter:people:person:list
rel="urn:oracle:webcenter:people:person:list:connections
```

8. Make the invitation by using the URI to your `connections` list to execute a POST:

```
POST http://<host:port>/rest/api/people/@me/lists/@connections?utoken=ASDF

Headers --  Accept:application/json, Content-Type:application/json
Body -- {"message":"Monty, do you want to join my connections
list?","guid":"1AE5AF102E2611E09F062B573E287934"}
```

9. Now, if you log in to Monty's account and you can see the invitation has been added, as shown in Figure C-4.

Figure C-4   Activity Stream Showing Invitation

C.14.5.2 Accepting an Invitation

After an invitation has been sent, the next step is for the invitee (Monty) to accept the invitation.

1. Using a second REST client, retrieve the resource index:

```
GET http://<host:port>/rest/api/resourceIndex
```

2. Login as Monty.

3. List all the invitations sent to Monty by making the following request:

```
GET http://<host:port>/rest/api/people/invitations?
q=invitee:equals:@me&utoken=ASDF
```

Each invitation element listed in the response contains a link that supports the UPDATE operation and that looks something like this:

```
<links>
<link capabilities="urn:oracle:webcenter:delete urn:oracle:webcenter:update"
href="http://<host:port>/rest/api/people/invitations/<invitationid>?utoken=ASDF"
rel="self" resourceType="urn:oracle:webcenter:people:invitation"/>
</links>
<id><invitationid></id>
```

4. To accept the invitation, make the following request:

```
PUT http://<host:port>/rest/api/people/invitations/<invitationid>?utoken=ASDF
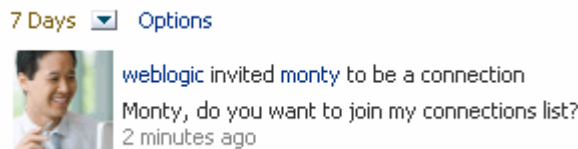
Headers
  Accept -> application/xml
  Content-Type -> application/xml

Body
  <invitation>
      <id><invitationid></id>
      <status>accepted</status>
  </invitation>
```

C.14.5.3 Deleting an Invitation

After you initiate an invitation, you can view the invitation from your account by specifying `invitor:equals:@me`. For example:

```
GET http://<host:port>/rest/api/people/invitations?
q=invitor:equals:@me&utoken=ASDF
```

Each invitation element listed in the response contains a link that supports the DELETE operation and that looks something like this:

```
<links>
<link resourceType="urn:oracle:webcenter:people:invitation"
rel="self"
href="http://host_name:port_name/rest/api/people/invitations/
      e9073cdb-56ab-423d-8b1f-1220c802bdd4?
      utoken="FN0SEFwX42OCntwtx9a1dSbhqocO_w**"
      capabilities="urn:oracle:webcenter:delete"/>
</links>
```

The invitee can also delete an invitation from his or her own account. The invitee can get a list of his or her invitations by specifying:

```
GET http://<host:port>/rest/api/people/invitations?
q=invitee:equals:@me&utoken=ASDF
```

Note that the response from a DELETE is simply a status code of 204.

C.15 Content Management REST API

This section supplements the OASIS CMIS specification, and provides details on the specific implementation of the .

The OASIS CMIS (Content Management Interoperability Services) Technical Committee works to standardize a Web services interface specification that will enable greater interoperability of Enterprise Content Management (ECM) systems. For more information, see the OASIS CMIS site:

http://www.oasis-open.org/committees/cmis/

Before continuing, all users should review the OASIS CMIS specification. This section references the Content Management Interoperability Services (CMIS) Version 1.0, which can be viewed at the following URL:

http://docs.oasis-open.org/cmis/CMIS/v1.0/cmis-spec-v1.0.html.

The provides a server that uses the CMIS RESTful AtomPub server binding to provide access to Oracle Content Server repositories configured in your application.

The specification includes the domain model and two server bindings. As mentioned above, only the RESTful AtomPub binding is currently implemented by the . Users should be familiar with Atom and AtomPub, which are the default formats for responses.

> **Note:**
>
> CMIS provides a lowest common denominator for a wide range of different content systems; it is not aligned directly with Oracle WebCenter Content functionality. Refer to the service document to identify available functionality.

This section includes the following topics:

- CMIS Domain Model

- CMIS Part II: RESTful AtomPub Binding

- Best Practices and Examples

C.15.1 CMIS Domain Model

Section 2: Domain Model in the CMIS specification defines a domain model that can be used by applications to work with one or more Content Management repositories.

- Data Model

- Services

C.15.1.1 Data Model

The service document consists of AtomPub workspaces. Each workspace maps to a content connection (only Oracle WebCenter Content repositories are supported by the). For details on the service document, see the next section, CMIS Part II: RESTful AtomPub Binding.

C.15.1.1.1 Repository

For this release, some of the optional capabilities listed in section 2.1.1 have not been implemented. Versioning, ACL, Policies, Relationships, Change Log, Folder Descendants/Tree, and Renditions will be considered for future releases.

Specifically, the implementation has the following optional capabilities:

```
capabilityGetDescendants = true
capabilityGetFolderTree = false
capabilityContentStreamUpdatability = anytime
capabilityChanges = none
capabilityRenditions = none
capabilityMultifiling = false
capabilityUnfiling = false
capabilityVersionSpecificFiling = false
capabilityPWCUpdateable = false
capabilityPWCSearchable = false
capabilityAllVersionsSearchable = false
capabilityJoin = none
capabilityACL = none
capabilityQuery = none, metadataonly, or both combined
```

C.15.1.1.2 Object

The supports document and folder objects. In CMIS the `cmis:baseTypeId` for a Node will be `cmis:folder` or `cmis:document`. Also, the `cmis:baseId` for a Type will be `cmis:folder` or `cmis:document`.

C.15.1.1.3 Object-Type

A CMIS Object-Type contains fields mapped from the Oracle WebCenter Content: Content Server metadata field definitions and Oracle WebCenter Content's Site Studio region definitions.

The mapping from Oracle WebCenter Content metadata fields to CMIS property definitions is as follows:

Table C-33   Content Server Metadata Mappings

Oracle WebCenter Content Metadata	CMIS Property Definition
TEXT metadata field with option list configured with select list validated and YesNoView or TrueFalseView view	cmis:propertyBoolean
All other TEXT metadata fields	cmis:propertyString
LONG TEXT metadata field	cmis:propertyString
MEMO metadata field	cmis:propertyString
INTEGER metadata field	cmis:propertyInteger
DATE metadata field	cmis:propertyDateTime
DECIMAL metadata field	cmis:propertyDecimal

The mapping from Site Studio region definition fields to CMIS property definitions is as follows:

Table C-34   Site Studio Region Definition Mappings

Site Studio Region Definition	CMIS Property Definition
Image Element Definition fields	cmis:propertyString
WYSIWYG Element Definition fields	cmis:propertyString
Plain Text Element Definition fields	cmis:propertyString
Static List Element Definition fields	cmis:propertyString

C.15.1.1.4 Document Object

Document Objects are the elementary information entities managed by the repository. As defined by the CMIS specification, Document Objects may be version-able, file-able, query-able, control-able and ACLControl-able. As stated earlier, the does not support versioning, multi-filing, Policies or ACL for this release.

If a Node is determined to be a Document (not a Folder) then any children it has will not be exposed through CMIS. In CMIS, each Document Object is associated with a

single content stream, and for WebCenter CMIS REST, this stream is the Oracle WebCenter Content: Content Server binary associated with the document.

C.15.1.1.5 Folder Object

The CMIS specification states that Folder Objects do not have a content-stream and are not version-able. If a Node is determined to be a Folder, then the exposes it in this manner. (In Oracle WebCenter Content, folders do not have a content stream and are not versionable).

C.15.1.1.6 Relationship Object

Section 2.1.6: Relationship Object in the CMIS specification is not applicable; the does not support Relationships for this release.

C.15.1.1.7 Policy Object

Section 2.1.7: Policy Object in the CMIS specification is not applicable; the does not support Policies for this release.

C.15.1.1.8 Access Control

Most of *Section 2.1.8: Access Control* in the CMIS specification is not applicable; the does not support ACL for this release. See the next section for details on allowable actions.

C.15.1.1.8.1 AllowableActions Mapping

This section lists allowable actions defined for Objects. Because of how this release is implemented, some of these are hard-coded for all objects. Other allowable actions will be set based on the repository configuration.

- canGetObjectRelationships = false
- canCreateRelationship = false
- canGetDescendants = false
- canGetFolderTree = false
- canCheckOut = false (versioning)
- canCancelCheckOut = false (versioning)
- canCheckIn = false (versioning)
- canAddObjectToFolder = false (multi-filing)
- canRemoveObjectFromFolder = false (unfiling/multi-filing)
- canApplyPolicy = false
- canGetAppliedPolicies = false
- canRemovePolicy = false
- canCreatePolicy = false
- canApplyACL = false
- canGetACL = false
- canGetRenditions = false

- canDeleteTree = true

- canGetAllVersions = false (versioning)

C.15.1.1.9 Versioning

Section 2.1.9: Versioning in the CMIS specification is not applicable; the does not support versioning for this release.

C.15.1.1.10 Query

CMIS queries return a Result Set where each Entry object will contain only the properties that were specified in the query. The does not support JOINs in queries, so each result entry will represent properties from a single node. Common searches use a query like "SELECT * FROM …".

- The FROM clause specifies a content-type to be searched.

  – FROM cmis:document ==> any Oracle WebCenter Content document (for example, IDC:GlobalProfile)

  – FROM cmis:folder ==> any Oracle WebCenter Content folder (for example, IDC:Folder)

  – FROM typeQueryName ==> type's cmis queryName, as long as the type is queryable (for example, ora:t:IDC!;GlobalProfile)

- The cmis:document and cmis:folder types are always queryable. Other types will be queryable if they are searchable in the repository.

- The IN_FOLDER predicate is implemented as the folder ID specified, being the parent of the results.

- The IN_TREE predicate is implemented as the folder ID specified, being a parent in the folder structure of the results.

- The CONTAINS() predicate is a full-text query expression operator.

- Properties of cmis:document and cmis:folder will be queryable and orderable if their corresponding Oracle WebCenter Content system property is searchable and sortable. The system property mappings are:

  – cmis:createdBy ==> dDocAuthor

  – cmis:lastModifiedBy ==> dDocCreator

  – cmis:creationDate ==> dCreateDate

  – cmis:lastModificationDate ==> dLastModifiedDate (for 10g, folders map to dLastModifiedDate and documents map to dCreateDate)

  – cmis:name ==> dOriginalName (for a document) or dCollectionName (for a folder)

  – cmis:contentStreamFileName ==> dOriginalName

  – cmis:contentStreamLength ==> VaultFileSize

  – cmis:contentStreamMimeType ==> dFormat

  – cmis:objectId ==> dDocName

- cmis:objectTypeId ==> Oracle WebCenter Content profile name or SiteStudio Region Definition name

> **Note:**
>
> cmis:objectTypeId is never orderable.

- cmis:path ==> use IN_FOLDER or IN_TREE predicate

> **Note:**
>
> Some repositories may have capabilities that are not representable in a CMIS query, and some repositories may have restrictions which will limit the CMIS-query predicates (or combinations of predicates) that can be used in a query. Use the mappings above to translate repository capabilities and restrictions into corresponding considerations for CMIS queries.

- Nested properties are not queryable or orderable.

- The implementation reports as orderable any properties which Oracle WebCenter Content specifies as sortable. This list can include properties which Oracle WebCenter Content cannot actually sort on. To allow ordering on a field for which Oracle WebCenter Content reports a sort error, follow the steps below to make the specified Oracle WebCenter Content field sortable:

  1. Go to **Administration** and open **Admin Applets**.

  2. Open the Configuration Manager applet and click **Advanced Search Design.**

  3. Edit the field you wish to make orderable and select **Is sortable**.

  4. Save your changes and exit Administration.

Table C-35 lists specific search considerations and recommendations. For example queries, see Best Practices and Examples.

Table C-35   Search Considerations and Recommendations

Consideration	Recommendation
Oracle WebCenter Content provides limited support for querying on null or non-null values.	Be aware of the differences in search behavior and do not write search expressions that depend on unsupported criteria.
Recursive search for folders is not supported by Oracle WebCenter Content, but is supported for documents if configured on the Oracle WebCenter Content server as described to the right.	Scope the search to only include documents (add a select clause like `select * from cmis:document`). Set the search path on the Search object (add a where clause like `"where IN_TREE('/StellentRepository/IDC:Folder/2')`. Configure the `folders_g CollectiveSearchRecursiveContent` and related settings like `CollectionMaxBranch` in the Oracle WebCenter Content config.cfg file.

Table C-35   (Cont.) Search Considerations and Recommendations

Consideration	Recommendation
Multivalued property operators perform substring matches. This is true for `ANY <multiValuedQueryName> IN (<literal>, ...)` or `<literal> = ANY < multiValuedQueryName>`. In Oracle WebCenter Content, a field with an option list stores values in a comma-delimited manner. For example, if you have values "A", "B", and "C", these will be represented as "A, B, C". Using an ANY or ANY IN search for 'A, B' will find this item.	Be aware of the differences in search behavior and consider changing the Oracle WebCenter Content option list delimiter character in the Configuration Manager applet to reduce the potential for finding extra matches.
When searching folders (`FROM cmis:folder`), at most one value can be specified per criteria. Each criteria is logically ANDed with the others to make a more selective query. There is no support for `OR` or `NOT` operators when searching folders.	Do not use `OR` and `NOT` in Oracle WebCenter Content folder search.
Not all properties are searchable, and if the search encounters a property that is not searchable, it will return a ParseException (400 error).	Understand which properties are searchable for a given content type by examining the Oracle WebCenter Content Configuration Manager information fields section, or by reviewing the ContentType definition. Examine the URL for the `isSearchable` field setting Example URLS: `http://myContentServer/idc/ idcplg? IdcService=VCR_GET_CONTENT_TYPE&v crContentType=IDC:Folder&IsSoap=1` Or, examine the specific type through CMIS.
Not all ContentTypes are searchable. An attempt to search for a non-searchable ContentTypes will throw an exception. For example, the `IDC:FileReference` ContentType is not searchable.	Be aware that not all ContentTypes are searchable.
Only String multi-valued properties can be searched.	Do not specify search for multi-valued property types other than String.
The `NOT` operator cannot be used for `LONG` properties.	Try to restructure the query using supported syntax.
Only one sort criteria is supported.	Do not write search expressions that use more than a single sort criteria.

Table C-35   (Cont.) Search Considerations and Recommendations

Consideration	Recommendation
Sorting on non-indexed fields results in an exception. Searching on a non-indexed field throws an exception, with the embedded exception code of, for example, "DRG-10837: section dStatus does not exist".	Understand which fields have been indexed before using them as sort criteria. Examine the URL for the `isSortable` field definition. Example URL: `http://myContentServer/idc/idcplg?IdcService=GET_ADVANCED_SEARCH_OPTIONS&IsSoap=1`, Or examine the type through CMIS to see if the property definition is queryable.
Empty values are not allowed in a search query.	Do not use a criteria such as `cmis:name != ''`..
The `Notequals` operator is not supported for non-String properties	Be aware of the differences in search behavior and do not write search expressions that depend on unsupported criteria.
Multiple search paths on the same Oracle WebCenter Content repository are not supported.	Be aware of the differences in search behavior and do not write search expressions that depend on unsupported criteria.
When searching for documents, recursive search (folder tree search) is supported if Oracle WebCenter Content is configured properly. If the search path is not set, then all documents in the repository will be searched (both filed and unfiled).	Configure the content server `folders_g CollectionSearchRecursiveContent` and related settings like `CollectionMaxBranch` in the Oracle WebCenter Content config.cfg file. These settings are described in the Oracle WebCenter Content documentation. To perform a document search scoped to a folder tree, use the IN_TREE predicate.
When searching for documents and using the `LIKE` operator, wildcards (`%`) are only supported in the last path element.	Be aware of the differences in search behavior and do not write search expressions that depend upon unsupported criteria.
When searching documents (select * from cmis:document), it is not possible to limit the search to more than just a single content type; for example, this is not supported: `select * from IDC:MyProfile, IDC:AnotherProfile` because it has multiple explicit content types and `JOINS` are not supported.	If you need to limit the search to more than just a single content type, issue multiple queries to achieve the same behavior. If you want to search across all types, use cmis:document in the select statement.
Oracle WebCenter Content search does not support `OR` when `cmis:objectTypeId` is specified in a query; other parameters can be ANDed with this criteria, but `OR` is not supported.	If this functionality is necessary, issue multiple queries to achieve the same behavior.
The `cmis:objectTypeId` criteria only supports `==`, `!=`, and `like`. Use of `!=` is restricted to the case of excluding folders (which behaves the same as adding `select * from cmis:document`).	Be aware of the valid operators when using `cmis:objectTypeId` criteria.

Table C-35   (Cont.) Search Considerations and Recommendations

Consideration	Recommendation
Queries may be case-sensitive depending on the selected Oracle WebCenter Content search engine.	Be aware that the Oracle WebCenter Content search engine selection can affect case-sensitivity. Metadata searches using OracleTextSearch as the search engine are generally case-insensitive. Metadata searches using `DATABASE.FULLTEXT` as the search engine are generally case-sensitive. The exact behavior sometimes varies by metadata field.

C.15.1.2 Services

The methods described in *Section 2.2 Services* of the CMIS specification are implemented by the ; specific implementation details are covered in CMIS Part II: RESTful AtomPub Binding.

C.15.2 CMIS Part II: RESTful AtomPub Binding

Section 3: RESTful AtomPub Binding in the CMIS specification defines a specification based on AtomPub that can be used by applications to work with one or more Content Management Repositories. REST services are available through a portal instance.

C.15.2.1 Service Document

All navigation of a repository begins with the AtomPub Service Document. From this document, all accessible content in a repository can be discovered through the collections, links, and templates. The URI to the service document, relative to the CMIS web application's context-root, is /rest/cmis/repository.

Therefore, if an application is deployed with a library-context-root-override as in the example above, the service document would be accessed through the following URL: `http://hostname:port/rest/cmis/repository`

By default, this document will contain a workspace for each configured Oracle WebCenter Content repository (only Oracle WebCenter Content repositories are supported by CMIS REST in Oracle WebCenter Portal). A service document for a single repository can be obtained by using the `repositoryId` query parameter, as described in section 3.6.2.1 of the CMIS AtomPub binding specification.

As noted in the previous section, the service document consists of AtomPub workspaces. Each workspace maps to a Oracle WebCenter Portal Content Server connection.

Specific URIs beyond the service document are not published; it is assumed that users will start at the service document and navigate the collections and links down expected paths. The relationships of the links and the titles and types of the collections are all defined in the CMIS specification, and thus can be commonly navigated by a client implementation. There are also templates defined for each repository, for easier access of objects by path, object by ID, type by ID, and queries. The format of the variables for the path and ID templates can be discovered by viewing the Entries of Folders and Documents.

C.15.2.2 Response Formats

Section 3.1.3: Response Formats in the CMIS specification indicates that Atom/AtomPub style formats will be returned by default unless overridden by a supported media type expressed in the Accept header.

A generic AtomPub feed reader can walk through any of the feeds returned by the CMIS REST server. It will not see all the CMIS specifics, but will be able to navigate through links. In general, to set up a feed reader, you need to know the URI of a particular feed, which can be discovered by navigating through the service document, for example, the workspace link for "typesdescendants".

For details on query syntax, see the CMIS specification. For best practices and examples, see Best Practices and Examples.

C.15.2.3 Additional Functionality

The provides the following additional functionality beyond the CMIS specification.

- Folder Children Collection
- Document Entry
- Content Stream

C.15.2.3.1 Folder Children Collection

The specification defines the following CMIS services:

- GET: getChildren
- POST: createDocument or createFolder or createPolicy or moveObject or addObjectToFolder

The provides the following additional services:

- POST: create This service has five query parameters: uid, fileName, contentId, comments, simpleResponse, and one header parameter: Slug. This service is meant to be used as a simple binary request upload. A new document is created with this service. Slug and fileName are used to name the binary that is attached to the request (using both parameters is optional; only one needs to be defined and fileName is checked first). The comments parameter is optional and contentId is optional if Oracle WebCenter Content is set up to auto-generate the dDocName.

- POST: create Content-Type: multipart/form-data This service has a single query parameter: uid, which is the uid of the folder in which the document is to be created. The boolean query parameter simpleResponse will return a response of media type: application/atom+xml;type=entry, if set to false. If set to true, a response of media type: text/html will be returned with a URI pointing to the newly created document. The comments and simpleResponse parameters are both optional, contentId is optional if Oracle WebCenter Content is set up to auto-generate the dDocName, and the name "fileUpload" is required.

```
<html>
<head>
    <title>simple post</title>
</head>
<body>
<form action="http://<host>:<port>/rest/api/cmis/children/StellentRepository?
uid=IDC:Folder/2"
```

```
                method="POST"
                enctype="multipart/form-data">
           Select a document to upload: <input type="file" name="fileUpload"/><br>
            <input type="hidden" name="comments" value="this is just a comment"/>
            <input type="hidden" name="contentId" value="uniqueID1"/>
            <input type="hidden" name="simpleResponse" value="true"/>
           <input type="submit" value="Submit"/>
     </form>
     </body>
     </html>
```

C.15.2.3.2 Document Entry

The specification defines the following CMIS services:

- GET: getObject, getObjectOfLatestVersion (getObject)

- PUT: updateProperties

- DELETE: deleteObject

The provides the following additional services:

- POST: postToDelete This service has two query parameters: uid and _method, and allows a document to be deleted through POST.

  ```
  http://<host>:<port>/rest/api/cmis/document/repoName?uid=ABC&_method="delete"
  ```

C.15.2.3.3 Content Stream

The specification defines the following CMIS services:

- GET: getContentStream

- PUT: setContentStream

- DELETE: deleteContentStream

The provides the following additional services:

- POST: postTunnelContentStream This service has five query parameters: uid, overwriteFlag, fileName, comments, _method, and one header parameter: Slug. This service is meant to be used as a simple binary request upload or delete through POST. A document must already exist for this service. Slug and fileName are used to name the binary that is attached to the request (using both parameters is optional; only one needs to be defined and fileName is checked first). The overwriteFlag parameter defaults to true, the comments parameter is optional and valid values for _method are "delete" or "put" (not case sensitive).

  ```
  http://<host>:<port>/rest/api/cmis/stream/repoName?uid=ABC&_method="delete"
  ```

- POST: postTunnelContentStream Content-Type: multipart/form-data This service has a single query parameter: uid and is meant to be used as a simple html multipart/form-data upload or delete through POST. A document must already exist for this service. The attribute name="fileUpload" is required, "comments" is optional, and valid values for "_method" are "delete" or "put" (not case sensitive).

  ```
  <form action="http://<host>:<port>/rest/api/cmis/stream/repoName?uid=WDOC019113"
        method="POST"
        enctype="multipart/form-data">
      Select a document to upload: <input type="file" name="fileUpload"/><br>
       <input type="hidden" name="comments" value="this is just a comment"/>
  ```

```
      <input type="hidden" name="_method" value="PUT"/>
    <input type="submit" value="Submit"/>
</form>
```

C.15.3 Best Practices and Examples

This section provides best practices and examples using the . For details on query syntax, see the CMIS specification.

This section includes the following topics:

- Best Practices

- Examples

C.15.3.1 Best Practices

The following list provides suggested best practices for repositories that use the .

- To determine the types that can be used in the "FROM" portion of a query, see the types collection from the AtomPub service document. The type must be queryable and the query name of that type must be used.

  For example, IDC:GlobalProfile might have type information similar to the following:

  ```
  <cmis:localName>IDC:GlobalProfile</cmis:localName>
  <cmis:displayName>IDC:GlobalProfile</cmis:displayName>
  <cmis:queryName>ora:t:IDC!;GlobalProfile</cmis:queryName>
  <cmis:queryable>true</cmis:queryable>
  ```

  An example query for the type information above could be: "SELECT * FROM ora:t:IDC!;GlobalProfile".

- To determine the properties that can be used in the "SELECT" and "WHERE" portions of a query, see the entry for the associated type. Each property definition of that type will be listed and will have a setting for queryable and orderable. The cmis:queryname will be the value to be used in the query.

  For example, IDC:GlobalProfile might have property definition information similar to the following:

  ```
  <cmis:propertyStringDefinition>
              <cmis:id>/stanl18-ucm11g/IDC:GlobalProfile.ora:p:dDocName</cmis:id>
              <cmis:localName>dDocName</cmis:localName>
              <cmis:displayName>dDocName</cmis:displayName>
              <cmis:queryName>ora:p:dDocName</cmis:queryName>
              <cmis:description>Content ID</cmis:description>
              <cmis:propertyType>string</cmis:propertyType>
              <cmis:cardinality>single</cmis:cardinality>
              <cmis:updatability>readwrite</cmis:updatability>
              <cmis:inherited>false</cmis:inherited>
              <cmis:required>false</cmis:required>
              <cmis:queryable>true</cmis:queryable>
              <cmis:orderable>true</cmis:orderable>
          </cmis:propertyStringDefinition>
  ```

  An example query for the property definition information above could be: "SELECT ora:p:dDocName FROM ora:t:IDC!;GlobalProfile"

- To keep queries more readable, avoid non-alphanumeric characters in ContentType and PropertyDefinition names.

C.15.3.2 Examples

This section provides some example queries. For details on query syntax, see the CMIS specification. (To get the full URI for a query, see the query URI template in the service document.)

- SELECT * from cmis:folder

- SELECT cmis:name, cmis:contentStreamFileName, cmis:contentStreamMimeType, cmis:contentStreamLength FROM cmis:document WHERE cmis:contentStreamFileName = 'BinaryName' AND cmis:contentStreamMimeType = 'text/html' AND cmis:contentStreamLength > 1

- SELECT cmis:name, cmis:creationDate, cmis:lastModificationDate FROM cmis:folder WHERE cmis:name = 'Trash' AND cmis:lastModificationDate > TIMESTAMP '2008-05-18T10:32:44.703-06:00'

- SELECT * FROM cmis:document WHERE cmis:name LIKE 'baker%'

- SELECT * FROM cmis:document WHERE cmis:name NOT IN ('nodeBoolean', 'nodeLong')

- SELECT cmis:name from cmis:document where IN_TREE('/StellentRepository')

- SELECT * FROM ora:t:IDC:GlobalProfile WHERE ora:p:xBooleanTestField = FALSE ORDER BY ora:p:dDocTitle ASC

- SELECT ora:p:xMultiValuedDelimiterTest FROM ora:t:IDC:GlobalProfile WHERE ANY ora:p:xMultiValuedDelimiterTest NOT IN ('four')

- SELECT cmis:name FROM ora:t:IDC:GlobalProfile WHERE CONTAINS('test') ORDER BY ora:p:dInDate DESC

- SELECT * FROM cmis:document where IN_TREE('/StellentRepository/IDC:Folder/2')

C.16 Using the Events REST API

WebCenter Portal provides a REST API to support events. Use the REST API to create your own interface for providing access to portal events.

> **Note:**
>
> The Events REST API is available for portal events only. You cannot use the REST API to work with personal events.

This section describes the REST API methods associated with events. It includes the following topics:

- Events Entry Point

- Events Resource Type Taxonomy

- Security Considerations

- Events Resource Types

C.16.1 Events Entry Point

The entry point for events can be reached only through a portal. First you need to navigate to the appropriate portal and then find the link element with a `resourceType` of:

```
urn:oracle:webcenter:events:gsEvents
```

The corresponding `href` or `template` element provides the URI entry point. The client sends HTTP requests to this entry point to work with events.

For more information about the Resource Index, see Using the Resource Index.

For more information about resource types, see Resource Type.

C.16.2 Events Resource Type Taxonomy

When the client has identified the entry point, it can then navigate through the resource type taxonomy to perform the required operations. For more information about the individual resources types, see the appropriate section in Events Resource Types.

The taxonomy for events is:

```
urn:oracle:webcenter:events:gsEvents
  urn:oracle:webcenter:events:gsEvent
urn:oracle:webcenter:events:gsCategories
```

C.16.3 Security Considerations

There are no specific security considerations for events. For general security considerations, see Security Considerations for WebCenter Portal REST APIs.

C.16.4 Events Resource Types

This section provides you with all the information you need to know about each resource type. It includes the following topics:

- urn:oracle:webcenter:events:gsEvents

- urn:oracle:webcenter:events:gsEvent

- urn:oracle:webcenter:events:gsCategories

C.16.4.1 urn:oracle:webcenter:events:gsEvents

Use this resource to identify the URI to use to retrieve (`GET`) and create (`POST`) portal events. The response from a `GET` operation includes each portal event in this collection of events, and each event includes links used to operate on that event. The response from a `POST` operation includes the event that was created in this collection of events and a link to operate on that event.

C.16.4.1.1 Navigation Paths to gsEvents

This section shows how the client can navigate through the hypermedia to access this resource:

```
resourceIndex
   spaces
      gsEvents
```

C.16.4.1.2 Supported Methods for gsEvents

The following methods are supported by this resource:

- `GET`

  - **request - body:** <gsCategory>, **Parameters:** `startIndex`, `itemsPerPage`, `utoken`

    For information about these common parameters, see Common Request Query Parameters.

    The following additional parameters are available:

    ◆ `startDate`—the date from which to start listing portal events

    ◆ `endDate`—the date at which to stop listing portal events

    For the start and end dates, use the format `YYYY-MM-DD`, for example `2011-09-01`. You can also specify a time (for example, `2011-09-01T09:00:00`) and a timezone sign (for example, for UTC, `2011-09-01T09:00:00Z`).

  - **response - body:** 0 or more events

- `POST`

  - **request - body:** event

  - **response - body:** event

C.16.4.1.3 Resource Types Linked to From gsEvents

Table C-36 lists the resource types that the client can link to from this resource.

Table C-36   Related Resource Types for gsEvents

rel	resourceType
self	urn:oracle:webcenter:events:gsEvents

C.16.4.2 urn:oracle:webcenter:events:gsEvent

Use this resource type to identify the URI to use to read (`GET`), update (`PUT`), or delete (`DELETE`) a specific portal event. The response from a `GET` operation includes the specific event identified by the URI. The response from a `PUT` operation includes the modified version of the event identified by the URI. The response from a `DELETE` operation is a 204.

C.16.4.2.1 Navigation Paths to gsEvent

```
resourceIndex
   spaces
      gsEvents
         gsEvent
```

C.16.4.2.2 Supported Methods for gsEvent

The following methods are supported by this resource:

- GET
  - **request - body:** none
  - **response - body:** event

- PUT
  - **request - body:** event
  - **response - body:** event

- DELETE
  - **request - body:** none
  - **response - body:** none

C.16.4.2.3 Writable Elements for gsEvent

Table C-37 lists the writable elements for this resource.

Table C-37   Writable Elements for gsEvent

Element	Type	Required	Constraints	Description
category	String	No	The name of an event category defined for the portal	The category to which the event belongs. **Note:** If the category does not exist, the default is used if the event is being created. However, an error will occur if the event is being modified.
details	String	No	1 or more characters	Additional details about the event
endTime	Date	Yes	YYYY-MM-DDTHH:MM:SS endTime must be greater than or equal to startTime	The date and time at which the event ends
location	String	No	1 or more characters	The location at which the event takes place
priority	String?	No	1 - Highest 2 - High 3 - Normal 4 - Low 5 - Lowest	The priority of the event, which determines where it appears when events clash

Table C-37   (Cont.) Writable Elements for gsEvent

Element	Type	Required	Constraints	Description
startTime	Date	Yes	YYYY-MM-DDTHH:MM:SS	The date and time at which the event starts
summary	String	Yes	1 or more characters	A brief description of the event to serve as the title of the event

C.16.4.2.4 Read-only Elements for gsEvent

Table C-38 lists the read-only elements for this resource.

Table C-38   Read-only Elements for gsEvent

Element	Type	Description
author	personReference	User who created the event
created	Date	Date on which the event was created
duration	String	The length (in minutes) of the event
groupSpace	groupSpaceReference	The portal to which the event belongs
id	String	Unique ID of the event
isAllDayEvent	Boolean	Indicates whether the event takes place over an entire day
modified	Date	Date on which the event was last updated
modifiedByUser	personReference	User who last updated the event

C.16.4.2.5 Resource Types Linked to from gsEvent

Table C-39 lists the resource types that the client can link to from this resource.

Table C-39   Related Resource Types for gsEvent

rel	resourceType
self	urn:oracle:webcenter:events:gsEvent

C.16.4.3 urn:oracle:webcenter:events:gsCategories

Use this resource to identify the URI to use to retrieve (GET) and create (POST) portal event categories. The response from a GET operation includes each category in this collection of categories, and each category includes links used to operate on that category. The response from a POST operation includes the category that was created in this collection of categories and a link to operate on that category.

C.16.4.3.1 Navigation Paths to gsCategories

This section shows how the client can navigate through the hypermedia to access this resource:

```
resourceIndex
   spaces
      gsCategories
```

C.16.4.3.2 Supported Methods for gsCategories

The following methods are supported by this resource:

- GET
    - **request - body:** category
    - **response - body:** 0 or more categories

- POST
    - **request - body:** category
    - **response - body:** category

C.16.4.3.3 Resource Types Linked to From gsCategories

Table C-40 lists the resource types that the client can link to from this resource.

Table C-40   Related Resource Types for gsCategories

rel	resourceType
self	urn:oracle:webcenter:events:gsCategories

C.17 Using the Tags REST APIs

WebCenter Portal provides REST APIs to support tags. Use the tags REST APIs to do the following:

- Find all tags or find a subset of tags by criteria

- Get a specific tag

- Delete tags

- Update or rename tags

- Get tagged items

- Untag an item

- Add a new tagged item

- Change tags for a tagged item

- Find all tagged items or find tagged items by a filter, such as a date or a tag word

- Get related users by tag

This section describes the REST APIs associated with tags. It includes the following topics:

- Tags Entry Point

- Tags Resource Type Taxonomy

- Security Considerations

- Tags Resource Types

For an introduction to the REST APIs, see Introduction to the WebCenter Portal REST APIs.

C.17.1 Tags Entry Point

Each REST service has a link element within the Resource Index that provides the entry point for that service. To find the entry point for tags, find the link element with a `resourceType` of one of the following:

- `urn:oracle:webcenter:tagging:tags`

- `urn:oracle:webcenter:tagging:taggeditems`

- `urn:oracle:webcenter:tagging:users`

The corresponding `href` or `template` element provides the URI entry point. The client sends HTTP requests to this entry point to work with tags.

For more information about the Resource Index, see Using the Resource Index.

For more information about resource types, see Resource Type.

C.17.2 Tags Resource Type Taxonomy

When the client has identified the entry point, it then can navigate through the resource type taxonomy to perform the required operations. For more information about the individual resource types, see the appropriate section in Resource Type.

The taxonomy for tags is:

```
urn:oracle:webcenter:tagging:tags
    urn:oracle:webcenter:tagging:tag
urn:oracle:webcenter:tagging:taggedItems
    urn:oracle:webcenter:tagging:taggedItem
urn:oracle:webcenter:tagging:users
```

Under users, you can add the resource type for that object, such as `peopleprofile`.

Beyond the service entry points, URL templates allow clients to pass query parameters to customize their requests and control the form of returned data.

Collection resources in the tags resources support pagination (`itemsPerPage` and `startIndex` for tags and users and `itemsPerPage` for taggedItems). Other query parameters (`search` and `projection`) are not supported.

C.17.3 Security Considerations

Authentication is required before using REST API methods.

For general security considerations, see Security Considerations for WebCenter Portal REST APIs.

C.17.4 Tags Resource Types

This section provides information about each resource type. It includes the following topics:

- urn:oracle:webcenter:tagging:tags

- urn:oracle:webcenter:tagging:tag

- urn:oracle:webcenter:tagging:taggedItems

- urn:oracle:webcenter:tagging:taggedItem

- urn:oracle:webcenter:tagging:users

C.17.4.1 urn:oracle:webcenter:tagging:tags

Use this resource type to identify the URI to use to read (`GET`) tags. The response from a `GET` operation includes all tags, a specific tag, a subset of tags, or a tag related to a specific list of tags or users.

C.17.4.1.1 Navigation Paths to tags

This section shows how the client can navigate through the hypermedia to access this resource:

```
resourceindex
    tags

resourceindex
    spaces
        spaces:resourceindex
            spaces:tags
```

C.17.4.1.2 Supported Methods for tags

The following methods are supported by this resource:

- `GET`

  - **request - body:** none, **Parameters**: `startIndex`, `itemsPerPage`, `keyword`, `serviceId`, `relatedTag`, `userId`, `scopeGuid`, `shared`, `tags`, `users`

  - **response - body:** tag

  where:

  - `startIndex`—Specifies the index of the first matching result that should be included in the result set (0-*n* ... zero based). This is used for pagination.

  - `itemsPerPage`—Specifies the maximum number of results to return in the response (1-*n*). This is used for pagination.

  - `keyword`—Specifies a keyword for substring matches.

  - `serviceId`—Specifies the service ID for which you want to find tags (null for all).

  - `relatedTag`—Specifies a tag that is related to the list of tags. For example, if you have tagged item1 as "webcenter help" and item2 as "webcenter document"

and you set `relatedTag=webcenter`, then you would get both tags (for help and document) returned.

– `userId`—Specifies a single user ID to find tags for (null for all). Uses logged-in user if null and not looking for shared tags.

– `scopeGuid`—Specifies the GUID of the portal. When set to the Home portal, it shows all (unfiltered) tags.

– `shared`—Specify `true` to return personal resources only (that is, resources tagged privately only by that same user); `false` to return system resources (that is, resources with at least one public tag).

– `tags`—Specifies to filter by this list of tags separated by URL-encoded spaces ('+'); for example, an item tagged with "webcenter" and "help" could be searched with `tags=webcenter+help`.

– `users`—Specifies to filter by this list of users separated by URL-encoded spaces ('+'); for example, `users=monty+vicki+pat`.

> **Note:**
>
> Parameters can be used in certain combinations only.

Table C-41 shows the available parameter combinations for each operation. For example, the `tags` parameter cannot be specified with the `serviceId` parameter.

Table C-41   Parameter Combinations Available for Getting Tags

API Method	keyword	service Id	related Tag	userId	scope Guid	shared	tags	users
findRelatedSystemTags	-	-	X	-	X	-	X	X
findPopularTags Common	X	X	-	X	-	X	-	-
findPopularTags (with GUID)	X	-	-	X	X	X	-	-
findPopularTags	X	-	-	X	-	X	-	-

C.17.4.1.3 Resource Types Linked to From tags

The resourceType link attribute indicates the type of resource to which the link points. Clients should use the resourceType to determine the expected response bodies for `GET` and `POST` and allowable request bodies for `POST` and `PUT`.

Table C-42 lists the resource types that the client can link to from this resource. Tags have a reference link to themselves and to the next page or previous page of results.

Table C-42   Related Resource Types for tags

rel	resourceType
self	urn:oracle:webcenter:tagging:tags

Table C-42   (Cont.) Related Resource Types for tags

rel	resourceType
Previous/Next	urn:oracle:webcenter:tagging:tags

C.17.4.2 urn:oracle:webcenter:tagging:tag

Use this resource type to identify the URI to use to read (`GET`), rename (`PUT`), and delete (`DELETE`) a tag. The response from a `GET` operation includes all tags, a specific tag, a subset of tags, or a tag related to a specific list of tags or users.

C.17.4.2.1 Navigation Paths to tag

This section shows how the client can navigate through the hypermedia to access this resource:

```
resourceindex
   tags
      tag

resourceindex
   spaces
      spaces:resourceindex
         spaces:tags
            spaces:tag
```

C.17.4.2.2 Supported Methods for tag

The following methods are supported by this resource:

- `GET`

  - **request - body:** none, **Parameters**: none

  - **response - body:** tag

- `PUT`

  - **request - body:** tag

  - **response - body:** tag

- `DELETE`

  - **request - body:** none

  - **response - body:** none

Example: Body of the PUT command for Renaming a Tag

```
<tag>
<name>newId</name>
</tag>
```

C.17.4.2.3 Resource Types Linked to From tag

The resourceType link attribute indicates the type of resource to which the link points. Clients should use the resourceType to determine the expected response bodies for `GET` and `POST` and allowable request bodies for `POST` and `PUT`.

Table C-43 lists the resource types that the client can link to from this resource. A tag has a reference link to itself, to related taggedItems, to related users, and to related tags.

Table C-43   Related Resource Types for tag

rel	resourceType
self	urn:oracle:webcenter:tagging:tag
	urn:oracle:webcenter:tagging:taggedItems
	urn:oracle:webcenter:tagging:users
	urn:oracle:webcenter:tagging:tags

C.17.4.3 urn:oracle:webcenter:tagging:taggedItems

Use this resource type to identify the URI to use to read tagged items and their related resources (GET) or to add tagged items (POST).

The response from a GET operation includes a tagged item or all tagged items, which can be filtered by date, tag words, and so on. Each tag includes links used to operate on that tag. The response from a POST operation includes the tagged items that were created in this collection of tags and a link to operate on them.

C.17.4.3.1 Navigation Paths to taggedItems

This section shows how the client can navigate through the hypermedia to access this resource:

```
resourceindex
   taggedItems

resourceindex
   spaces
      spaces:resourceindex
         spaces:taggedItems
```

C.17.4.3.2 Supported Methods for taggedtems

The following methods are supported by this resource:

- POST

  – **request - body:** taggedItem

  – **response - body:** taggedItem

  ```
  <taggeditem>
     <serviceId>oracle.webcenter.page</serviceId>

  <resourceId>/oracle/webcenter/page/somepath/someotherpath/a.jspx</resourceId>
     <tagWords>foo bar boo far</tagWords>
     <resourceTitle>This is renamed by me to be foofoo</resourceTitle>
     <scopeGuid>(optional) scope for the item</scopeGuid>
     <shared>(optional) make the item private </shared>
   </taggeditem>
  ```

- GET

  – **request - body:** none, **Parameters**: serviceId, keyword, relatedRID, tags, users, dt, scopeGuid, shared, itemsPerPage

- **response - body:** tagged items

where:

- `serviceId`—Specifies the service ID of the item.

- `keyword`—Specifies a keyword in the resource title.

- `relatedRID`—Specifies a related tag. For example, if you tagged item1 as "webcenter help" and item2 as "fusionapps documents" and item3 as "webcenter documents" and set `relatedRID=item3`, then you would get back item1 and item2, because they share tags in common with item3.

- `tags`—Specifies to filter by this list of tags separated by URL-encoded spaces ('+'); for example, an item tagged with "webcenter" and "help" could be searched with `tags=webcenter+help`.

- `users`—Specifies to filter by these users, separated by URL-encoded spaces ('+'); for example, {USER}+{USER}+...+{USER}.

- `dt`—Specifies to filter bookmarks by this date, which defaults to the most recent date on which bookmarks were saved {DD-MM-CCYY}.

- `scopeGuid`—Specifies the GUID of the portal. When set to the Home portal, it shows all (unfiltered) tags.

- `shared`—Specify `true` to return personal resources only (that is, resources tagged privately only by that same user); `false` to return system resources (that is, resources with at least one public tag).

- `itemsPerPage`—Specifies the maximum number of results to return in the response (1-*n*). This is used for pagination.

Note:

Parameters can be used in certain combinations only.

Table C-44 shows the available parameter combinations for each operation. For example, the `tags` parameter cannot be specified with the `serviceId` parameter.

Table C-44   Parameter Combinations Available for Getting Tagged Items

API Method	service Id	keywor d	related RID	tags	users	dt	scope Guid	shared
findRelatedSyste mResources	X	-	X	-	-	-	-	-
findUpdatedRes ources	-	-	-	-	-	X	X	-
findSystemReso urces		-	-	X	X	-	X	X
findFilterPerson alResources	X	X	-	-	-	-	X	-

Table C-44   (Cont.) Parameter Combinations Available for Getting Tagged Items

API Method	service Id	keyword	related RID	tags	users	dt	scope Guid	shared
findPersonalResources	-	-	-	X	X	-	X	X
findFilteredPersonalResources (no keyword)	X	-	-	-	-	-	X	-

C.17.4.3.3 Resource Types Linked to From taggedItems

The resourceType link attribute indicates the type of resource to which the link points. Clients should use the resourceType to determine the expected response bodies for GET and POST and allowable request bodies for POST and PUT.

Table C-45 lists the resource types that the client can link to from this resource. Tagged items have a reference link to themselves and to the next page or previous page of results.

Table C-45   Related Resource Types for taggedItems

rel	resourceType
self	urn:oracle:webcenter:tagging:taggedItems
Previous/Next	urn:oracle:webcenter:tagging:taggedItems

C.17.4.4 urn:oracle:webcenter:tagging:taggedItem

Use this resource type to identify the URI to use to read tagged items and their related resources (GET), add tagged items (POST), update tags for a tagged item (PUT), and remove all tags from an item (DELETE).

The response from a GET operation includes a tagged item or all tagged items, which can be filtered by date, tag words, and so on. Each tag includes links used to operate on that tag. The response from a POST operation includes the tagged item that was created in this collection of tags and a link to operate on that tagged item.

Navigation Paths to taggedItem

This section shows how the client can navigate through the hypermedia to access this resource:

```
resourceindex
   taggedItems
      taggedItem

resourceindex
   spaces
      spaces:resourceindex
         spaces:taggedItems
            spaces:taggedItem
```

C.17.4.4.1 Supported Methods for taggedItem

The following methods are supported by this resource:

- GET

  - **request - body:** none, **Parameters**: `serviceId`, `keyword`, `tags`, `users`, `dt`, `shared`, `itemsPerPage`

  - **response - body:** tagged item

  where:

  - `serviceId`—Specifies the service ID of the item.

  - `keyword`—Specifies a keyword in the resource title.

  - `tags`—Specifies to filter by this list of tags separated by URL-encoded spaces ('+'); for example, an item tagged with "webcenter" and "help" could be searched with `tags=webcenter+help`.

  - `users`—Specifies to filter by this list of users separated by URL-encoded spaces ('+'); for example, `users=monty+vicki+pat`.

  - `dt`—Specifies to filter bookmarks by this date, which defaults to the most recent date on which bookmarks were saved {DD-MM-CCYY}.

  - `shared`—Specify `true` to return personal resources only (that is, resources tagged privately only by that same user); `false` to return system resources (that is, resources with at least one public tag).

  - `itemsPerPage`—Specifies the maximum number of results to return in the response (1-*n*). This is used for pagination.

- PUT

  - **request - body:** taggedItem

  - **response - body:** taggedItem

  ```
  <taggeditem>
  <tagWords>foo bar boo far</tagWords>
  </taggeditem>
  ```

- DELETE

  - **request - body:** none

  - **response - body:** none

C.17.4.4.2 Resource Types Linked to From taggedItem

The resourceType link attribute indicates the type of resource to which the link points. Clients should use the resourceType to determine the expected response bodies for GET and POST and allowable request bodies for POST and PUT.

Table C-46 lists the resource types that the client can link to from this resource. A tagged item has a reference link to itself, related tagged items, and an external link.

Table C-46   Related Resource Types for taggedItem

rel	resourceType
self	urn:oracle:webcenter:tagging:taggedItem
	urn:oracle:webcenter:tagging:taggedItems

Table C-46 (Cont.) Related Resource Types for taggedItem

rel	resourceType
alternate	The resource type depends on the type of resource tagged; for example, a page would be urn:oracle:webcenter:page:page

C.17.4.5 urn:oracle:webcenter:tagging:users

Use this resource type to get a list of people related to a given list of tags.

C.17.4.5.1 Navigation Paths to users

This section shows how the client can navigate through the hypermedia to access this resource:

```
resourceindex
   taggingusers

resourceindex
   spaces
      spaces:resourceindex
         spaces:taggingusers
```

C.17.4.5.2 Supported Methods for users

The following methods are supported by this resource:

- `GET`

    - **request - body:** none, **Parameters**: `startIndex`, `itemsPerPage`, `tags`, `users`

    - **response - body:** people profile

    where:

    - `startIndex`—Specifies the index of the first matching result that should be included in the result set (0-*n* ... zero based). This is used for pagination.

    - `itemsPerPage`—Specifies the maximum number of results to return in the response (1-*n*). This is used for pagination.

    - `tags`—Specifies to filter by this list of tags separated by URL-encoded spaces ('+'); for example, an item tagged with "webcenter" and "help" could be searched with `tags=webcenter+help`.

    - `users`—Specifies to filter by this list of users separated by URL-encoded spaces ('+'); for example, `users=monty+vicki+pat`.

C.17.4.5.3 Resource Types Linked to From users

The resourceType link attribute indicates the type of resource to which the link points. Clients should use the resourceType to determine the expected response bodies for `GET` and `POST` and allowable request bodies for `POST` and `PUT`.

Table C-47 lists the resource types that individual users can link to from this resource. Users have a reference link to themselves and to the next page or previous page of results.

Table C-47   Related Resource Types for users

rel	resourceType
self	urn:oracle:webcenter:tagging:users
Previous/Next	urn:oracle:webcenter:tagging:users

C.18 Using the Discussions REST API

WebCenter Portal provides a REST API to support discussions. Use the discussions REST API to post, read, update, and delete discussion forums, topics, and messages.

This section describes the REST API methods associated with discussions. It includes the following topics:

- Discussions Entry Point

- Discussions Resource Type Taxonomy

- Security Considerations

- Discussions Resource Types

C.18.1 Discussions Entry Point

Each REST service has a link element within the Resource Index that provides the entry point for that service. To find the entry point for discussions, find the link element with a `resourceType` of:

`urn:oracle:webcenter:discussions:forums`

The corresponding `href` or `template` element provides the URI entry point. The client sends HTTP requests to this entry point to work with discussions.

For more information about the Resource Index, see Using the Resource Index.

For more information about resource types, see Resource Type.

C.18.2 Discussions Resource Type Taxonomy

When the client has identified the entry point, it can then navigate through the resource type taxonomy to perform the required operations. For more information about the individual resource types, see the appropriate section in Resource Type.

The taxonomy for discussions is:

```
urn:oracle:webcenter:discussions:forums
   urn:oracle:webcenter:discussions:forum
   urn:oracle:webcenter:discussions:forum:topics
      urn:oracle:webcenter:discussions:forum:topic
      urn:oracle:webcenter:discussions:forum:topic:messages
         urn:oracle:webcenter:discussions:forum:topic:message
```

Beyond the service entry points, URL templates allow clients to pass query parameters to customize their requests and control the form of returned data.

Collection resources in the discussions resources support pagination (`startIndex` and `itemsPerPage`). Other query parameters are not supported (that is, `search` and `projection`).

C.18.3 Security Considerations

There are no specific security considerations for discussions. For general security considerations, see Introduction to the WebCenter Portal REST APIs.

C.18.4 Discussions Resource Types

This section provides the information about each resource type. It includes the following topics:

- urn:oracle:webcenter:discussions:forums

- urn:oracle:webcenter:discussions:forum

- urn:oracle:webcenter:discussions:forum:topics

- urn:oracle:webcenter:discussions:forum:topic

- urn:oracle:webcenter:discussions:forum:topic:messages

- urn:oracle:webcenter:discussions:forum:topic:message

C.18.4.1 urn:oracle:webcenter:discussions:forums

Use this resource type to identify the URI to use to read (GET) and write (POST) discussion forums. The response from a GET operation includes each forum in this collection of forums, and each forum includes links used to operate on that forum. The response from a POST operation includes the forum that was created in this collection of forums and a link to operate on that forum.

C.18.4.1.1 Navigation Paths to forums

This section shows how the client can navigate through the hypermedia to access this resource:

```
resourceindex
   forums

resourceindex
   spaces
      spaces:resourceindex
         spaces:forums
```

C.18.4.1.2 Supported Methods for forums

The following methods are supported by this resource:

- GET

  - **request - body:** none, **Parameters**: startIndex, itemsPerPage (pagination)

  - **response - body:** forums

- POST

  - **request - body:** forum

  - **response - body:** forum

Note that the number of forums displayed per page defaults to 10. For more information, see Templates.

C.18.4.1.3 Resource Types Linked to From forums

Table C-48 lists the resource types that the client can link to from this resource.

Table C-48   Related Resource Types for forums

rel	resourceType
self	urn:oracle:webcenter:discussions:forums
	urn:oracle:webcenter:discussions:forum

C.18.4.2 urn:oracle:webcenter:discussions:forum

Use this resource type to identify the URI to use to read (GET), update (PUT), and delete (DELETE) a specific discussion forum. The response from a GET operation includes the specific forum identified by the URI. The response from a PUT operation includes the modified version of the forum identified by the URI. The response from a DELETE operation is a 204.

C.18.4.2.1 Navigation Paths to forum

This section shows how the client can navigate through the hypermedia to access this resource:

```
resourceindex
    forums
        forum

resourceindex
    spaces
        spaces:resourceindex
            spaces:forums
                forum

resourceindex
    activities
        forum
```

C.18.4.2.2 Supported Methods for forum

The following methods are supported by this resource:

- GET

    - **request - body:** none

    - **response - body:** forum

- PUT

    - **request - body:** forum

    - **response - body:** forum

- DELETE

    - **request - body:** none

– **response - body:** none

C.18.4.2.3 Writable Elements for forum

Table C-49 lists the writable elements for this resource.

Table C-49   Writable Elements for forum

Element	Type	Required	Constraints	Description
name	String	Yes	1 or more characters	Name of the forum
displayName	String	No	1 or more characters	Name used in presentation
description	String	No	1 or more characters	Description of the forum

C.18.4.2.4 Read-only Elements for forum

Table C-50 lists the read-only elements for this resource.

Table C-50   Read-only Elements for forum

Element	Type	Description
id	Integer	ID of the forum
parentId	Integer	ID of the parent category
createdBy	String	ID of the user that created the forum
author	personReference	User information about the user that created the forum, including GUID, ID, display name, and a link to the profile icon (same user as createdBy)
createdOn	Date	Date on which the forum was created
updatedBy	String	ID of the user that performed the last modification
modifiedBy	personReference	User information about the user that performed the last modification, including GUID, ID, display name, and a link to the profile icon (same user as updatedBy)
updatedOn	Date	Date on which the forum was last modified
webUrl	String	URL for direct access to the discussions server
topicCount	Integer	Number of topics
messageCount	Integer	Number of messages
locked	Boolean	True if this forum is locked

Table C-50   (Cont.) Read-only Elements for forum

Element	Type	Description
favorite	Boolean	True if this forum is marked as a favorite

C.18.4.2.5 Resource Types Linked to From forum

Table C-51 lists the resource types that the client can link to from this resource.

Table C-51   Related Resource Types for forum

rel	resourceType
self	urn:oracle:webcenter:discussions:forum
	urn:oracle:webcenter:discussions:forum:topics

C.18.4.3 urn:oracle:webcenter:discussions:forum:topics

Use this resource type to identify the URI to use to read (GET) and write (POST) discussion topics. The response from a GET operation includes each topic in this collection of topics, and each topic includes links used to operate on that topic. The response from a POST operation includes the topic that was created in this collection of topics and a link to operate on that topic.

C.18.4.3.1 Navigation Paths to topics

This section shows how the client can navigate through the hypermedia to access this resource:

```
resourceindex
   forums
      topics

resourceindex
   spaces
      spaces:resourceindex
         spaces:forums
            topics
```

C.18.4.3.2 Supported Methods for topics

The following methods are supported by this resource:

- GET

  - **request - body:** none, **Parameters:** startIndex, itemsPerPage (pagination)

  - **response - body:** topics

- POST

  - **request - body:** topic

  - **response - body:** topic

For more information, see Templates.

C.18.4.3.3 Resource Types Linked to From topics

Table C-52 lists the resource types that the client can link to from this resource.

Table C-52   Related Resource Types for topics

rel	resourceType
self	urn:oracle:webcenter:discussions:forum:topics
	urn:oracle:webcenter:discussions:forum:topic

C.18.4.4 urn:oracle:webcenter:discussions:forum:topic

Use this resource type to identify the URI to use to read (GET), update (PUT), and delete (DELETE) a specific discussion topic. The response from a GET operation includes the specific topic identified by the URI. The response from a PUT operation includes the modified version of the topic identified by the URI. The response from a DELETE operation is a 204.

C.18.4.4.1 Navigation Paths to topic

This section shows how the client can navigate through the hypermedia to access this resource:

```
resourceindex
   forums
      topics
         topic

resourceindex
   spaces
      spaces:resourceindex
         spaces:forums
            topics
               topic

resourceindex
   activities
      topicT
```

C.18.4.4.2 Supported Methods for topic

The following methods are supported by this resource type:

- GET

    - **Request—Body:** none

    - **Response—Body:** topic

- PUT

    - **Request—Body:** topic

    - **Response—Body:** topic

- DELETE

    - **Request—Body:** none

– **Response—Body:** none

C.18.4.4.3 Writable Elements for topic

Table C-53 lists the writable elements for this resource type.

Table C-53 Writable Elements for topic

Element	Type	Required	Constraints	Description
subject	String	Yes	1 or more characters	Subject of the topic
body	String	No	0 or more characters	Contents of the topic

C.18.4.4.4 Read-only Elements for topic

Table C-54 lists the read-only elements for this resource type.

Table C-54 Read-only Elements for topic

Element	Type	Description
id	Integer	Identifier for the topic
parentId	Integer	Identifier for the parent message
forumId	Integer	Identifier for the forum under which this topic is posted
topicId	Integer	Identifier for the topic under which this topic is posted
createdBy	String	ID of the user who created the topic
author	personReference	User information about the user who created the topic, including GUID, ID, display name, and a link to the profile icon (same user as createdBy)
createdOn	Date	Date on which the topic was created
updatedBy	String	User who lasted updated the topic
updatedOn	Date	Date on which the topic was last updated
webUrl	String	URL for direct access to the discussions server
depth	Integer	Depth in the hierarchy of messages
messageCount	Integer	Number of child messages under this topic
numberOfReplies	Integer	Number of replies to this topic
favorite	Boolean	Whether the topic is marked as a favorite for the user

Table C-54 (Cont.) Read-only Elements for topic

Element	Type	Description
locked	Boolean	Whether the topic is locked
hidden	Boolean	Whether the topic is hidden
hasAttachment	Boolean	Whether the topic has attachments

C.18.4.4.5 Resource Types Linked to From topic

Table C-55 lists the resource types that the client can link to from this resource.

Table C-55 Related Resource Types for topic

rel	resourceType
self	urn:oracle:webcenter:discussions:forum:topic
	urn:oracle:webcenter:discussions:forum:topic:messages

C.18.4.5 urn:oracle:webcenter:discussions:forum:topic:messages

Use this resource type to identify the URI to use to read (GET) and write (POST) discussion topic messages. The response from a GET operation includes each message in this collection of messages, and each message includes links used to operate on that message. The response from a POST operation includes the message that was created in this collection of messages and a link to operate on that message.

C.18.4.5.1 Navigation Paths to messages

This section shows how the client can navigate through the hypermedia to access this resource:

```
resourceindex
   forums
      topics
         messages

resourceindex
   spaces
      spaces:resourceindex
         spaces:forums
            topics
               messages
```

C.18.4.5.2 Supported Methods for messages

The following methods are supported by this resource:

- GET

  – **request - body:** none, **Parameters:** startIndex, itemsPerPage (pagination)

  – **response - body:** messages

- POST

  – **request - body:** message

    – **response - body:** message

For more information, see Templates.

C.18.4.5.3 Resource Types Linked to From messages

Table C-56 lists the resource types that the client can link to from this resource.

Table C-56   Related Resource Types for messages

rel	resourceType
self	urn:oracle:webcenter:discussions:forum:topic:messages
	urn:oracle:webcenter:discussions:forum:topic:message

C.18.4.6 urn:oracle:webcenter:discussions:forum:topic:message

Use this resource type to identify the URI to use to read (GET), update (PUT), and delete (DELETE) a specific discussion topic message. The response from a GET operation includes the specific message identified by the URI. The response from a PUT operation includes the modified version of the message identified by the URI. The response from a DELETE operation is a 204.

C.18.4.6.1 Navigation Paths to message

This section shows how the client can navigate through the hypermedia to access this resource:

```
resourceindex
   forums
      topics
         messages
            message


resourceindex
   spaces
      spaces:resourceindex
         spaces:forums
            topics
               messages
                  message
```

C.18.4.6.2 Supported Methods for message

The following methods are supported by this resource:

- GET

    – **request - body:** none

    – **response - body:** message

- PUT

    – **request - body:** message

    – **response - body:** message

- DELETE

     – **request - body:** none

     – **response - body:** none

C.18.4.6.3 Writable Elements for message

Table C-57 lists the writable elements for this resource.

Table C-57   Writable Elements for message

Element	Type	Required	Constraints	Description
subject	String	Yes	1 or more characters	Subject of this message
body	String	No	0 or more characters	Content of this message

C.18.4.6.4 Read-only Elements for message

Table C-58 lists the read-only elements for this resource.

Table C-58   Read-only Elements for message

Element	Type	Description
id	Integer	ID of the message
parentId	Integer	ID of the parent message
forumId	Integer	ID of the forum under which the message is posted
topicId	Integer	ID of the topic under which the message is posted
createdBy	String	ID of user that created the message
author	personReference	User information about the user that created the message, including GUID, ID, display name and a link to the profile icon (same user as createdBy)
createdOn	Date	Date on which the message was created
updatedBy	String	User that performed the last modification
updatedOn	Date	Date on which the message was last modified
webUrl	String	URL for direct access to the discussions server
depth	Integer	Depth in the hierarchy of messages
messageCount	Integer	Number of messages
numberOfReplies	Integer	Number of replies to this message
hidden	Boolean	True if this message is hidden

Table C-58   (Cont.) Read-only Elements for message

Element	Type	Description
hasAttachment	Boolean	True if this message has attachments

C.18.4.6.5 Resource Types Linked to From message

Table C-59 lists the resource types that the client can link to from this resource.

Table C-59   Related Resource Types for message

rel	resourceType
self	urn:oracle:webcenter:discussions:forum:topic:message
	urn:oracle:webcenter:discussions:forum:topic:messages

C.19 Using the Lists REST API

Oracle WebCenter Portal provides a REST API to allow access to Lists functionality through interfaces other than the provided task flows. You can use the Lists REST API to perform the following actions:

- Work with lists in a portal, such as retrieving a list of lists or adding a new list

- Work with one list, such as retrieving or updating list metadata, or deleting the list

- Work with rows in a list

- Work with a list row, such as retrieving, updating, or deleting the row

- Work with list columns, such as adding a column

- Work with a list column, such as retrieving, updating, or deleting the column

This section describes the REST API methods associated with lists. It includes the following topics:

- Entry Point for Lists

- Lists Resource Type Taxonomy

- Lists Security Considerations

- Lists Resource Types

> **See Also:**
>
> For an introduction to the REST APIs, see Introduction to the WebCenter Portal REST APIs.

C.19.1 Entry Point for Lists

To get to the REST entry point for lists in a portal in WebCenter Portal, you search for the specific portal and retrieve the resource index. The corresponding href or

`template` element provides the URI entry point, which retrieves the lists in the portal.

1. Navigate to the portal for which to retrieve the lists.

2. Retrieve the resource index:

   `urn:oracle:webcenter:resourceindex`

3. Navigate to the returned `href`.

4. Search for the resource type. For example, for all the lists in a portal, search for:

   `urn:oracle:webcenter:space:lists`

See Also:

For more information about the Resource Index, see Using the Resource Index

For more information about resource types, see Resource Type

C.19.2 Lists Resource Type Taxonomy

When the client has identified the entry point, it can then navigate through the resource type taxonomy to perform the required operations. The resource type taxonomy for lists is:

```
urn:oracle:webcenter:space:lists
urn:oracle:webcenter:space:list
     urn:oracle:webcenter:space:list:rows
     urn:oracle:webcenter:space:list:row
     urn:oracle:webcenter:space:list:columns
     urn:oracle:webcenter:space:list:column
```

C.19.3 Lists Security Considerations

You must be logged in to the REST service to access any of the Lists REST API methods. After that, the underlying service handles permission checking.

See Also:

For general security considerations, see Security Considerations for WebCenter Portal REST APIs.

C.19.4 Lists Resource Types

This section provides you with all the information you need to know about each resource type. It includes the following topics:

- urn:oracle:webcenter:space:lists

- urn:oracle:webcenter:space:list

- urn:oracle:webcenter:space:list:rows

- urn:oracle:webcenter:space:list:row

- urn:oracle:webcenter:space:list:columns

- urn:oracle:webcenter:space:list:column

C.19.4.1 urn:oracle:webcenter:space:lists

The `lists` response provides a means of retrieving the lists in a given portal (`GET`) and adding lists to a portal (`POST`). This section includes the following topics:

- Navigation Paths to lists

- Supported Methods for lists

- Method (lists): `GET`

- Method (lists): `POST`

- Resource Types Linked to from lists

C.19.4.1.1 Navigation Paths to lists

This section shows how the client can navigate through the hypermedia to access the `lists` resource:

```
urn:oracle:webcenter:resourceindex
    urn:oracle:webcenter:spaces
        urn:oracle:webcenter:space:resourceindex
            urn:oracle:webcenter:space:lists
```

C.19.4.1.2 Supported Methods for lists

The `lists` resource type supports the following methods:

- Method (lists): `GET`

- Method (lists): `POST`

C.19.4.1.2.1 Method (lists): `GET`

This method retrieves the lists in a portal.

- Resource URI, for example:

  ```
  http://host:port/rest/api/spaces/vs1/lists?
  q=name:equals:ProjectIssues&utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**
  ```

- Request body: none

- Request headers: [Accept = `application/xml` | `application/json`]

- Request parameters - `startIndex`, `itemsPerPage`, `q`, `projection`

  - `q` parameter for query, format `'q=attribute:operator:value'`, for example:

    ```
    'q=name:equals:ProjectIssues'
    ```

    - Supported operators for Strings: `equals`, `not.equals`, `contains`, `starts.with`

    - Supported operators for Dates: `equals`, `not.equals`, `greater.than`, `greater.than.or.equals`, `less.than`, `less.than.or.equals`

    - May specify several conditions separated by semicolon (;)

- Searchable attributes (Table C-60)

Table C-60   Searchable Attributes for lists GET Method

Element	Type	Description
name	string	Name of list
description	string	Description of list
creator	string	User who created list
created	date	Date the list was created
modifier	string	User who modified list
modified	date	Date the list was last modified

- Response status: 200 [OK]

- Response body: `<lists>`

- Example:

```
<lists resourceType="urn:oracle:webcenter:space:lists">
      <links>
            <link resourceType="urn:oracle:webcenter:list:lists"
                  rel="self" href="opaque"/>
            <link template="opaque?startIndex={startIndex}&
                  itemsPerPage={itemsPerPage}&q={searchTerms}&
                  projection={projection}
                  &stoken=FDgsOu7a2NTTM1cLJvnpkDXfihtHx5Q*"
resourceType="urn:oracle:webcenter:list:lists"/>
      </links>
      <items>
            <list resourceType="urn:oracle:webcenter:space:list">
                  <links>
                        <link resourceType="urn:oracle:webcenter:list" rel="self"
                              href="opaque"
                              capabilities="urn:oracle:webcenter:update
                              urn:oracle:webcenter:delete"/>
                        <link resourceType="urn:oracle:webcenter:list:rows"
                              href="opaque"
                              capabilities="urn:oracle:webcenter:create"/>
                        <link template="opaque?
                              startIndex={startIndex}&
                              itemsPerPage={itemsPerPage}&q={searchTerms}&
                              stoken=FDgsOu7a2NTTM1cLJvnpkDXfihtHx5Q*"
                              resourceType="urn:oracle:webcenter:list:rows"/>
                        <link resourceType="urn:oracle:webcenter:list:columns"
                              href="opaque"
                              capabilities="urn:oracle:webcenter:create"/>
                  </links>
                  <id>/oracle/webcenter/list/scopedMD/
                        se0dea180_e2c1_45ac_b08b_
                        ba2c0b26aa72/lists/ProjectIssues.xml</id>
                  <name>Project Issues</name>
                  <description/>
            </list>
      </items>
</lists>
```

C.19.4.1.2.2 Method (lists): *POST*

This method creates a list in a portal.

- Resource URI, for example:

  http://*host*:*port*/rest/api/spaces/vs1/lists?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**

- Request body: `<list>`

- Writable elements ([Table C-61](#))

Table C-61   Create List Writable Elements

Element	Type	Constraints	Description
name[1]	string	—	Name of list
description	string	—	Description of list
columns[2]	urn:oracle:webcenter:space:list:columns	1 to 30	List columns

[1] Denotes required element

[2] Denotes required element

- Example:

```xml
<list>
      <name>RestExample</name>
      <description>Created using rest api</description>
      <columns>
            <items>
                  <metaColumn>
                        <name>No.</name>
                        <dataType>number</dataType>
                  </metaColumn>
                  <metaColumn>
                        <name>Description</name>
                        <dataType>string</dataType>
                  </metaColumn>
            </items>
      </columns>
</list>
```

- Request headers: `Content-Type = application/xml | application/json`, [Accept = `application/xml | application/json`]

- Response status: `201` [Created]

- Response body: `<list>`

- Retrieved elements ([Table C-62](#))

Table C-62   Retrieved Elements from lists

Element	Type	Description
id	string	List ID
name	string	List name

Table C-62   (Cont.) Retrieved Elements from lists

Element	Type	Description
description	string	List description
scope.guid	string	GUID of portal to which the list belongs
scope.name	string	Name of the portal to which the list belongs
creator	string	Name of the user who created the list
created	date	Date the list was created
modifier	string	Name of the user who last modified the list
modified	date	Date the list was last modified
columns	urn:oracle:webcenter:space:list:columns	List columns

- Example:

```
<list resourceType="urn:oracle:webcenter:space:list">
    <links>
        <link resourceType="urn:oracle:webcenter:space:list" rel="self"
            href="http://host:port/rest/api/spaces/vs1/
            lists/(/oracle/webcenter/list/scopedMD/s355923f0_2f04_4fd0_
            83ad_f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
            fb03874979c0.xml)?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
            capabilities="urn:oracle:webcenter:read
            urn:oracle:webcenter:update urn:oracle:webcenter:delete"/>
        <link template="http://host:port
            /rest/api/spaces/vs1/lists/(/oracle/webcenter/list/scopedMD/
            s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/ls_c9cecbc7_756b_
            417b_a35f_fb03874979c0.xml)/rows?
            utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_
            w**&startIndex={startIndex}&itemsPerPage={itemsPerPage}
            &q={searchTerms}&utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
            resourceType="urn:oracle:webcenter:space:list:rows"
            href="http://host:port/rest/api/spaces/
            vs1/lists/(/oracle/webcenter/list/scopedMD/s355923f0_2f04_4fd0_
            83ad_f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
            fb03874979c0.xml)/rows?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
            capabilities="urn:oracle:webcenter:read
            urn:oracle:webcenter:create"/>
        <link resourceType="urn:oracle:webcenter:space:list:columns"
            href="http://host:port/rest/api/spaces/
            vs1/lists/(/oracle/webcenter/list/scopedMD/s355923f0_2f04_4fd0_
            83ad_f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
            fb03874979c0.xml)/columns?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_
            w**" capabilities="urn:oracle:webcenter:read
            urn:oracle:webcenter:create"/>
    </links>
    <id>/oracle/webcenter/list/scopedMD/s355923f0_2f04_4fd0_83ad_
        f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_fb03874979c0.xml</id>
    <name>RestExample</name>
    <description>Created using rest api</description>
```

```
                         <scope>
                                <guid>s355923f0_2f04_4fd0_83ad_f7dac2a7ceed</guid>
                                <name>vs1</name>
                         </scope>
                         <creator>weblogic</creator>
                         <author>
                                <links>
                                        <link resourceType="urn:oracle:webcenter:people:person"
                                                rel="via" href="http://host:port/
                                                rest/api/people/E9A35E80F0BC11DFBFBE0FE339E55B21/lists/
                                                @self?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
                                                capabilities="urn:oracle:webcenter:read"/>
                                        <link type="image/png"
                                                template="http://host:port/
                                                webcenter/profilephoto/45394133354538304630424
                                                33131444642464245304645333339453535423231/{size}?
                                                _xResourceMethod=wsrp"
                                                resourceType="urn:oracle:webcenter:people:person"
                                                rel="urn:oracle:webcenter:people:icon"
                                                href="http://host:port/webcenter/
                                                profilephoto/
45394133354538304630424331314446424642453046 45
                                                333339453535423231/SMALL?_xResourceMethod=wsrp"
                                                capabilities="urn:oracle:webcenter:read"/>
                                        <link type="text/html"
                                                resourceType="urn:oracle:webcenter:spaces:profile"
                                                rel="alternate" href="http://host:port/
                                                webcenter/faces/oracle/webcenter/webcenterapp/view/pages/
                                                peopleconn/UserProfileGallery.jspx?wc.username=weblogic"
                                                capabilities="urn:oracle:webcenter:read"/>
                                </links>
                                <displayName>weblogic</displayName>
                                <guid>E9A35E80F0BC11DFBFBE0FE339E55B21</guid>
                                <id>weblogic</id>
                         </author>
                         <created>2010-11-18T06:10:32.250-08:00</created>
                         <modifier>weblogic</modifier>
                         <modifiedByUser>
                                <displayName>weblogic</displayName>
                                <guid>E9A35E80F0BC11DFBFBE0FE339E55B21</guid>
                                <id>weblogic</id>
                         </modifiedByUser>
                         <modified>2010-11-18T06:10:32.250-08:00</modified>
                         <columns resourceType="urn:oracle:webcenter:space:list:columns">
                                <links>
                                        <link resourceType="urn:oracle:webcenter:space:list:columns"
                                                rel="self"
                                                href="http://host:port/rest/api/spaces
                                                /vs1/lists/(/oracle/webcenter/list/scopedMD/
                                                s355923f0_2f04_4fd0_83ad_
                                                f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
                                                fb03874979c0.xml)/columns?utoken=
                                                FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
                                                capabilities="urn:oracle:webcenter:read
                                                urn:oracle:webcenter:create"/>
                                        <link resourceType="urn:oracle:webcenter:space:list"
                                                rel="urn:oracle:webcenter:parent"
                                                href="http://host:port/
                                                rest/api/spaces/vs1/lists/(/oracle/webcenter/list/
                                scopedMD/
                                                s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/
```

```
                                    ls_c9cecbc7_756b_417b_a35f_fb03874979c0.xml)?
                                    utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"/>
                            </links>
                            <items>
                                    <metaColumn resourceType=
                                            "urn:oracle:webcenter:space:list:column">
                                            <links>
                                                    <link
resourceType="urn:oracle:webcenter:space:
                                                            list:column" rel="self
                                                            "href="http://host:port/
                                                            rest/api/spaces/vs1/lists/
                                                            (/oracle/webcenter/list/scopedMD/
                                                            s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/
                                                            lists/ls_c9cecbc7_756b_417b_a35f_
                                                            fb03874979c0.xml)/columns/

(lco_9bdd1418_6004_40ba_a052_04e8335b7ee8)?

utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
                                                            capabilities="urn:oracle:webcenter:read
                                                            urn:oracle:webcenter:update
                                                            urn:oracle:webcenter:delete"/>
                                            </links>
                                            <id>lco_9bdd1418_6004_40ba_a052_04e8335b7ee8</id>
                                            <name>No.</name>
                                            <dataType>number</dataType>
                                            <required>false</required>
                                            <displayLength>10</displayLength>
                                            <format>number</format>
                                            <allowLinks>false</allowLinks>
                                    </metaColumn>
                                    <metaColumn resourceType=
                                            "urn:oracle:webcenter:space:list:column">
                                            <links>
                                                    <link resourceType=

"urn:oracle:webcenter:space:list:column"
                                                            rel="self"
                                                            href="http://host:port/
                                                            rest/api/spaces/vs1/lists/(/oracle/
                                                            webcenter/list/scopedMD/
                                                            s355923f0_2f04_4fd0_83ad_
                                                            f7dac2a7ceed/lists/ls_c9cecbc7_756b_
                                                            417b_a35f_fb03874979c0.xml)/columns/

(lco_3a31fd20_24f1_4422_99fd_c00d7bed4b24)?

utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
                                                            capabilities="urn:oracle:webcenter:read
                                                            urn:oracle:webcenter:update
                                                            urn:oracle:webcenter:delete"/>
                                            </links>
                                            <id>lco_3a31fd20_24f1_4422_99fd_c00d7bed4b24</id>
                                            <name>Description</name>
                                            <dataType>string</dataType>
                                            <required>false</required>
                                            <maxLength>500</maxLength>
                                            <displayLength>20</displayLength>
                                            <allowLinks>false</allowLinks>
                                            <editLines>1</editLines>
```

```
                    </metaColumn>
                </items>
            </columns>
        </list>
```

C.19.4.1.3 Resource Types Linked to from lists

Table C-63 lists the resource types that the client can link to from the `lists` resource.

Table C-63   Resource Types Linked to from lists

rel	resourceType
self	`urn:oracle:webcenter:space:lists`
	urn:oracle:webcenter:space:list

C.19.4.2 urn:oracle:webcenter:space:list

The `list` response provides a means of retrieving, updating, and deleting an individual list. This section includes the following topics:

- Navigation Paths to list

- Supported Methods for list

- Method (list): `GET`

- Method (list): `PUT`

- Method (list): `DELETE`

- Resource Types Linked to from list

C.19.4.2.1 Navigation Paths to list

This section shows how the client can navigate through the hypermedia to access the `list` resource:

```
urn:oracle:webcenter:resourceindex
     urn:oracle:webcenter:spaces
           urn:oracle:webcenter:space:resourceindex
                 urn:oracle:webcenter:space:lists
                       urn:oracle:webcenter:space:list
```

C.19.4.2.2 Supported Methods for list

The list resource type supports the following methods:

- Method (list): `GET`

- Method (list): `PUT`

- Method (list): `DELETE`

C.19.4.2.3 Method (list): **GET**

Use this method to retrieve a list.

- Resource URI, for example:

```
http://host:port/rest/api/spaces/vs1/lists/(/oracle/webcenter/list/scopedMD/
s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/ls_bb806754_0652_4d49_9354_
5fb62dc3514d.xml)?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**
```

- Request body: none

- Request headers: [Accept = `application/xml` | `application/json`]

- Response status: 200 [OK]

- Response body: `<list>`

- Retrieved elements (Table C-64)

Table C-64   Retrieved Elements for list

Element	Type	Description
id	string	ID of the list
name	string	Name of the list
description	string	Description of the list
scope.guid	string	GUID of the portal to which the list belongs
scope.name	string	Name of the portal to which the list belongs
creator	string	User who created the list
created	Date	Date on which the list was created
modifier	string	User who last modified the list
modified	Date	Date on which the list was last modified
columns	urn:oracle:webcenter:space:list:columns	The columns that make up the list

- For example:

```
<list resourceType="urn:oracle:webcenter:space:list">
    <links>
        <link resourceType="urn:oracle:webcenter:space:list" rel="self"
            href="http://host:port/rest/api/spaces/vs1/
            lists/(/oracle/webcenter/list/scopedMD/s355923f0_2f04_4fd0_83ad_
            f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
            fb03874979c0.xml)?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
            capabilities="urn:oracle:webcenter:read
            urn:oracle:webcenter:update
            urn:oracle:webcenter:delete"/>
        <link template="http://host:port
            /rest/api/spaces/vs1/lists/(/oracle/webcenter/list/scopedMD/
            s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/ls_c9cecbc7_756b_
            417b_a35f_fb03874979c0.xml)/rows?
            utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**&
            startIndex={startIndex}&itemsPerPage={itemsPerPage}&q=
            {searchTerms}&utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
            resourceType="urn:oracle:webcenter:space:list:rows"
            href="http://host:port/rest/api/
            spaces/vs1/lists/(/oracle/webcenter/list/scopedMD/s355923f0_
            2f04_4fd0_83ad_f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
```

```
                              fb03874979c0.xml)/rows?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
                              capabilities="urn:oracle:webcenter:read
                              urn:oracle:webcenter:create"/>
                 <link resourceType="urn:oracle:webcenter:space:list:columns"
                       href="http://host:port/rest/api/spaces/
                       vs1/lists/(/oracle/webcenter/list/scopedMD/s355923f0_2f04_4fd0_
                       83ad_f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
                       fb03874979c0.xml)/columns?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_
                       w**" capabilities="urn:oracle:webcenter:read
                       urn:oracle:webcenter:create"/>
         </links>
         <id>/oracle/webcenter/list/scopedMD/s355923f0_2f04_4fd0_83ad_
             f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_fb03874979c0.xml</id>
         <name>RestExample</name>
         <description>Created using rest api</description>
         <scope>
             <guid>s355923f0_2f04_4fd0_83ad_f7dac2a7ceed</guid>
             <name>vs1</name>
         </scope>
         <creator>weblogic</creator>
         <author>
             <links>
                 <link resourceType="urn:oracle:webcenter:people:person"
                       rel="via" href="http://host:port/
                       rest/api/people/E9A35E80F0BC11DFBFBE0FE339E55B21/lists/
                       @self?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
                       capabilities="urn:oracle:webcenter:read"/>
                 <link type="image/png"
                       template="http://host:port/
                       webcenter/profilephoto/45394133354538304630424331314446
                       4424642453046453333394453535423231/{size}?
                       _xResourceMethod=wsrp"
                       resourceType="urn:oracle:webcenter:people:person"
                       rel="urn:oracle:webcenter:people:icon"
                       href="http://host:port/
                       webcenter/profilephoto/45394133354538304630424331314446
                       4424642453046453333394453535423231/SMALL?
                       _xResourceMethod=wsrp"
                       capabilities="urn:oracle:webcenter:read"/>
                 <link type="text/html"
                       resourceType="urn:oracle:webcenter:spaces:profile"
                       rel="alternate" href="http://host:port/
                       webcenter/faces/oracle/webcenter/webcenterapp/view/pages/
                       peopleconn/UserProfileGallery.jspx?wc.username=weblogic"
                       capabilities="urn:oracle:webcenter:read"/>
             </links>
             <displayName>weblogic</displayName>
             <guid>E9A35E80F0BC11DFBFBE0FE339E55B21</guid>
             <id>weblogic</id>
         </author>
         <created>2010-11-18T06:10:32.250-08:00</created>
         <modifier>weblogic</modifier>
         <modifiedByUser>
             <displayName>weblogic</displayName>
             <guid>E9A35E80F0BC11DFBFBE0FE339E55B21</guid>
             <id>weblogic</id>
         </modifiedByUser>
         <modified>2010-11-18T06:10:32.250-08:00</modified>
         <columns resourceType="urn:oracle:webcenter:space:list:columns">
             <links>
                 <link resourceType="urn:oracle:webcenter:space:list:columns"
```

```
                          rel="self" href="http://host:port/
                          rest/api/spaces/vs1/lists/(/oracle/webcenter/list/
scopedMD/
                          s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/ls_c9cecbc7_
                          756b_417b_a35f_fb03874979c0.xml)/
                          columns?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
                          capabilities="urn:oracle:webcenter:read
                          urn:oracle:webcenter:create"/>
                  <link resourceType="urn:oracle:webcenter:space:list"
                        rel="urn:oracle:webcenter:parent"
                        href="http://host:port/rest/api/spaces/
                        vs1/lists/(/oracle/webcenter/list/scopedMD/s355923f0_2f04_
                        4fd0_83ad_f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
                        fb03874979c0.xml)?utoken=
                        FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"/>
              </links>
              <items>
                      <metaColumn resourceType=
                          "urn:oracle:webcenter:space:list:column">
                          <links>
                                  <link resourceType=
                                      "urn:oracle:webcenter:space:list:column"
                                      rel="self"
                                       href="http://host:port/
                                      rest/api/spaces/vs1/lists/(/oracle/webcenter/
                                      list/scopedMD/s355923f0_2f04_4fd0_83ad_
                                      f7dac2a7ceed/lists/ls_c9cecbc7_
                                      756b_417b_a35f_fb03874979c0.xml)/columns/(lco_
                                      9bdd1418_6004_40ba_a052_04e8335b7ee8)?
                                      utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
                                      capabilities="urn:oracle:webcenter:read
                                      urn:oracle:webcenter:update
                                      urn:oracle:webcenter:delete"/>
                          </links>
                          <id>lco_9bdd1418_6004_40ba_a052_04e8335b7ee8</id>
                          <name>No.</name>
                          <dataType>number</dataType>
                          <required>false</required>
                          <displayLength>10</displayLength>
                          <format>number</format>
                          <allowLinks>false</allowLinks>
                      </metaColumn>
                      <metaColumn resourceType=
                          "urn:oracle:webcenter:space:list:column">
                          <links>
                                  <link resourceType=
                                      "urn:oracle:webcenter:space:list:column"
                                      rel="self"
                                      href="http://host:port
                                      /rest/api/spaces/vs1/lists/(/oracle/webcenter/
                                      list/scopedMD/s355923f0_2f04_4fd0_83ad_
                                      f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
                                      fb03874979c0.xml)/columns/(lco_3a31fd20_24f1_
                                      4422_99fd_c00d7bed4b24)?
                                      utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
                                      capabilities="urn:oracle:webcenter:read
urn:oracle:webcenter:update
urn:oracle:webcenter:delete"/>
                          </links>
                          <id>lco_3a31fd20_24f1_4422_99fd_c00d7bed4b24</id>
```

```
                                    <name>Description</name>
                                    <dataType>string</dataType>
                                    <required>false</required>
                                    <maxLength>500</maxLength>
                                    <displayLength>20</displayLength>
                                    <allowLinks>false</allowLinks>
                                    <editLines>1</editLines>
                            </metaColumn>
                    </items>
            </columns>
    </list>
```

C.19.4.2.4 Method (list): `PUT`

Use this method to update a list name or description.

- Resource URI, for example:

```
http://host:port/rest/api/spaces/vs1/lists/(/oracle/webcenter/list/scopedMD/
s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/ls_bb806754_0652_4d49_9354_
5fb62dc3514d.xml)?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**
```

- Request body: `<list>`

- Writable elements for list (Table C-65)

Table C-65   Writable Elements for list

Element	Type	Required	Constraints	Description
name[1]	string	Yes	1 or more characters	Name of this list
description	string	No	none	Description of this list

[1]   Denotes required element

- For example:

```
<list>
        <name>RestExampleUpdate</name>
        <description>Updated using rest api</description>
</list>
```

- Request headers: `Content-Type = application/xml | application/json`, [Accept = `application/xml | application/json`]

- Response status: `200` [OK]

- Response body: `<list>`

C.19.4.2.5 Method (list): `DELETE`

Use this method to delete a list.

- Resource URI, for example:

```
http://host:port/rest/api/spaces/vs1/lists/(/oracle/webcenter/list/scopedMD/
s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/ls_bb806754_0652_4d49_9354_
5fb62dc3514d.xml)?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**
```

- Request body: none

- Response status: 204 [No Content]

- Response body: none

C.19.4.2.6 Resource Types Linked to from list

Table C-66 lists the resource types that the client can link to from the list resource.

Table C-66   Resource Types Linked to from list

rel	resourceType
self	urn:oracle:webcenter:space:list
	urn:oracle:webcenter:space:list:rows
	urn:oracle:webcenter:space:list:columns

C.19.4.3 urn:oracle:webcenter:space:list:rows

The rows response provides a means of retrieving and adding rows to a list. This section includes the following topics:

- Navigation Paths to rows

- Supported Methods for rows

- Resource Types Linked to from rows

C.19.4.3.1 Navigation Paths to rows

This section shows how the client can navigate through the hypermedia to access the rows resource:

```
urn:oracle:webcenter:resourceindex
      urn:oracle:webcenter:spaces
            urn:oracle:webcenter:space:resourceindex
                  urn:oracle:webcenter:space:lists
                        urn:oracle:webcenter:space:list
                              urn:oracle:webcenter:space:list:rows
```

C.19.4.3.2 Supported Methods for rows

The rows resource type supports the following methods:

- Method (rows): GET

- Method (rows): POST

Method (rows): GET

Use this method to retrieve list rows.

- Resource URI, for example:

```
http://host:port/rest/api/spaces/vs1/lists/(/oracle/webcenter/list/scopedMD/
s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/ls_bb806754_0652_4d49_9354_
5fb62dc3514d.xml)/rows?q=lco_9bdd1418_6004_40ba_a052_
04e8335b7ee8:equals:2&utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**
```

- Request body: none

- Request headers: [Accept = application/xml | application/json]

- Request parameters - `startIndex`, `itemsPerPage`, `q`, `projection`

  - `q` parameter for query, format `'q=columnId:operator:value'`, for example:

    `'q=lco_9bdd1418_6004_40ba_a052_04e8335b7ee8:equals:2'`

    - Supported operators for `Strings`: `equals`, `not.equals`, `contains`, `starts.with`

    - Supported operators for `Numbers`, `Dates`: `equals`, `not.equals`, `greater.than`, `greater.than.or.equals`, `less.than`, `less.than.or.equals`

    - May specify several conditions separated by semi-colon (;)

- Searchable attributes for rows: Rows are searchable by column values.

- Response status: `200` [OK]

- Response body: `<rows>`

- For example:

```
<rows resourceType="urn:oracle:webcenter:space:list:rows">
    <links>
        <link template="http://host:port/
            rest/api/spaces/vs1/lists/(/oracle/webcenter/list/
            scopedMD/s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/ls_
            c9cecbc7_756b_417b_a35f_fb03874979c0.xml)/rows?
            utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**&
            startIndex={startIndex}&itemsPerPage={itemsPerPage}&
            q={searchTerms}&utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
            resourceType="urn:oracle:webcenter:space:list:rows" rel="self"
            href="http://host:port/rest/api/spaces/
            vs1/lists/(/oracle/webcenter/list/scopedMD/s355923f0_2f04_4fd0_
            83ad_f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
            fb03874979c0.xml)/rows?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
            capabilities="urn:oracle:webcenter:read
            urn:oracle:webcenter:create"/>
        <link resourceType="urn:oracle:webcenter:space:list"
            rel="urn:oracle:webcenter:parent"
            href="http://host:port/rest/api/spaces/
            vs1/lists/(/oracle/webcenter/list/scopedMD/s355923f0_2f04_4fd0_
            83ad_f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
            fb03874979c0.xml)?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"/>
    </links>
    <itemsPerPage>1</itemsPerPage>
    <startIndex>0</startIndex>
    <items>
        <row resourceType="urn:oracle:webcenter:space:list:row">
            <links>
                <link resourceType="urn:oracle:webcenter:space:list:row"
                    rel="self" href="http://host:port/
                    rest/api/spaces/vs1/lists/(/oracle/webcenter/list/
                    scopedMD/s355923f0_2f04_4fd0_83ad_
                    f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
                    fb03874979c0.xml)/rows/(lr_a14d427f_8515_4c82_956f_
                    a60e6e08668c)?
utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
                    capabilities="urn:oracle:webcenter:read
```

```
                            urn:oracle:webcenter:update
                            urn:oracle:webcenter:delete"/>
                    </links>
                    <id>lr_a14d427f_8515_4c82_956f_a60e6e08668c</id>
                    <listId>/oracle/webcenter/list/scopedMD/s355923f0_2f04_4fd0_
                           83ad_f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
                           fb03874979c0.xml</listId>
                    <scope>s355923f0_2f04_4fd0_83ad_f7dac2a7ceed</scope>
                    <creator>weblogic</creator>
                    <author>
                            <links>
                                    <link resourceType=
                                            "urn:oracle:webcenter:people:person"
                                            rel="via"
                                            href="http://host:port/
                                            rest/api/people/
E9A35E80F0BC11DFBFBE0FE339E55B21/
                                            lists/@self?
utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_
                                            w**"
capabilities="urn:oracle:webcenter:read"/>
                                    <link type="image/png"
                                            template="http://host:port/
                                            webcenter/profilephoto/45394133354538304630424
                                            3313144464246424530464533333394535354233231/
                                            {size}?_xResourceMethod=wsrp"

resourceType="urn:oracle:webcenter:people:person"
                                            rel="urn:oracle:webcenter:people:icon"
                                            href="http://host:port/
                                            webcenter/profilephoto/45394133354538304630424
                                            3313144464246424530464533333394535354233231/
SMALL?
                                            _xResourceMethod=wsrp"
                                            capabilities="urn:oracle:webcenter:read"/>
                                    <link type="text/html"
                                            resourceType="urn:oracle:webcenter:spaces:
                                            profile" rel="alternate"
                                            href="http://host:port/
                                            webcenter/faces/oracle/webcenter/webcenterapp/
                                            view/pages/peopleconn/UserProfileGallery.jspx?
                                            wc.username=weblogic"
                                            capabilities="urn:oracle:webcenter:read"/>
                            </links>
                            <displayName>weblogic</displayName>
                            <guid>E9A35E80F0BC11DFBFBE0FE339E55B21</guid>
                            <id>weblogic</id>
                    </author>
                    <created>2010-11-18T07:12:06.599-08:00</created>
                    <modifier>weblogic</modifier>
                    <modifiedByUser>
                            <links>
                                    <link resourceType=
                                            "urn:oracle:webcenter:people:person"
rel="via"
                                            href="http://host:port/rest/
                                            api/people/E9A35E80F0BC11DFBFBE0FE339E55B21/
lists
                                            /@self?
utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
                                            capabilities="urn:oracle:webcenter:read"/>
```

```
                                        <link type="image/png"
                                               template="http://host:port/
                                               webcenter/profilephoto/4539413335453830463042
                                               33131444642462453046453333394535354 23231/
                                               {size}?_xResourceMethod=wsrp"

                    resourceType="urn:oracle:webcenter:people:person"
                                               rel="urn:oracle:webcenter:people:icon"
                                               href="http://host:port/
                                               webcenter/profilephoto/4539413335453830463042
                                               33131444642462453046453333394535354 23231/
                    SMALL?
                                               _xResourceMethod=wsrp"
                                               capabilities="urn:oracle:webcenter:read"/>
                                        <link type="text/html"
                                               resourceType="urn:oracle:webcenter:spaces:
                                               profile" rel="alternate"
                                               href="http://host:port/
                                               webcenter/faces/oracle/webcenter/webcenterapp/
                                               view/pages/peopleconn/UserProfileGallery.jspx?
                                               wc.username=weblogic"
                                               capabilities="urn:oracle:webcenter:read"/>
                                </links>
                                <displayName>weblogic</displayName>
                                <guid>E9A35E80F0BC11DFBFBE0FE339E55B21</guid>
                                <id>weblogic</id>
                        </modifiedByUser>
                        <modified>2010-11-18T07:12:06.599-08:00</modified>
                        <columns>
                                <column>
                                        <id>lco_9bdd1418_6004_40ba_a052_04e8335b7ee8</id>
                                        <name>No.</name>
                                        <value>1.0</value>
                                </column>
                                <column>
                                        <id>lco_3a31fd20_24f1_4422_99fd_c00d7bed4b24</id>
                                        <name>Description</name>
                                        <value>test</value>
                                </column>
                        </columns>
                </row>
        </items>
</rows>
```

Method (rows): POST

Use this method to create a row in a list.

*   Resource URI, for example:

```
http://host:port/rest/api/spaces/vs1/lists/(/oracle/webcenter/list/scopedMD/
s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/ls_bb806754_0652_4d49_9354_
5fb62dc3514d.xml)/rows?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**
```

*   Request body: <row>

*   Writable elements (Table C-67)

Table C-67   Writable Elements for rows

Table C-67   (Cont.) Writable Elements for rows

Element	Type	Constraints	Description
columns.column.id[1]	string	—	Column ID
columns.column.value[2]	string	Valid value for column data type	Column value

[1] Denotes required element
[2] Denotes required element

- For example:

```
<row>
      <columns>
            <column>
                  <id>lco_9bdd1418_6004_40ba_a052_04e8335b7ee8</id>
                  <value>1</value>
            </column>
            <column>
                  <id>lco_3a31fd20_24f1_4422_99fd_c00d7bed4b24</id>
                  <value>test</value>
            </column>
      </columns>
</row>
```

- Request headers: `Content-Type = application/xml | application/json`, [Accept = `application/xml | application/json`]

- Response status: `201` [Created]

- Response body: `<row>`

- Retrieved elements (Table C-68)

Table C-68   Retrieved Elements for rows

Element	Type	Description
id	string	Row ID
list id	string	List ID
scope	string	GUID of portal to which list belongs
creator	string	Person who created the list
created	date	Date the list was created
modifier	string	Last user to modify the list
modified	date	Date the list was last modified
columns	—	Column values

- For example:

```
<row resourceType="urn:oracle:webcenter:space:list:row">
      <links>
            <link resourceType="urn:oracle:webcenter:space:list:row" rel="self"
```

```
                              href="http://host:port/rest/api/spaces/vs1/
                              lists/(/oracle/webcenter/list/scopedMD/s355923f0_2f04_4fd0_83ad_
                              f7dac2a7ceed/lists/ls_bb806754_0652_4d49_9354_
                              5fb62dc3514d.xml)/rows/(lr_4cc07327_49cb_4cd5_a270_
                              182ddcc8db4a)?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
                              capabilities="urn:oracle:webcenter:read
                              urn:oracle:webcenter:update urn:oracle:webcenter:delete"/>
              </links>
              <id>lr_4cc07327_49cb_4cd5_a270_182ddcc8db4a</id>
              <listId>/oracle/webcenter/list/scopedMD/s355923f0_2f04_4fd0_83ad_
                    f7dac2a7ceed/lists/ls_bb806754_0652_4d49_9354_
                    5fb62dc3514d.xml</listId>
              <scope>s355923f0_2f04_4fd0_83ad_f7dac2a7ceed</scope>
              <creator>weblogic</creator>
              <author>
                    <links>
                          <link resourceType="urn:oracle:webcenter:people:person"
                                rel="via" href="http://host:port
                                /rest/api/people/E9A35E80F0BC11DFBFBE0FE339E55B21/lists/
                                @self?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
                                capabilities="urn:oracle:webcenter:read"/>
                          <link type="image/png"
                                template="http://host:port/
                                webcenter/profilephoto/4539413335453830463042433131
                                4446424642453046453333339453535423231/{size}?
                                _xResourceMethod=wsrp"
                                resourceType="urn:oracle:webcenter:people:person"
                                rel="urn:oracle:webcenter:people:icon"
                                href="http://host:port/webcenter/
                                profilephoto/4539413335453830463042433131444642464245 30
                                4645333339453535423231/SMALL?_xResourceMethod=wsrp"
                                capabilities="urn:oracle:webcenter:read"/>
                          <link type="text/html"
                                resourceType="urn:oracle:webcenter:spaces:profile"
                                rel="alternate" href="http://host:port/
                                webcenter/faces/oracle/webcenter/webcenterapp/view/pages/
                                peopleconn/UserProfileGallery.jspx?wc.username=weblogic"
                                capabilities="urn:oracle:webcenter:read"/>
                    </links>
                    <displayName>weblogic</displayName>
                    <guid>E9A35E80F0BC11DFBFBE0FE339E55B21</guid>
                    <id>weblogic</id>
              </author>
              <created>2010-11-17T06:34:25.042-08:00</created>
              <modifier>weblogic</modifier>
              <modifiedByUser>
                    <links>
                          <link resourceType="urn:oracle:webcenter:people:person"
                                rel="via" href="http://host:port/
                                rest/api/people/E9A35E80F0BC11DFBFBE0FE339E55B21/lists/
                                @self?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
                                capabilities="urn:oracle:webcenter:read"/>
                          <link type="image/png"
                                template="http://host:port/
                                webcenter/profilephoto/4539413335453830463042433131
                                4446424642453046453333339453535423231/{size}?
                                _xResourceMethod=wsrp"
                                resourceType="urn:oracle:webcenter:people:person"
                                rel="urn:oracle:webcenter:people:icon"
                                href="http://host:port/webcenter/
                                profilephoto/4539413335453830463042433131444642464245 30
```

```
                                        4645333339453535423231/SMALL?_xResourceMethod=wsrp"
                                        capabilities="urn:oracle:webcenter:read"/>
                            <link type="text/html"
                                        resourceType="urn:oracle:webcenter:spaces:profile"
                                        rel="alternate" href="http://host:port/
                                        webcenter/faces/oracle/webcenter/webcenterapp/view/pages/
                                        peopleconn/UserProfileGallery.jspx?wc.username=weblogic"
                                        capabilities="urn:oracle:webcenter:read"/>
                    </links>
                    <displayName>weblogic</displayName>
                    <guid>E9A35E80F0BC11DFBFBE0FE339E55B21</guid>
                    <id>weblogic</id>
            </modifiedByUser>
                    <modified>2010-11-17T06:34:25.042-08:00</modified>
            <columns>
                    <column>
                            <id>lco_9bdd1418_6004_40ba_a052_04e8335b7ee8</id>
                            <name>No.</name>
                            <value>1.0</value>
                    </column>
                    <column>
                            <id>lco_3a31fd20_24f1_4422_99fd_c00d7bed4b24</id>
                            <name>Description</name>
                            <value>test</value>
                    </column>
            </columns>
        </row>
```

C.19.4.3.3 Resource Types Linked to from rows

Table C-69 lists the resource types that the client can link to from the `rows` resource.

Table C-69 Resource Types Linked to from rows

rel	resourceType
self	`urn:oracle:webcenter:space:list.rows`
parent	urn:oracle:webcenter:space:list
	urn:oracle:webcenter:space:list:row

C.19.4.4 urn:oracle:webcenter:space:list:row

The `row` response provides a means of retrieving and adding, and deleting an individual list row. This section includes the following topics:

- Navigation Paths to row

- Supported Methods for row

- Method (row): `GET`

- Method (row): `PUT`

- Method (row): `DELETE`

- Resource Types Linked to from row

C.19.4.4.1 Navigation Paths to row

This section shows how the client can navigate through the hypermedia to access the row resource:

```
urn:oracle:webcenter:resourceindex
      urn:oracle:webcenter:spaces
            urn:oracle:webcenter:space:resourceindex
                  urn:oracle:webcenter:space:lists
                        urn:oracle:webcenter:space:list
                              urn:oracle:webcenter:space:list:rows
                                    urn:oracle:webcenter:space:list:row
```

C.19.4.4.2 Supported Methods for row

The row resource type supports the following methods:

- Method (row): `GET`

- Method (row): `PUT`

- Method (row): `DELETE`

C.19.4.4.3 Method (row): `GET`

Use this method to retrieve data from a list row.

- Resource URI, for example:

```
http://host:port/rest/api/spaces/vs1/lists/(/oracle/webcenter/list/scopedMD/
s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
fb03874979c0.xml)/rows/(lr_a14d427f_8515_4c82_956f_
a60e6e08668c)?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**
```

- Request body: none

- Request headers: [Accept = `application/xml` | `application/json`]

- Response status: `200` [OK]

- Response body: `<row>`

- Retrieved elements (Table C-70)

Table C-70   Retrieved Elements for row

Element	Type	Description
id	string	ID of the row
listId	string	ID of the parent list
scope	string	GUID of the portal to which the list belongs
creator	string	User who created the row
created	Date	Date on which the row was created
modifier	string	User who last modified the row
modified	Date	Date on which the row was last modified

Table C-70   (Cont.) Retrieved Elements for row

Element	Type	Description
columns	urn:oracle:webcenter:space:list:columns	Column values

- For example:

```
<row resourceType="urn:oracle:webcenter:space:list:row">
    <links>
        <link resourceType="urn:oracle:webcenter:space:list:row"
            rel="self" href="http://host:port/
            rest/api/spaces/vs1/lists/((/oracle/webcenter/list/scopedMD
            /s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/ls_bb806754_0652_
            4d49_9354_5fb62dc3514d.xml)/rows/(lr_4cc07327_49cb_4cd5_a270_
            182ddcc8db4a)?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
            capabilities="urn:oracle:webcenter:read
            urn:oracle:webcenter:update urn:oracle:webcenter:delete"/>
    </links>
    <id>lr_4cc07327_49cb_4cd5_a270_182ddcc8db4a</id>
    <listId>/oracle/webcenter/list/scopedMD/s355923f0_2f04_4fd0_83ad_
        f7dac2a7ceed/lists/ls_bb806754_0652_4d49_9354_
        5fb62dc3514d.xml</listId>
    <scope>s355923f0_2f04_4fd0_83ad_f7dac2a7ceed</scope>
    <creator>weblogic</creator>
    <author>
        <links>
            <link resourceType="urn:oracle:webcenter:people:person"
                rel="via" href="http://host:port/
                rest/api/people/E9A35E80F0BC11DFBFBE0FE339E55B21/lists/
                @self?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
                capabilities="urn:oracle:webcenter:read"/>
            <link type="image/png"
                template="http://host:port/webcenter/
                profilephoto/45394133354538304630424331314446424642453  0
                4645333339453535423231/{size}?_xResourceMethod=wsrp"
                resourceType="urn:oracle:webcenter:people:person"
                rel="urn:oracle:webcenter:people:icon"
                href="http://host:port/webcenter/
                profilephoto/45394133354538304630424331314446424642453  0
                4645333339453535423231/SMALL?_xResourceMethod=wsrp"
                capabilities="urn:oracle:webcenter:read"/>
            <link type="text/html"
                resourceType="urn:oracle:webcenter:spaces:profile"
                rel="alternate" href="http://host:port/
                webcenter/faces/oracle/webcenter/webcenterapp/view/pages/
                peopleconn/UserProfileGallery.jspx?wc.username=weblogic"
                capabilities="urn:oracle:webcenter:read"/>
        </links>
        <displayName>weblogic</displayName>
        <guid>E9A35E80F0BC11DFBFBE0FE339E55B21</guid>
        <id>weblogic</id>
    </author>
    <created>2010-11-17T06:34:25.042-08:00</created>
    <modifier>weblogic</modifier>
    <modifiedByUser>
        <links>
            <link resourceType="urn:oracle:webcenter:people:person"
                rel="via" href="http://host:port/
                rest/api/people/E9A35E80F0BC11DFBFBE0FE339E55B21/lists/
                @self?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
```

```
                                        capabilities="urn:oracle:webcenter:read"/>
                        <link type="image/png"
                              template="http://host:port/webcenter/
                              profilephoto/45394133354538304630424331314446424642453
                              0464533333394534535423231/{size}?_xResourceMethod=wsrp"
                              resourceType="urn:oracle:webcenter:people:person"
                              rel="urn:oracle:webcenter:people:icon"
                              href="http://host:port/webcenter/
                              profilephoto/45394133354538304630424331314446424642453
                              0464533333394534535423231/SMALL?_xResourceMethod=wsrp"
                              capabilities="urn:oracle:webcenter:read"/>
                        <link type="text/html"
                              resourceType="urn:oracle:webcenter:spaces:profile"
                              rel="alternate" href="http://host:port/
                              webcenter/faces/oracle/webcenter/webcenterapp/view/pages/
                              peopleconn/UserProfileGallery.jspx?wc.username=weblogic"
                              capabilities="urn:oracle:webcenter:read"/>
                </links>
                <displayName>weblogic</displayName>
                <guid>E9A35E80F0BC11DFBFBE0FE339E55B21</guid>
                <id>weblogic</id>
        </modifiedByUser>
        <modified>2010-11-17T06:34:25.042-08:00</modified>
        <columns>
                <column>
                        <id>lco_9bdd1418_6004_40ba_a052_04e8335b7ee8</id>
                        <name>No.</name>
                        <value>1.0</value>
                </column>
                <column>
                        <id>lco_3a31fd20_24f1_4422_99fd_c00d7bed4b24</id>
                        <name>Description</name>
                        <value>test</value>
                </column>
        </columns>
    </row>
```

C.19.4.4.4 Method (row): `PUT`

Use this method to update row data.

- Resource URI, for example:

```
http://host:port/rest/api/spaces/vs1/lists/(/oracle/webcenter/list/scopedMD/
s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
fb03874979c0.xml)/rows/(lr_a14d427f_8515_4c82_956f_
a60e6e08668c)?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**
```

- Request body: `<row>`

- Writable elements (Table C-71)

Table C-71   Writable Elements for row

Element	Type	Constraints	Description
columns.column.id[1]	string	—	Column ID
columns.column.value[2]	string	valid value for column data type	Column value

[1] Denotes required element

² Denotes required element

- For example:

```
<row>
      <columns>
            <column>
                  <id>lco_9bdd1418_6004_40ba_a052_04e8335b7ee8</id>
                  <value>1</value>
            </column>
            <column>
                  <id>lco_3a31fd20_24f1_4422_99fd_c00d7bed4b24</id>
                  <value>test</value>
            </column>
      </columns>
</row>
```

- Request headers: `Content-Type = application/xml | application/json`, `[Accept = application/xml | application/json]`

- Response status: `200` [OK]

- Response body: `<row>`

C.19.4.4.5 Method (row): `DELETE`

Use this method to delete a list row.

- Resource URI, for example:

```
http://host:port/rest/api/spaces/vs1/lists/(/oracle/webcenter/list/scopedMD/
s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
fb03874979c0.xml)/rows/(lr_a14d427f_8515_4c82_956f_
a60e6e08668c)?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**
```

- Request body: none

- Response status: `204` [No Content]

- Response body: none

C.19.4.4.6 Resource Types Linked to from row

Table C-72 lists the resource types that the client can link to from the `row` resource.

Table C-72   Resource Types Linked to from row

rel	resourceType
self	`urn:oracle:webcenter:space:list:row`

C.19.4.5 urn:oracle:webcenter:space:list:columns

The `columns` response provides a means of retrieving and adding list columns. This section includes the following topics:

- Navigation Paths to columns

- Supported Methods for columns

- Method (columns): `GET`

- Method (columns): `POST`

- Resource Types Linked to from columns

C.19.4.5.1 Navigation Paths to columns

This section shows how the client can navigate through the hypermedia to access the `columns` resource:

```
urn:oracle:webcenter:resourceindex
      urn:oracle:webcenter:spaces
            urn:oracle:webcenter:space:resourceindex
                  urn:oracle:webcenter:space:lists
                        urn:oracle:webcenter:space:list
                              urn:oracle:webcenter:space:list:columns
```

C.19.4.5.2 Supported Methods for columns

The `columns` resource type supports the following methods:

- Method (columns): `GET`

- Method (columns): `POST`

C.19.4.5.3 Method (columns): `GET`

Use this method to retrieve columns in a list.

- Resource URI, for example:

  ```
  http://host:port/rest/api/spaces/vs1/lists/(/oracle/webcenter/list/scopedMD/
  s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
  fb03874979c0.xml)/columns?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**
  ```

- Request body: none

- Request headers: [Accept = `application/xml` | `application/json`]

- Request parameters: none

- Response status: `200` [OK]

- Response body: `<metaColumns>`

- For example:

  ```
  <metaColumns resourceType="urn:oracle:webcenter:space:list:columns">
        <links>
              <link resourceType="urn:oracle:webcenter:space:list:columns"
                    rel="self" href="http://host:port/
                    rest/api/spaces/vs1/lists/(/oracle/webcenter/list/scopedMD/
                    s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/ls_c9cecbc7_756b_
                    417b_a35f_fb03874979c0.xml)/columns?
                    utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
                    capabilities="urn:oracle:webcenter:read
                    urn:oracle:webcenter:create"/>
              <link resourceType="urn:oracle:webcenter:space:list"
                    rel="urn:oracle:webcenter:parent"
                    href="http://host:port/rest/api/spaces/vs1/
                    lists/(/oracle/webcenter/list/scopedMD/s355923f0_2f04_4fd0_83ad_
                    f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
                    fb03874979c0.xml)?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"/>
        </links>
  ```

```
        <items>
                <metaColumn resourceType="urn:oracle:webcenter:space:list:column">
                        <links>
                                <link
resourceType="urn:oracle:webcenter:space:list:column"
                                        rel="self" href="http://host:port/
                                        rest/api/spaces/vs1/lists/(/oracle/webcenter/list/
                                        scopedMD/s355923f0_2f04_4fd0_83ad_
                                        f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
                                        fb03874979c0.xml)/columns/(lco_9bdd1418_6004_40ba_
                                        a052_04e8335b7ee8)?
                                        utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
                                        capabilities="urn:oracle:webcenter:read
                                        urn:oracle:webcenter:update
                                        urn:oracle:webcenter:delete"/>
                        </links>
                        <id>lco_9bdd1418_6004_40ba_a052_04e8335b7ee8</id>
                        <name>No.</name>
                        <dataType>number</dataType>
                        <required>false</required>
                        <displayLength>10</displayLength>
                        <format>number</format>
                        <allowLinks>false</allowLinks>
                </metaColumn>
                <metaColumn resourceType="urn:oracle:webcenter:space:list:column">
                        <links>
                                <link
resourceType="urn:oracle:webcenter:space:list:column"
                                        rel="self" href="http://host:port/
                                        rest/api/spaces/vs1/lists/(/oracle/webcenter/list/
                                        scopedMD/s355923f0_2f04_4fd0_83ad_
                                        f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
                                        fb03874979c0.xml)/columns/(lco_3a31fd20_24f1_4422_
                                        99fd_c00d7bed4b24)?
                                        utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
                                        capabilities="urn:oracle:webcenter:read
                                        urn:oracle:webcenter:update
                                        urn:oracle:webcenter:delete"/>
                        </links>
                        <id>lco_3a31fd20_24f1_4422_99fd_c00d7bed4b24</id>
                        <name>Description</name>
                        <dataType>string</dataType>
                        <required>false</required>
                        <maxLength>500</maxLength>
                        <displayLength>20</displayLength>
                        <allowLinks>false</allowLinks>
                        <editLines>1</editLines>
                </metaColumn>
        </items>
</metaColumns>
```

C.19.4.5.4 Method (columns): `POST`

Use this method to create a column in a list.

- Resource URI, for example:

```
http://host:port/rest/api/spaces/vs1/lists/(/oracle/webcenter/list/scopedMD/
s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
fb03874979c0.xml)/columns?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**
```

- Request body: `<metaColumn>`

- Writable elements (Table C-73)

Table C-73   Writable Elements for columns

Element	Type	Constraints	Description
name[1]	string	—	The name of the column
dataType[2]	string	– `string` – `number` – `datetime` – `boolean` – `person` – `image` – `richtext`	The data type of the column
required	boolean	– `true` – `false`	Whether a value is required for the column
defaultValue	data type of column	Object must match data type	The default value of the column
maxLength	int	Valid only for `string` data type	The maximum length for a string value
rangeLow	int	Valid only for `number` data type	The low range value for a `number` data type
rangeHigh	int	Valid only for `number` data type	The high range value for a `number` data type
format	string	For `number` data type: – number – currency – percent For `datetime` data type: – date – time – both	The format of the column
allowLinks	boolean	– `true` – `false` Valid only for `string` data type	Whether a hyperlink can be specified for a column value
linkTarget	string	– `_blank` – `_self` Valid only if `allowLinks=true`	`_blank` opens link in a new window, `_self` opens link in the same window
editLines	int	Valid only for `string` data type	The number of lines when editing a column value (default=1)
peopleScope	string	– GLOBAL – SUB_SCOPE Valid only for `person` data type	Whether valid users are all users in directory or members of the portal that contains the list
displayWidth	int	—	Display width of column in pixels
hint	string	—	Hint displayed to help user when entering column value

[1] Denotes required element

[2] Denotes required element

- For example:

```
<metaColumn>
        <name>Notes</name>
        <dataType>richtext</dataType>
</metaColumn
```

- Request headers: Content-Type = application/xml | application/json, [Accept = application/xml | application/json]

- Response status: 201 [Created]

- Response body: <metaColumn>

- Retrieved elements (Table C-74)

Table C-74   Retrieved Elements from columns

Element	Type	Description
id	string	Column ID
name	string	The name of the column
dataType	string	The data type of the column
required	boolean	Whether a value is required for the column
defaultValue	data type of column	The default value of the column
maxLength	int	The maximum length for a string value
rangeLow	int	The low range value for a number data type
rangeHigh	int	The high range value for a number data type
format	string	The format of the column
allowLinks	boolean	Whether a hyperlink can be specified for a column value
linkTarget	string	_blank opens link in a new window, _self opens link in the same window
editLines	int	The number of lines when editing a column value (default=1)
peopleScope	string	Whether valid users are all users in directory or members of the portal that contains the list
displayWidth	int	Display width of column in pixels
hint	string	Hint displayed to help user when entering column value

- For example:

```
<metaColumn resourceType="urn:oracle:webcenter:space:list:column">
        <links>
                <link resourceType="urn:oracle:webcenter:space:list:column"
                        rel="self" href="http://host:port/
                        rest/api/spaces/vs1/lists/(/oracle/webcenter/list/scopedMD/
```

```
                                    s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/
                                    ls_c9cecbc7_756b_417b_a35f_fb03874979c0.xml)/columns/
                                    (lco_fe2b9856_32f3_449a_a277_18dc7f6a779e)?
                                    utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
                                    capabilities="urn:oracle:webcenter:read
                                    urn:oracle:webcenter:update urn:oracle:webcenter:delete"/>
                        </links>
                        <id>lco_fe2b9856_32f3_449a_a277_18dc7f6a779e</id>
                        <name>Notes</name>
                        <dataType>richtext</dataType>
                        <required>false</required>
                        <displayLength>20</displayLength>
                        <allowLinks>false</allowLinks>
                </metaColumn>
```

C.19.4.5.5 Resource Types Linked to from columns

Table C-75 lists the resource types that the client can link to from the `columns` resource.

Table C-75   Resource Types Linked to from columns

rel	resourceType
self	urn:oracle:webcenter:space:list.columns
parent	urn:oracle:webcenter:space:list
	urn:oracle:webcenter:space:list:column

C.19.4.6 urn:oracle:webcenter:space:list:column

The `column` response provides a means of retrieving, adding, and deleting an individual list column. This section includes the following topics:

- Navigation Paths to column

- Supported Methods for column

- Resource Types Linked to from column

C.19.4.6.1 Navigation Paths to column

This section shows how the client can navigate through the hypermedia to access the `column` resource:

```
urn:oracle:webcenter:resourceindex
      urn:oracle:webcenter:spaces
            urn:oracle:webcenter:space:resourceindex
                  urn:oracle:webcenter:space:lists
                        urn:oracle:webcenter:space:list
                              urn:oracle:webcenter:space:list:columns
                                    urn:oracle:webcenter:space:list:column
```

C.19.4.6.2 Supported Methods for column

The column resource type supports the following methods:

- Method (column): GET

- Method (column): PUT

- Method (column): DELETE

Method (column): GET

Use this method to retrieve a list column.

- Resource URI. for example:

```
http://host:port/rest/api/spaces/vs1/lists/(/oracle/webcenter/list/scopedMD/
s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
fb03874979c0.xml)/columns/(lco_fe2b9856_32f3_449a_a277_
18dc7f6a779e)?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**
```

- Request body: none

- Request headers: [Accept = application/xml | application/json]

- Response status: 200 [OK]

- Response body: <metaColumn>

- Retrieved elements (Table C-76)

Table C-76   column Retrieved Elements

Element	Type	Description
id	string	Column ID
name	string	The name of the column
dataType	string	The data type of the column
required	boolean	Whether a value is required for the column
defaultValue	data type of column	The default value of the column
maxLength	int	The maximum length for a string value
rangeLow	int	The low range value for a number data type
rangeHigh	int	The high range value for a number data type
format	string	The format of the column
allowLinks	boolean	Whether a hyperlink can be specified for a column value
linkTarget	string	_blank opens link in a new window, _self opens link in the same window
editLines	int	The number of lines when editing a column value (default=1)
peopleScope	string	Whether valid users are all users in directory or members of the portal that contains the list
displayWidth	int	Display width of column in pixels
hint	string	Hint displayed to help user when entering column value

- For example:

```
<metaColumn resourceType="urn:oracle:webcenter:space:list:column">
    <links>
        <link resourceType="urn:oracle:webcenter:space:list:column"
            rel="self" href="http://host:port/
            rest/api/spaces/vs1/lists/(/oracle/webcenter/list/scopedMD/
            s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/
            ls_c9cecbc7_756b_417b_a35f_fb03874979c0.xml)/
            columns/(lco_fe2b9856_32f3_449a_a277_18dc7f6a779e?
            utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**"
            capabilities="urn:oracle:webcenter:read
            urn:oracle:webcenter:update urn:oracle:webcenter:delete"/>
    </links>
    <id>lco_fe2b9856_32f3_449a_a277_18dc7f6a779e</id>
    <name>Notes</name>
    <dataType>richtext</dataType>
    <required>false</required>
    <displayLength>20</displayLength>
    <allowLinks>false</allowLinks>
</metaColumn>
```

Method (column): PUT

Use this method to update column data.

- Resource URI, for example:

```
http://host:port/rest/api/spaces/vs1/lists/(/oracle/webcenter/list/scopedMD/
s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
fb03874979c0.xml)/columns/(lco_fe2b9856_32f3_449a_a277_
18dc7f6a779e)?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**
```

- Request body: <metacolumn>

- Writable elements (Table C-77)

Table C-77   column Writable Elements

Element	Type	Constraints	Description
name[1]	string	—	The name of the column
dataType[2]	string	– string – number – datetime – boolean – person – image – richtext	The data type of the column
required	boolean	– true – false	Whether a value is required for the column
defaultValue	data type of column	Object must match data type	The default value of the column
maxLength	int	Valid only for string data type	The maximum length for a string value
rangeLow	int	Valid only for number data type	The low range value for a number data type
rangeHigh	int	Valid only for number data type	The high range value for a number data type

Table C-77   (Cont.) column Writable Elements

Element	Type	Constraints	Description
format	string	For `number` data type: – number – currency – percent For `datetime` data type: – date – time – both	The format of the column
allowLinks	boolean	– `true` – `false` Valid only for `string` data type	Whether a hyperlink can be specified for a column value
linkTarget	string	– `_blank` – `_self` Valid only if `allowLinks=true`	`_blank` opens link in a new window, `_self` opens link in the same window
editLines	int	Valid only for `string` data type	The number of lines when editing a column value (default=1)
peopleScope	string	– `GLOBAL` – `SUB_SCOPE` Valid only for `person` data type	Whether valid users are all users in directory or members of the portal that contains the list
displayWidth	int	—	Display width of column in pixels
hint	string	—	Hint displayed to help user when entering column value

[1] Denotes required element

[2] Denotes required element

- For example:

```
<metaColumn>
        <name>Comments</name>
        <dataType>richtext</dataType>
</metaColumn>
```

- Request headers: `Content-Type` = `application/xml` | `application/json`, [Accept = `application/xml` | `application/json`]

- Response status: `200` [OK]

- Response body: `<metaColumn>`

Method (column): `DELETE`

Use this method to delete a list column.

- Resource URI, for example:

```
http://host:port/rest/api/spaces/vs1/lists/(/oracle/webcenter/list/scopedMD/
s355923f0_2f04_4fd0_83ad_f7dac2a7ceed/lists/ls_c9cecbc7_756b_417b_a35f_
fb03874979c0.xml)/columns/(lco_fe2b9856_32f3_449a_a277_
18dc7f6a779e)?utoken=FKld8lalI3QRdi8TgQkOCGEzxL5x_w**
```

- Request body: none

- Response status: 204 [No Content]

- Response body: none

C.19.4.6.3 Resource Types Linked to from column

Table C-78 lists the resource types that the client can link to from the column resource.

Table C-78   Resource Types Linked to from column

rel	resourceType
self	urn:oracle:webcenter:space:list:column

C.20 Using the Activity Graph REST APIs

WebCenter Portal provides REST APIs to support the activity graph. Use the activity graph REST APIs to create your own interface for providing recommendations for connections, portals, and items.

This section describes the REST APIs associated with the activity graph. It includes the following topics:

- Activity Graph Entry Point

- Activity Graph Resource Type Taxonomy

- Security Considerations

- Activity Graph Resource Types

For an introduction to the REST APIs, see Introduction to the WebCenter Portal REST APIs.

C.20.1 Activity Graph Entry Point

Each REST service has a link element within the Resource Index that provides the entry point for that service. For the activity graph, there are two entry points: one for recommendations and one for items. To find the entry points for the activity graph, find the link element with one of the following resourceTypes:

```
urn:oracle:webcenter:activitygraph:recommendations
urn:oracle:webcenter:activitygraph:items
```

The corresponding href or template element provides the URI entry point. The client sends HTTP requests to this entry point to work with the activity graph.

For more information about the Resource Index, see Using the Resource Index.

For more information about resource types, see Resource Type.

C.20.2 Activity Graph Resource Type Taxonomy

When the client has identified the entry point to use, it can then navigate through the resource type taxonomy to perform the required operations.

The taxonomy for the activity graph is:

```
urn:oracle:webcenter:activitygraph:recommendations
    urn:oracle:webcenter:activitygraph:recommendations:recommendation
urn:oracle:webcenter:activitygraph:items
    urn:oracle:webcenter:activitygraph:items:item
```

C.20.3 Security Considerations

For general security considerations, see Security Considerations for WebCenter Portal REST APIs.

C.20.4 Activity Graph Resource Types

This section provides you with all the information you need to know about each resource type. It includes the following topics:

- urn:oracle:webcenter:activitygraph:recommendations

- urn:oracle:webcenter:activitygraph:recommendations:recommendation

- urn:oracle:webcenter:activitygraph:items

- urn:oracle:webcenter:activitygraph:items:item

C.20.4.1 urn:oracle:webcenter:activitygraph:recommendations

Use this resource to identify the URI to use to retrieve (`GET`) recommended connections, portals, or items based on their similarity to the specified object. The response from a `GET` operation includes each object in the requested list, and each object includes links used to operate on that object.

Nodes in the activity graph are identified by a combination of node class URN and object URN.

For example, to identify the user, monty, you can specify:

- `classURN=WC.user`

- `objectURN=monty`

The nodes provided out of the box are all WebCenter Portal resources (users, documents, portals, and so on) and so have WebCenter Portal service and resource IDs. The activity graph REST APIs provide another way for you to identify these out of the box nodes using the `serviceId` and `objectURN` (which contains the resource ID).

For example, to identify the user, monty, you can specify:

- `serviceId=oracle.webcenter.people`

- `objectURN=monty`

If a service's objects are further classified by resource type, for example, the Documents tool, you must also specify the resource type.

For example, to identify a particular document, you can specify:

- `serviceId=oracle.webcenter.doclib`

- `resourceType=document`

- `objectURN=document1`

Any new node classes that are created (that is, non-native WebCenter Portal objects) should be identified using node class and object URNs.

C.20.4.1.1 Navigation Paths to recommendations

This section shows how the client can navigate through the hypermedia to access this resource:

```
resourceIndex
    recommendations
```

C.20.4.1.2 Supported Methods for recommendations

The following methods are supported by this resource:

- GET

    – **request - Parameters:** `startIndex`, `itemsPerPage`, `utoken`

    For information about these common parameters, see Common Request Query Parameters.

    The following additional parameters are available:

    ◆ `classURN`—The node class URN that identifies the type of the object for which you are requesting recommendations. For example `WC.user`, `WC.group-document`.

    ◆ `objectURN`—The object URN that provides a unique identifier for the object for which you are requesting recommendations. For example `monty`, `1000`.

    ◆ `recipe`—A semicolon-separated list of similarity URNs and optional associated weights (indicated by a colon) that is used to determine which objects to recommend. For example, `gs-edit:10;gs-all:1`

    ◆ `classURNRestrictions`—A comma-separated list of types of objects, identified by node class URN, to exclude from the recommendations

    ◆ `excludeObjectActions`—A comma-separated list of actions, identified by action URN, used to exclude objects from the recommendations if the current user has performed that action on the object. For example, if the client is retrieving recommended portals, then the `gs-edit` action could be specified to exclude portals that the current user has edited (since the user already knows about those portals).

    ◆ `serviceId`—The WebCenter Portal service ID that identifies the type of object for which you are requesting recommendations (you can use this instead of `classURN` for out of the box objects).

    ◆ `resourceType`—The WebCenter Portal resource type of the object for which you are requesting recommendations, used in combination with `serviceId` if necessary.

    ◆ `userCredentialClassURN`—The node class URN of the user exercising the REST API. The default value is `WC.user`. If you are integrating the activity graph engine with another application, this may need to be a different node class.

    – **response - body:** 0 or more recommendations

For more information, see Templates.

C.20.4.1.3 Resource Types Linked to from recommendations

Table C-79 lists the resource types that the client can link to from this resource.

Table C-79   Related Resource Types for recommendations

rel	resourceType
self	urn:oracle:webcenter:activitygraph:recommendations

C.20.4.2 urn:oracle:webcenter:activitygraph:recommendations:recommendation

The recommendation response contains the recommended objects and the URIs for use in accessing those objects.

C.20.4.2.1 Navigation Paths to recommendation

This section shows how the client can navigate through the hypermedia to access the recommendation resource:

```
resourceIndex
   recommendations
      recommendation
```

C.20.4.2.2 Read-only Elements for recommendation

Table C-80 lists the read-only elements for the recommendations resource.

Table C-80   Read-only Elements for recommendation

Element	Type	Description
score	Float	The overall score of this recommendation relative to the other recommendations in the list. This is the weighted sum of the component scores associated with each of the similarity URNs that comprise the recipe and is a floating point number between 0 and 1.
item	urn:oracle:webcenter:activitygraph:items:item	The recommended user, item, or portal.
componentScores	A list of componentScore elements	A list of the component scores associated with the different similarity URNs in the recipe for the recommendation. A component score may have a reason and a link that can be used to retrieve the common items with which the user and the recommended object have interacted.

C.20.4.3 urn:oracle:webcenter:activitygraph:items

Use this resource to identify the URI to use to retrieve (GET) objects that the source object and recommended object have in common. You can use this to determine the reasons why a particular object was recommended. The response from a GET operation includes each item in this collection of items, and each item includes links used to operate on that item.

C.20.4.3.1 Navigation Paths to items

This section shows how the client can navigate through the hypermedia to access this resource:

```
resourceIndex
    items
```

C.20.4.3.2 Supported Methods for items

The following methods are supported by this resource:

- GET

  - **request - Parameters:** `startIndex, itemsPerPage, utoken`

    For information about these common parameters, see Common Request Query Parameters.

    The following additional parameters are available:

    ◆ `similarityURN`—The URN of the similarity calculation to be used determine which objects are similar to the recommended object. For example `item-tag, gs-edit, user-connect`

    ◆ `srcClassURN`—The node class URN that identifies the type of the object that was used to request the recommendations. For example `WC.user, WC.group-document`.

    ◆ `srcObjectURN`—The URN of the object that was used to request the recommendations. For example `monty, 1000`.

    ◆ `trgClassURN`—The node class URN that identifies the type of the recommended object

    ◆ `trgObjectURN`—The object URN that provides a unique identifier for the recommended object

    ◆ `userCredentialClassURN`—The node class URN of the user exercising the REST API. The default value is `WC.user`. If you are integrating the activity graph engine with another application, this may need to be a different node class.

  - **response - body:** 1 or more items

For more information, see Templates.

C.20.4.3.3 Resource Types Linked to from items

Table C-81 lists the resource types that the client can link to from this resource.

Table C-81   Related Resource Types for items

rel	resourceType
self	urn:oracle:webcenter:activitygraph:items

C.20.4.4 urn:oracle:webcenter:activitygraph:items:item

Use this resource type to identify the URI to use to update (PUT) a recommendation to indicate user interest in the recommended object.

C.20.4.4.1 Navigation Paths to item

This section shows how the client can navigate through the hypermedia to access the item resource:

```
resourceIndex
   recommendations
      recommendation
         item
resourceIndex
   items
      item
```

C.20.4.4.2 Supported Methods for item

The following method is supported by this resource type:

- PUT

    – **Request—Body:** item

    – **Response—Body:** item

C.20.4.4.3 Writable Elements for item

Table C-82 lists the writable elements for this resource type.

Table C-82   Writable Elements for item

Element	Type	Required	Constraints	Description
status	String	Yes	interested not interested	Whether the user is interested in the object

C.20.4.4.4 Read-only Elements for item

Table C-83 lists the read-only elements for this resource type. Not all of these elements are available for all objects.

Table C-83   Read-only Elements for item

Element	Type	Description
classURN	String	Node class of the object
objectURN	String	Identifier for the object
name	String	Name of the object
description	String	Description of the object
modified	Date	Date on which the object was last updated
modifiedByUser	PersonReference	User information about the user who last updated the object, including GUID, ID, display name, and a link to the profile icon

Table C-83   (Cont.) Read-only Elements for item

Element	Type	Description
author	PersonReference	User information about the user who created the object, including GUID, ID, display name, and a link to the profile icon

C.20.4.4.5 Resource Types Linked to from item

Table C-84 lists the resource types that the client can link to from this resource.

Table C-84   Related Resource Types for item

rel	resourceType
self	urn:oracle:webcenter:activitygraph:items:item
via	resourceType of the underlying object

C.21 Using the Search REST APIs

WebCenter Portal provides REST APIs that let you build a customized search user interface. With Oracle SES 11.2.2.2, you can use the search REST APIs to read (GET) searches and facets. Facets are returned with search results and can be used to filter the results. You can specify keywords and the scope of the search; for example, suppose a search for the term 'webcenter' produces thousands of results. A user could use facets (such as author or last modified date) in the URL to get the results for particular facet values, like author=Karen and last-modified-date=this week.

This section includes the following topics:

- Search Entry Point

- Search Resource Type Taxonomy

- Security Considerations

- Search Resource Types

For an introduction to the REST APIs, see Introduction to the WebCenter Portal REST APIs.

C.21.1 Search Entry Point

Each REST service has a link element within the resource index that provides the entry point for that service. To find the entry point for search, find the link element with a `resourceType` of:

`urn:oracle:webcenter:searchcollection`

The corresponding `href` or `template` element provides the URI entry point. The client sends HTTP requests to this entry point to work with search.

See Also:

There are two entries for search in the resource index:

- `urn:oracle:webcenter:searchcollection` (This requires WebCenter Portal to be configured with Oracle SES 11.2.2.2.)

- `urn:oracle:webcenter:searchresults` (This requires WebCenter Portal to be configured with Oracle SES--any supported version.)

This topic describes only `urn:oracle:webcenter:searchcollection`. For detailed information about `urn:oracle:webcenter:searchresults`, see:

http://docs.oracle.com/cd/E28280_01/webcenter.1111/e10148/jpsdg_search.htm.

For more information about the resource index, see Using the Resource Index.

For more information about resource types, see Resource Type.

C.21.2 Search Resource Type Taxonomy

When the client has identified the entry point, it then can navigate through the resource type taxonomy to perform the required operations. For more information about the individual resource types, see the appropriate section in Resource Type.

The taxonomy for the search is:

```
urn:oracle:webcenter:searchcollection
   urn:oracle:webcenter:searchcollection:results
```

Beyond the service entry points, URL templates allow clients to pass query parameters to customize their requests and control the form of returned data.

Collection resources in the search resources support pagination (`startIndex` and `itemsPerPage`). The entry point returns all the results in `searchcollection` element. Query various components by specifying appropriate values in `dataType` argument.

Support for `searchTerms`, `data`, and `dataType` where `dataType` has one or more of the following values:

- `resultCount` - number of results satisfying the query
- `results` - actual results
- `facetCount` - number of facets for the query result
- `facets` - facets for the current results

Specify more than one `dataType` with comma separated values. For example:

```
http://host:port//rest/api/searchcollection?&dataType=results,facets
&facetParams=author:weblogic&utoken=FDX7xKPzzrnsbWRHNP-b-iUoWiJ4_w\*\\\\\\\\\*
```

C.21.2.1 Navigation Paths to results

This section shows how the client can navigate through the hypermedia to access this resource:

```
resourceindex
    searchcollection

resourceindex
    spaces
        spaces:resourceindex
            spaces:searchcollection
```

C.21.2.2 Supported Methods for results

The following methods are supported by this resource:

- `GET`

    - **request - body:** none, **Parameters**: `q`, `dataType`, `startIndex` (pagination), `itemsPerPage` (pagination), `data`, `facetParams`

    - **response - body:** search results, and optionally facets

where:

- `q={queryString}`

    Searches may be specified using the following format:

    `[[field1:[operand]][:]value1[;field2:operand:value2]]`

    Where multiple clauses are joined by an implicit `AND` and are syntactically separated by the ";" semicolon. Square brackets `[ ]` denote optional values. Each clause can be a keyword simple string with no ":" colon separator.

    The query string does not support anything other than a query keyword.

- `dataType` is the type of results

    For example, if you want only results, then `dataType=results` (or left empty). To include facets in the results, then `dataType=results,facets`.

- `data` is parameters for custom attributes

    For example:

    `data=numberofreplies,wc_tagsWords`

- `facetParams` is parameters for facet refinement

    Default set is Service ID, scope GUID, Tags, Author, Last Modified Date, Mimetype.

    For example, to refine by people, set the facet Service ID only to `oracle.webcenter.people`:

    `facetParams=Service%20ID:oracle.webcenter.people`

Note:

If you want to use the `facetParams` parameter, then the `dataType` parameter must include `facets`.

Example: Search REST Commands

```
http://examplehost:8888/rest/api/searchcollection?q=document&utoken={utoken}

http://examplehost:8888/rest/api/searchcollection?q={keyword}
&data=dDocName,dOriginalName&dataType=results,facets
&facetParams=Service%20ID:oracle.webcenter.people
```

For more information, see Templates.

Table C-85   Writable Elements for results

Element	Type	Description
author	String	Any user who has written to the result
resourceId	String	Unique identifier
serviceId	String	Service ID of the result; for example, oracle.webcenter.doclib
title	String	Title of result
description	String	Description of result
modified	Date	Last modified date
size	Number	Size of result
docid	String	Document identifier
created	Date	Original author of result
icon	String	Icon for result type
url	String	URL of result
mimetype	String	Mimetype of result
guid	String	GUID value of result
scope	String	GUID value of the portal
scopename	String	Portal name
resourceType	String	Type of result
language	String	Language of result
snippet	String	Snippet of result
modifier	String	Modifier of result
customattributes	any	List of any custom attributes specified in data

C.21.2.3 Resource Types Linked to From results

Table C-86 lists the resource types that the client can link to from this resource.

Table C-86   Related Resource Types for search

Table C-86 (Cont.) Related Resource Types for search

rel	resourceType
self	

C.21.3 Security Considerations

Authentication is required before using search REST API methods.

For general security considerations, see Security Considerations for WebCenter Portal REST APIs.

C.21.4 Search Resource Types

This section provides information about each resource type. It includes the following topics:

- urn:oracle:webcenter:searchcollection:links

- urn:oracle:webcenter:searchcollection:results

- urn:oracle:webcenter:searchcollection:facets

C.21.4.1 urn:oracle:webcenter:searchcollection:links

Use this resource type for template links and pagination links.

C.21.4.2 urn:oracle:webcenter:searchcollection:results

Use this resource type to identify the URI to use to read (GET) a query containing keywords and facets.

The request is represented by the URL and the response is a list of search results, each with metadata to help the end user choose an item to drill down, as well as cross links to the owning REST services, if available. If the owning REST service is unavailable, then an HREF link is provided.

The response XML can be paginated with standard URL parameters, and appropriate previous and next links provided along with a general template (with the query prepopulated) for the consuming application to do its own custom pagination.

C.21.4.3 urn:oracle:webcenter:searchcollection:facets

Use this resource type to identify the URI to use to read (GET) a query containing facets. Facets are returned with the query results and can be used to filter the results. Users can use these facets in the URL to get the results for particular facet values, like author=weblogic and last-modified-date=this week.

The request is represented by the URL and the response is a list of facets, each with metadata to help the end user choose an item to drill down, as well as cross links to the owning REST services, if available. If the owning REST service is unavailable, then an HREF link is provided.

The response XML can be paginated with standard URL parameters, and appropriate previous and next links provided along with a general template (with the query prepopulated) for the consuming application to do its own custom pagination.

D

Troubleshooting

This appendix provides troubleshooting information for WebCenter Portal assets.

This appendix contains the following topics:

- Troubleshooting WebCenter Portal Shared Library Deployment
- Troubleshooting Portlet Creation
- Troubleshooting Pagelets
- Troubleshooting OmniPortlet

D.1 Troubleshooting WebCenter Portal Shared Library Deployment

- **Changes are not available after deployment even though deployment successful**.

  WebCenter Portal always uses the latest shared library version. Check that the implementation version in `MANIFEST.MF` matches the implementation version displayed in the WebLogic Server Administration Console.

  For example, check the value in `PortalSharedLibrary/Application Sources/META-INF/MANIFEST.MF` is the same as that displayed in the Administration Console under **Deployments> extend.spaces.webapp>Overview**

- **"DeployerException: Task 9 failed" displays:**

  ```
  weblogic.Deployer$DeployerException:
  weblogic.deploy.api.tools.deployer.DeployerException: Task 9 failed:
  [Deployer:149117]deploy library com.mycompanyname.shared.lib
  [LibSpecVersion=11.1.1.2,LibImplVersion=11.1.1.2.5] on AdminServer,WC_Portal.
  ```

  This error occurs if the implementation version of the new deployment and the existing deployment are the same. Use the Administration Console to verify the current implementation version and if necessary change the version and redeploy the shared library.

- **"java.lang.IllegalArgumentException" displays**:

  Restart the managed server on which WebCenter Portal is deployed.

D.2 Troubleshooting Portlet Creation

This section includes the following topics:

- Cannot Access the Create Portlet Wizard
- Cannot Add the Portlet Functionality that I Want to the Portlet

D.2.1 Cannot Access the Create Portlet Wizard

In the New Gallery, I cannot find the Standards-based Java Portlet (JSR 286) option.

Problem

In the New Gallery, I cannot find the Standards-based Java Portlet (JSR 286) option.

Cause

Cause

The application in which you are trying to create the portlet was created using WebCenter Portal's Portal Framework template and therefore is not scoped for portlet creation.

Solution

Perform the following steps:

1. Try creating the portlet in a Portlet Producer Application or any application scoped for portlet creation (any application except for those built using WebCenter Portal's Portal Framework template).

2. In the New Gallery, click the All Technologies tab to list all available options regardless of the technology scope of the application.

D.2.2 Cannot Add the Portlet Functionality that I Want to the Portlet

I cannot find the option to add certain features, for example portlet events or public render parameters, to the portlet in the Create JSR 286 Java Portlet wizard.

Problem

I cannot find the option to add certain features, for example portlet events or public render parameters, to the portlet in the Create JSR 286 Java Portlet wizard.

Cause

The wizard does not provide the option to add certain features to portlets.

Solution

After you create a portlet using the Create JSR 286 Java Portlet wizard, edit the `portlet.xml` file using the Overview Editor to add advanced functionality. For more information, see Developing JSR 286 Java Portlets.

D.3 Troubleshooting Pagelets

Keep the following common configuration errors in mind when troubleshooting resources and pagelets. For details on configuring Pagelet Producer settings, see Section 23.2, "Configuring Pagelet Producer Settings."

- If you are using SSL, make sure both HTTP (nonsecure) and HTTPS (secure) ports are configured properly on the Transform page of the Pagelet Producer settings.

- If you are proxying external sites on a network that requires an HTTP proxy, you must configure the proxy URL on the Proxy page of the Pagelet Producer settings.

- Confirm that the `login_resource` and `pagelet_api` resources, created by default, are present and correctly configured.

- If you are proxying KD Browser or other CSP pagelets, make sure the image service URL is absolute and the CSP SOAP API URL is set correctly in the CSP page of the Pagelet Producer settings.

- Due to restrictions in the Sun Java Virtual Machine (JVM), pagelets cannot use HTTPS content where the underlying certificate uses MD2 signing algorithm.

For additional troubleshooting information, use logging. Pagelet Producer logs messages to the standard Oracle Diagnostic Logging facility. In Oracle WebLogic Server, that location is: `user-projects/domains/<domain>/servers/<server>/logs/<server>-diagnostic.log`. For details on configuring logging, see Section 23.2.2, "How to Configure Logging Settings." For more information on debugging pagelets, see the "Creating Pagelets with Oracle WebCenter Portal's Pagelet Producer" chapter in *Developing WebCenter Portal Assets and Custom Components with Oracle JDeveloper*.

D.4 Troubleshooting OmniPortlet

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 WebCenter Portal, and does not have access to the session information, you must expose XML files as PUBLIC in order for OmniPortlet to access them.

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`, meaning that OmniPortlet must be rendered within an IFRAME. 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`. Consequently, 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`.