# BEA WebLogic Personalization Server

# Guide to Creating Portals and Portlets

BEA WebLogic Personalization Server3.5
Document Edition 3.5.2
May 2002

## Copyright



## Restricted Rights Legend



## Trademarks or Service Marks



**Guide to Creating Portals and Portlets**

| Document Edition | Date | Software Version |
|---|---|---|
| 3.5.2 | May 2002 | BEA WebLogic Commerce Server 3.5 |

# Contents

## 4. Developing Portlets



## 5. Building a Custom Portal Step-by-Step

## 6. Advanced Portal Topics



## 7. Portal Management Database Schema

# About This Document

This document explains how to use the BEA WebLogic Personalization Server™ to create personalized applications for use in an e-commerce site.

This document includes the following topics:

- Chapter 1, “Introduction to Portals and Portlets,” provides an overview of portals and portlets as they are used on the Internet.
- Chapter 2, “Creating a Portal Using the Example Portal,”demonstrates how to use the Acme Portal to quick-start your portal development.
- Chapter 3, “Using the Portal Management Administration Tool,”describes how to create personalized application content on the Internet.
- Chapter 4, “Developing Portlets,” provides developers with in depth information about creating the portlets that are included in your portal.
- Chapter 5, “Building a Custom Portal Step-by-Step,” is a tutorial for building your own custom e-commerce portal.
- Chapter 6, “Advanced Portal Topics,”describes how to add the Webflow functionality of the WebLogic Commerce Server to your portal. This chapter also discusses deploying your application as a Web application.
- Chapter 7, “Portal Management Database Schema,” documents the portion of the database schema specific to portals.
- Chapter 8, “Portal Management JSP Tag Library Reference,” describes the JSP tags included with WebLogic Personalization Server for Portal Management. These JSP tags allow developers to create personalized applications without having to program using Java.

# What You Need to Know

This document is intended for business analysts, Web developers, and Web site administrators involved in setting up an e-commerce site using WebLogic Server™ and WebLogic Commerce Server and WebLogic Personalization Server. It assumes a familiarity with WebLogic Commerce Server and WebLogic Personalization Server, the WebLogic Server platform, J2EE Specifications, as well as the database management system that your organization uses.

# e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation or go directly to the "e-docs" Product Documentation page at http://e-docs.beasys.com.

# How to Print the Document

You can print a copy of this document from a Web browser, one file at a time, by using the File—>Print option on your Web browser.

A PDF version of this document is available on the WebLogic Personalization Server documentation Home page on the e-docs Web site (and also on the documentation CD). You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the WebLogic Personalization Server documentation Home page, click the PDF files button and select the document you want to print.

If you do not have the Adobe Acrobat Reader, you can get it for free from the Adobe Web site at http://www.adobe.com/.

# Contact Us!

Your feedback on the BEA WebLogic Personalization Server documentation is important to us. Send us e-mail at **docsupport@beasys.com** if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the WebLogic Personalization Server documentation.

In your e-mail message, please indicate the release number of the WebLogic Personalization Server documentation you are using.

If you have any questions about this version of BEA WebLogic Personalization Server, or if you have problems installing and running BEA WebLogic Personalization Server, contact BEA Customer Support through BEA WebSUPPORT at **www.beasys.com**. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

# Documentation Conventions

The following documentation conventions are used throughout this document.

| Convention | Item |
|---|---|
| **boldface text** | Indicates terms defined in the glossary. |
| Ctrl+Tab | Indicates that you must press two or more keys simultaneously. |

| Convention | Item |
|---|---|
| *italics* | Indicates emphasis or book titles. |
| `monospace text` | Indicates code samples, commands and their options, data structures and their members, data types, directories, and filenames and their extensions. Monospace text also indicates text that you must enter from the keyboard.<br>*Examples*:<br>`#include <iostream.h> void main ( ) the pointer psz`<br>`chmod u+w *`<br>`\tux\data\ap`<br>`.doc`<br>`tux.doc`<br>`BITMAP`<br>`float` |
| **`monospace boldface text`** | Identifies significant words in code.<br>*Example*:<br>`void` **`commit`** `( )` |
| *`monospace italic text`* | Identifies variables in code.<br>*Example*:<br>`String` *`expr`* |
| UPPERCASE TEXT | Indicates device names, environment variables, and logical operators.<br>*Example*s:<br>LPT1<br>SIGNON<br>OR |
| `{ }` | Indicates a set of choices in a syntax line. The braces themselves should never be typed. |
| `[ ]` | Indicates optional items in a syntax line. The brackets themselves should never be typed.<br>*Example*:<br>`buildobjclient [-v] [-o name ] [-f` *`file-list`*`]...`<br>`[-l` *`file-list`*`]...` |

| Convention | Item |
|---|---|
| \| | Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed. |
| . . . | Indicates one of the following in a command line:<br>■ That an argument can be repeated several times in a command line<br>■ That the statement omits additional optional arguments<br>■ That you can enter additional parameters, values, or other information<br>The ellipsis itself should never be typed.<br>*Example*:<br>`buildobjclient [-v] [-o name ] [-f file-list]...`<br>`[-l file-list]...` |
| .<br>.<br>. | Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed. |

CHAPTER

# 1 Introduction to Portals and Portlets

Internet portals are a key part of many e-commerce applications. Portals provide an entry point to the Internet as well as value-added services such as searching and application integration. The Portal Framework provided with the BEA WebLogic Personalization Server allows you to quickly assemble both Business-to-Consumer and Business-to-Business portals that require personalized application content on the Internet.

This topic includes the following sections:

- What Is a Portal?
- What Is a Portlet?
- JSP Tags

## What Is a Portal?

Intelligent portals can act as tour guides to points of interest, tailored for individual user preferences. There are portals that concentrate on collecting and delivering specialized areas of information such as stock trading and finances, emerging technologies, or corporate information. For example, www.boston.com is a specialized news portal and www.schwab.com is a specialized financial portal. Other *megaportals* provide general channels of information such as health, weather, sports, news, e-mail services, chat rooms, news groups, and so on.

Internet portals are an efficient way to exchange large volumes of information with large groups of people. From the users' perspective, most portals are organized as a hierarchical Web site where the main page provides links to a set of pages that provide a more detailed view of the data.

*Static portals*, like many corporate home pages, provide a standard set of information to everyone who visits the site. In contrast, *dynamic personalized portals*, where the information presented may differ based on who is viewing the portal, represent a far more efficient and targeted way to do business. Well-known examples of dynamic portals include www.amazon.com, www.ebay.com, www.excite.com, and my.yahoo.com. With the Portal Framework, you can quickly build powerful, dynamic portals like these, as well as static ones.

# The Portal Framework

The Portal Framework provides a collection of prebuilt JSP pages that provide the core functionality for portals. They are distributed as a Web application, and are located in `%WL_COMMERCE_HOME%/config/wlcsDomain/applications/wlcsApp/examplepportal/portals/repository`. The Portal Framework home page is located at http://`localhost:portnumber`/.

When using the framework, your pages will have a common layout. In this layout, a page's real estate will be divided into three main areas: a header, a content area, and a footer. The header resides at the top of the page and typically contains a full-sized logo for the site plus some navigation features. Placeholders for advertising can also be put into the header. The footer resides at the bottom of the page and typically contains legal notices, copyright information, and a small logo. The middle section, the content area, contains any number of small independent components called portlets. The JSPs included in the Portal Framework manage the layout of these portlets on the page.

The example portal demonstrates how the portal framework functions. The JSP source code in the example portal contains the HTML that controls the portal layout. Within an HTML table, portlets are dynamically included with `<jsp:include>` in cells. While the standard portal can handle up to three columns, you can easily add more columns by including a non-portlet column on either the left or right side of the table. For instance, the portlets could be in the three right-hand columns and behave as in the standard example. The left-hand column could be it's own JSP and might be a navigation bar. You can place portlets in any position as long as it is legal in HTML. To persist the portlet row-column position, you can use the API (for example, see the `PortletJspBean` in the *Javadoc*). Rewriting the example portal layout require some

understanding of HTML and good Java programing skills. You can also make similar changes for headers and footers because they are also JSPs with HTML formatting. This means you can make the portal look like anything that you want.

# What Is a Portlet?

A *portlet* is a highly focused channel of information served up by a portal. A portal can contain many of these information channels. For example, an online retail portal could provide a variety of interactive merchandise portlets, each presenting a different specialty category such as mystery books, classical music CDs, and baseball memorabilia.

Unlike a static portal page, deploying portlets with WebLogic Personalization Server gives our online retailer the ability to dynamically respond to customers based on user profiles. With this technology, not only can the retailer provide dynamic content, but also the customer can easily select and arrange their e-commerce portlets.

For example, a returning customer, Samantha, who loves mystery novels and ghost stories could select the “mystery” portlet as central to her standard view of the retailer’s home page. This would be done by means of an edit page made available by the retailer. At the same time, the retailer could determine that Sam’s purchase choices and portlet selections convey a taste for the unexplained. The Personalization Server lets you automatically generate *responses* to user profiles. A response *could* be the delivery of specialized information via a portlet. In the case of the mystery hound, our fictitious retailer could offer up recommendations about the latest thrillers and *whodunnits* on video.

## Personalizing a Portal

Personalization allows you to customize your portals and portlets to serve a specific audience and purpose. The Portal Framework supports three levels of personalization, all of which can be administered with Web-based tools. The three levels of personalization are as follow:

- Portal

- Group
- User

Personalization includes Web-based forms for adding and removing portal content, editing the content layout, and customizing the portal content color schemes. The user personalization information includes user information and general user preferences.

# JSP Tags

Several JSP tags for Portal Management are included with WebLogic Personalization Server (Table 1-1). JSP tags allow developers to create personalized applications without having to program using Java.

For additional information see Chapter 8, "Portal Management JSP Tag Library Reference," in this guide.

**Table 1-1 Java Server Page (JSP) Tags for Portal Management Overview**

| | | |
|---|---|---|
| **Portal Management** | `<esp:eval>` | Evaluates a conditional attribute of a portlet. An example of a conditional attribute is `isMinimizeable`. |
| | `<esp:get>` | Retrieves a String attribute of a portlet. |
| | `<esp:getGroupsForPortal>` | Retrieves the names of the groups associated with a Portal. |
| | `<esp:monitorSession>` | Disallows access to a page if the session is not valid or if the user has not logged in. |
| | `<esp:portalManager>` | Provides the ability to do `create`, `get`, `getColumnInfo`, `update`, and `remove` actions on a Portal object. |
| | `<esp:portletManager>` | Provides the ability to do `create`, `get`, `getArranged`, `update`, and `remove` actions on a Portlet object. |

**Table 1-1 Java Server Page (JSP) Tags for Portal Management Overview (Continued)**

| | |
|---|---|
| `<esp:props>` | Used to get a property from the Portal Properties bean, whose deployment descriptor contains default values used by the Portal Administration Tool. |

# 2 Creating a Portal Using the Example Portal

This chapter shows you how to use the Acme Portal to quick start your portal development.

WebLogic Personalization Server includes the Acme Demo Portal, a complete demo portal that is set up for you and ready to run. This demo portal uses the Cloudscape Database Management System (DBMS) to store the Portal Demo data. Use the Acme Demo Portal to quick start your portal development.

**Note:** The directory name for the Acme Portal is `exampleportal`. Throughout the documentation "Acme Portal" and "example portal" are used interchangeably.

This topic includes the following sections:

- Introducing the Acme Portal
- Starting the Acme Portal
- Building the Acme Portal Components
- Creating Portlets for Your Demo Portal
- Associating Portlets with Your Demo Portal
- Editing Your Demo Portal Layout
- Editing Your Demo Portal Color Scheme
- Testing Your Demo Portal

# Introducing the Acme Portal

When you install the WebLogic Personalization Server, a complete demo portal is set up for you and ready to run. This ready-made demo portal uses the Cloudscape Database Management System (DBMS) to store the Acme Portal data. The demo is ready to run with the Cloudscape database immediately after you complete the WebLogic Personalization Server installation.

**Note:** The Cloudscape database is included with WebLogic Server under a limited evaluation license. Because of the limitations of the Cloudscape database, other databases should be considered. The database stores all of the portal framework information needed to support the portal components.

The Acme Portal includes the following:

- Portal page JSP templates—header, footer, and portal content layout JSP pages
- Sample portlet applications including portlet JSP pages
- A complete set of end user portal personalization tools, including:
  - User login and new user registration Web forms
  - Change-password and forgot-password Web forms
  - End user personalization tools for customizing portal content
  - Help pages

All of the JSP pages for the Acme Portal are located in the following installed product directory: application/portals/repository.

# Starting the Acme Portal

To start the Portal Demo:

1. Start the WebLogic server by executing the StartCommerce command file in your installation directory.
2. Open a Web browser window.

3. In your Web browser, enter the following demo portal page URL, `http://hostname:port/exampleportal` where `hostname` is the name of the host running your WebLogic Server, `port` is the port number at which the WebLogic Server is listening for requests, and `exampleportal` is the name of the property set.

   Example: http://mybigbox:7501/exampleportal

You can now use the Portal Administration Tool to view the Demo Portal or assemble your own portal, as described in Chapter 3, "Using the Portal Management Administration Tool."

**Notes:** Before using the Portal Administration Tool to assemble the Acme Portal, you must install and set up the WebLogic Personalization Server software. For more information, see the *Installation Guide*.

# Building the Acme Portal Components

To create the Acme Portal definition:

1. Click Create in the Portals banner of the Portal Administration Tool Home page to display the Create a New Portal tool.
2. Enter the following information in the appropriate fields:

**Table 2-1 Sample Data**

| Field Name | Data |
|---|---|
| Portal Name | Acme Portal |
| Header URL | `header.jsp` |
| Content URL | `portalcontent.jsp` |
| Footer URL | `footer.jsp` |
| Number of columns | 3 |
| Suspend | Defaults to `false`. Set to `true` to suspend the portal during maintenance. |

**Table 2-1 Sample Data (Continued)**

| Field Name | Data |
|---|---|
| Suspended URL | Enter `suspended.jsp` if you want to display the "under maintenance" URL during maintenance. |

3. Click Create.

   If the portal was successfully created, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.

4. Click Back to return to the home page. The name of the new portal, "Acme Portal," displays in the Portals section of the home page.

# Creating Portlets for Your Demo Portal

You create portlets in the administration tool by associating a URL with each portlet component. When a portlet is created, it is not automatically associated with a portal. You must add the portlets to the demo portal later from the portal view-page.

The Acme Portal includes six portlet applications that you can assemble with the Portal Administration Tool.

To create the demo portlets:

1. Click Create in the Portlets banner of the Portal Administration Tool Home page to display the Create a New Portlet tool.

2. To create the first of the demo portlets, My Bookmarks, enter the following information in the appropriate fields:

**Table 2-2 Creating MyBookmarks Portlet**

| Portlet Name | Data |
|---|---|
| Bookmarks | **Portlet Name:** Bookmarks |
| | **Content URL:** `portlets/bookmarks.jsp` |
| | **Editable:** Select the check box |

**Table 2-2 Creating MyBookmarks Portlet (Continued)**

| Portlet Name | Data |
|---|---|
| | **Edit URL:** `portlets/bookmarks_edit.jsp` |
| | **Maximizable:** Select the check box |
| | **Icon URL:** `portlets/images/pt_bookmark.gif` |
| | **Login Required:** Select the check box |
| | **Titlebar URL**: Enter a URL to display as the portlet titlebar. It can be a JSP or HTML fragment. |
| | **Mandatory:** A mandatory portlet is one that is always available and visible. |

3. Click Create.

   If the portlet was successfully created, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.

4. Follow steps 2–3 above to create the remaining five portlets. For each portlet, enter the following information in the appropriate fields:

**Table 2-3 Creating Other Portlets**

| Portlet Name | Data |
|---|---|
| My Dictionary | **Portlet Name:** My Dictionary |
| | **Content URL:** `portlets/dictionary.jsp` |
| | **Icon URL:** `portlets/images/pt_dictionary.gif` |
| | **Minimizable:** Select the check box |
| My To Do List | **Portlet Name:** My To Do List |
| | **Content URL:** `portlets/mytodo.jsp` |
| | **Editable:** Select the check box |
| | **Edit URL:** `portlets/mytodo_edit.jsp` |
| | **Maximizable:** Select the check box |
| | **Icon URL:** `portlets/images/pt_my_list.gif` |
| | **Minimizable:** Select the check box |
| | **Floatable:** Select the check box |
| | **Login Required:** Select the check box |
| My Group To Do List | **Portlet Name:** My Group To Do List |
| | **Content URL:** `portlets/grouptodo.jsp` |
| | **Banner URL:** `portlets/grouptodobanner.jsp` |
| | **Editable:** Select the check box |
| | **Edit URL:** `portlets/grouptodo_edit.jsp` |
| | **Maximizable:** Select the check box |
| | **Icon URL:** `portlets/images/pt_group_list.gif` |
| | **Minimizable:** select the check box |
| | **Floatable:** Select the check box |

**Table 2-3 Creating Other Portlets (Continued)**

| Portlet Name | Data |
|---|---|
| | **Login Required:** select the check box |
| Stock Quote | **Portlet Name:** Stock Quote |
| | **Content URL:** `portlets/quote.jsp` |
| | **Icon URL:** `portlets/images/pt_quote.gif` |
| | **Minimizable:** Select the check box |
| Search | **Portlet Name:** Search |
| | **Content URL:** `portlets/search.jsp` |
| | **Icon URL:** `portlets/images/pt_search.gif` |
| | **Minimizable:** Select the check box |
| News Index | **Portlet Name:** News Index |
| | **Content URL:** `portlets/new_index.jsp` |
| News Reader | **Portlet Name:**News Reader |
| | **Content URL:** `portlets/news_viewer.jsp` |
| | **Titlebar:** `content_titlebar.jsp` |

5. Click Back to return to the home page. The six new portlet names appear in the Portlets section.

# Associating Portlets with Your Demo Portal

After creating a portal, you can associate portlets to it. You can also personalize the portal's layout and color scheme, and make changes to the definition. For more information on editing a portal with the Portal Administration Tool, see “Editing Portals” on page 3-16 in this guide.

You choose which portlets are available to a portal by adding and removing them from the system's list of established portlets. From the narrowed list of portlets you associate with a portal, group and end users further define which portlets they want available and visible on their personalized portal page.

To associate the six portlets you just created with the demo portal:

1. On the Portal Administration Tool Home page, click the Demo Portal title link in the Portals section of the screen. The Acme Portal view-page displays.
2. On the portal view-page, click +/- in the Associated Portlets banner to display the Add or Remove Portlets tool.
3. To add a portlet to the portal, select Avail. The portlet is associated with the portal. It does not display on the portal page until it is made visible by you, the Group Administrator, or the end user.
4. To make a portlet visible, select Visible. The portlet is associated with the portal and now displays on the portal page.
5. To remove a portlet from the portal, select Unavail. The portlet becomes disassociated with the portal and unavailable to new groups and end users (including anonymous users). However, if the portlet has already been personalized at a group or user level, it remains associated with those levels.
6. Click Save.

   If the changes were successfully made, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.
7. Click Back to return to the Demo Portal view-page. Available portlets appear in the Associated Portlets section of the screen with a gray background. Visible portlets are marked with an X.

# Editing Your Demo Portal Layout

You can move a portal's associated portlets left and right between columns and up and down within columns depending on the column layout you selected when you created the portal. You can also change the percentage of the portal page that each column occupies in all portals except group portals. Group Administrators and end users can further personalize the portal layout.

To edit the layout of portlets in the demo portal:

1. On the Demo Portal view-page, click Edit in the Layout banner to display the Edit Portal Layout tool. This layout tool shows each portal column, its span percentage, and the portlets that display within those columns.
2. Select the portlet you want to move by clicking on it. The portlet name is highlighted.
3. Click an arrow to move the portlet up or down within a column, or right or left between columns.

To change the column spans of a portal layout:

1. Click in the percentage field associated with a column and enter a new percentage. The sum of all column spans should equal 100%.
2. When you finish editing the portal layout, click Save.

   If the changes were successfully made, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.
3. Click Back to return to the Demo Portal view-page. A table in the Portal Layout section lists the portlets as you arranged them within each column.

# Editing Your Demo Portal Color Scheme

You can edit the overall appearance of a portal by changing its background color as well as the portlet's component colors, title colors, and border appearance.

To edit the demo portal colors:

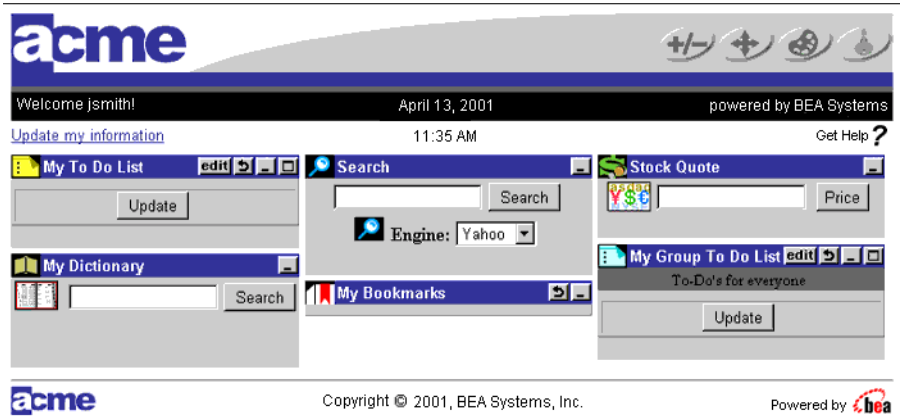
1. On the Demo Portal view-page, click Edit in the Colors banner to display the Edit Color Schemes tool. This tool provides five preset color schemes and a Custom Scheme tool.
2. In the Portlet Color Schemes section of the screen, select a preset color scheme or the custom color scheme. If you selected the custom color scheme, enter a hex color code in each text field, or click the color palette icon to select a color for each field from the Color Picker.
3. Select On to display portlet borders, or Off to omit portlet borders.
4. Select Black, White, or Other to choose the color of the text that will display in the portlet titlebar. If you selected Other, enter a hex color code in the text field, or click the color palette icon to select a color from the Color Picker.
5. In the Portal Background Color section of the screen, select Gray, White, or Other to choose a background color for the entire portal page. If you selected Other, enter a hex color code in the text field, or click the color palette icon to select a color from the Color Picker.
6. To preview your color selections, click the "Click here to save changes and preview colors" link. Your color changes will be saved and the Edit Color Schemes tool will redisplay the example portlet at the bottom of the screen to reflect your color preferences.
7. To save your color preferences without previewing them, click Save. The portal view-page displays the new colors associated with the portal in the Colors section of the screen.
8. To revert the portal appearance to the BEA original color scheme, click Restore Defaults. The portal view-page displays the default colors associated with the portal in the Colors section of the screen.

# Testing Your Demo Portal

Once your portal is operational, you should test it to verify that all the associated portlets are available and visible as you specified them, and that your portal displays the correct color scheme and layout.

Figure 2-1 shows the Acme Portal as it displays with the BEA default color scheme and layout to a registered user named jsmith.

**Figure 2-1 Acme Portal**




To test the demo portal:

1. In a Web browser, enter the Flow Manager URL: http://host:port/exampleportal

   The default portal home page should display all visible portlets and should reflect your default color and layout preferences.

2. To test end user personalization options, sign on to your portal by clicking the Sign On icon in the upper right corner of the home page.

   If you have not created a user profile, you can do so by following the registration wizard. If you have created a profile, enter your username and password, and click Sign On.

You can now use the personalization tools to customize the portal's color, layout, and visible portlets.

# 3 Using the Portal Management Administration Tool

The Portal Administration Tool contains a complete set of functions that enable portal administrators to easily create the components of a portal page and personalize the portal's content, layout, and appearance.

This topic includes the following sections:

- Setting Up
  - Logging On to the Administration Tool
  - Configuring the Flow Manager to Control Portal Access
  - Creating a Portal Web Site Directory
- Using the Portal Administration Tool
  - Administering Portlets
  - Administering Portals
  - Administering Portal Groups
- Where to Get More Information

# Setting Up

To properly create and administer a portal using the Portal Administration Tool, you should know how to configure and run the WebLogic Server, and set up database connections.

To create and administer a portal, first you must install and set up the WebLogic Commerce Server and WebLogic Personalization Server software. Then, refer to the following sections in this document to complete these setup and initialization tasks:



## Logging On to the Administration Tool

To log on to the Portal Administration Tool:

1. Start the WebLogic Server configured for portal use. Start the WebLogic Personalization Server.
2. Access `http://hostname:port/tools` in your Web browser where `hostname` is the name of the host running your WebLogic Server, `port` is the port number at which the WebLogic Server is listening for requests, and `tools` is the name of the deployed Web application.

   A dialog box appears and prompts you to enter a username and password.
3. Enter the username `administrator` and the password `password`.

**Note:** We recommend that after installation you immediately create a new password for the administrator.

4. Click OK to display the Administration Tool Home page, then click the Portal Administration page icon.

# Configuring the Flow Manager to Control Portal Access

The Flow Manager controls user access to your portal. The Flow Manager receives all incoming HTTP requests and dispatches each request to the appropriate destination URL. For more information about the Flow Manager, see the topic “Flow Manager” in the chapter “Foundation Classes and Utilities” in the *Guide to Building Personalized Applications*.

## _DEFAULT_PORTAL_INIT property set

To configure the Flow Manager, create a portal property set based on a default `APPLICATION_INIT` property set.

To create a new property set:

1. Open the Administration Tools Home page. Click the Property Set Management icon to open the Property Set Management screen.
2. From the main Property Set Management screen, click Create.
3. Name the new property set you are creating (100 character maximum). The name of the property set should be the same as the name you will use to create the portal, or the name you will use to access the application.
4. Enter a description of the property set (255 character maximum).
5. From the Copy Properties From drop-down list, select
   `APPLICATION_INIT._DEFAULT_PORTAL_INIT` (for a portal)
   or
   `APPLICATION_INIT._DEFAULT_APP_INIT` (for a non-portal application).
6. From the Property Set Type drop-down list, select Application Init.
7. Click Create.
8. At the top of the page, in red, you will see the message “Property Set creation was successful.” (Or, you will see an error message indicating why the property set was not created.)
9. Click Back to return to the main Property Set Management screen.

## Valid Flow Manager Parameters

To set parameters for your portal or application:

1. From the Property Set Management Home page, under the Application Initialization Property Sets heading, click the name of the property set you just created.
2. A Property Set page comes up, allowing you to set parameters.
3. **Note:** For non-portal applications, skip this step.
   To edit the portal name, click the Edit button to the right of the "portal name" property. Change the default value from `UNKNOWN` to the name of your portal, as you created it in Portal Management.
4. Edit the `destinationdeterminer` property. Either accept the default, or edit to provide your own implementation of these classes.
5. Edit the `destinationhandler` property. Either accept the default, or edit to provide your own implementation of these classes.
6. Customize any other properties you choose. For information about customizing properties in portals, see Chapter 5, "Building a Custom Portal Step-by-Step," in this guide.
7. When you have finished setting properties, click the Finished button at the bottom of the page.

The table below lists valid parameters for the Flow Manager servlet.

**Table 3-1 Valid Flow Manager Servlet Parameters**

| Parameter Name | Required | Description |
|---|---|---|
| `allowautologin` | No | Determines whether a client with valid cookies can automatically login. The default is `false`. |

**Table 3-1 Valid Flow Manager Servlet Parameters (Continued)**

| Parameter Name | Required | Description |
|---|---|---|
| `defaultdest` | Yes | The default destination page JSP if there is not a valid session for the user. (This page is qualified from your Web application's `DocumentRoot`.)<br>To display a default portal page for anonymous users, use:<br>`defaultdest=/portals/myPortal/portal.jsp`<br>or<br>to force anonymous users to the login page instead of the portal page use:<br>`defaultdest=/portals/myPortal/_userlogin.jsp` |
| `destinationdeterminer` | Yes | Used by the Flow Manager to determine JSP page navigation. |
| `destinationhandler` | Yes | Used by the Flow Manager to determine JSP page navigation. |
| `groupName` | Yes | The default group name for this portal instance. (When new users register, they are added to this group. This parameter allows you to register two Flow Managers that are alike except for the groups that they service.) This value defaults to everyone. |
| `homepage` | Yes | The home page JSP returned by the system in auto-login or from the portal home button. (This page is qualified from your Web application's `DocumentRoot`.) Example:<br>`homepage=/portals/myPortal/portal.jsp` |
| `portalname` | Yes | The name given to the portal you created in the Portal Administration Tool. Example: Acme Demo Portal |
| `repositorydir` | Yes | Location of default files (gifs, JSP, etc.). |
| `refreshworkingdir` | No | Number of seconds, defaults to -1, which means check every time. |
| `sessioncomparator` | Yes | How to determine if the session is valid. |

**Table 3-1 Valid Flow Manager Servlet Parameters (Continued)**

| Parameter Name | Required | Description |
|---|---|---|
| `timeout` | No | Timeout for the cookies or session valued in seconds and defaulting to (-1).<br>If set to (-1), the cookies expire upon exiting the browser. If cookies are disabled, the session invalidates upon browser exit. To retain user login information between browser sessions, set the timeout to a large positive number, such as 999999, and set autologin=`true`. |
| `ttl` | Yes | Number of seconds between Flow Manager reexaming this property set. |
| `workingdir` | Yes | The working directory JSP for the portal implementation that tells the portal framework where to find your portal pages and the WebLogic Commerce Server and WebLogic Personalization Server pages. (This page is qualified from your Web application's `DocumentRoot`.) Example: `workingdir=/portals/myPortal/` |

# Creating a Portal Web Site Directory

As a final step, create a Web site directory for your portal pages under the `wlcsApp` directory.

Create a new Web application directory named `'myPortal'` by copying and pasting the `exampleportal` directory and renaming it `myPortal` in the `wlcsApp` directory. For more information, refer to the section "Deploying New Portals as Web Applications" on page 6-1 in this guide.

Table 3-2 shows the structure of the Acme Portal. The same structure can be used for any new portal.

**Table 3-2 Acme Demo Portal Directory and Subdirectories**

| Directory | Description |
|---|---|
| `/portals/repository` | The portal root directory that contains pages such as `header.jsp`, `footer.jsp`, and `portalcontent.jsp`. |
| `/portals/repository/images` | A directory of images that support your portal and WebLogic Commerce Server and WebLogic Personalization Server components. |
| `/portals/repository/portlets` | The directory of all portal JSP and HTML pages and the WebLogic Commerce Server and WebLogic Personalization Server sample portlet applications. |
| `/portals/repository/portlets/images` | A directory of images that supports your portlets and WebLogic Commerce Server and WebLogic Personalization Server portlets. |

# Using the Portal Administration Tool

**Figure 3-1 Administration Tool Home Page**

Now that you have access to the Portal Administration Tool, you can use it to administer portlets, portals, and business-to-business portal groups. Administrative functions available in the tool include:

- Creating, editing, and deleting portals
- Creating, editing, and deleting portlets
- Personalizing a portal's content, layout, and color scheme at the portal and group levels
- Testing your portal

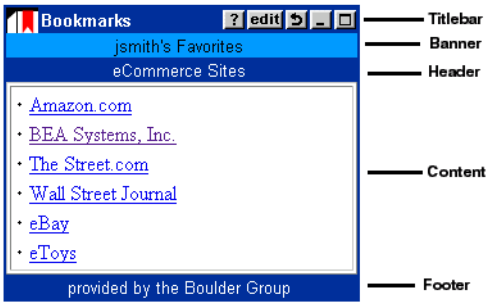
# Administering Portlets

To the portal framework, a portlet is a JSP page that knows how to retrieve specialized content and display it in the portal application. To users, a portlet is one of many content modules on a portal page that can be personalized to reflect appearance, content, and layout preferences. Once a portlet is created in the Portal Administration Tool, it can be associated with multiple portals.

You can create, edit, and delete portlets in the Portlets section of the Portal Administration Tool Home page. All screens in the administration tool related to portlet functions are color-coded with teal banners and command buttons. Screens related to portal functions display tan banners and command buttons.

A portlet includes two required components, a titlebar and content area, and several optional components including the banner, header, footer, edit URL, alternate header, alternate footer, maximized URL, and help URL as shown in the following graphic:

**Figure 3-2 Portlet Application Decomposed by Components**



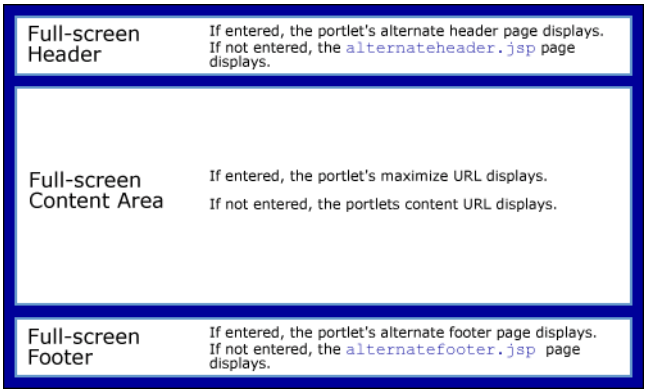

You can define each portlet application to include any of the following attributes:

- **Editable**—enables the user to customize a portlet's content. For example, in a stock portfolio portlet application users can click the Edit icon on the portlet titlebar to access a page that enables them to add or remove stock symbols. If you select this attribute, you must provide an Edit URL.
- **Maximizable**—allows the portlet to be viewed full screen in the browser window. This enables you to provide additional portlet content in the Maximized URL.

  The full screen page uses:

  - **An alternate header**—if no alternate header is specified, the framework uses the default alternate header.
  - **A maximize URL**—if no maximize URL is specified, the framework uses the portlet content area URL as a default.
  - **An alternate footer**—if no alternate footer is specified, the framework uses the default alternate footer.
- When the user clicks the edit or maximize icons in a portlet, the portal framework calls upon its `fullscreen.jsp` page to display in full screen mode. The diagram below explains how the `fullscreen.jsp` page determines which content to display.

**Figure 3-3 Content Display Criteria**




- **Floatable**—Allows the portlet to float on top of the portal screen in a separate browser window. This attribute uses the same header and footer rules as the maximized URL, but displays the content URL instead of the maximized URL.
- **Minimizable**—Reduces the portlet display to the titlebar to minimize the amount of space the portlet occupies on the portal page.
- **Helpable**—Provides a Help icon in the portlet title bar that users can click to access a URL that assists them with the portlet application. If you select this attribute, you must provide a Help URL.
- **Login Required**—Requires the user to be logged on to the portal to view the portlet. For example, if your portal contains a portlet that displays a user's favorite bookmarks, the user must be logged on before the Bookmark's portlet is visible on the portal screen. This attribute helps maintain a secure portal and allows users to retrieve personalized information.
- **Mandatory** —A portlet can now be personalized to be mandatory. A mandatory portlet is one that is always available and visible. The portlet can be made mandatory at the definition, portal personalization, and group personalization levels.
- **Titlebar URL** — A URL can display as the portlet titlebar. It can be a JSP or HTML fragment.

## Creating Portlets

Before you use the Portal Administration Tool to create a portlet, place all your portlet application files in the following directory:

```
%WL_COMMERCE_HOME%/config/wlcsDomain/applications/wlcsApp/examples
portal/portals/repository/portlets
```

You create a portlet in the administration tool by creating a portlet definition entity (referred to in this document as a portlet) and associating portlet JSP URLs that have been created by a portlet developer with the portlet entity. When a portlet is created, it is not automatically associated with a portal. You need to add portlets to a portal later from the portal-view page.

To create a portlet:

1. On the Portal Administration Tool Home page, click Create in the Portlets banner. The Create a New Portlet tool displays.
2. Enter the appropriate information in the following required fields:
   - *Portlet Name*—any combination of numbers and letters will be accepted in this field.
   - *Content URL*—enter a URL relative to your portal `workingdir`.
3. If desired, enter the appropriate information in the following optional fields:
   - *Header URL*—enter a URL to display as the portlet header. It can be a JSP or HTML fragment.
   - *Footer URL*—enter a URL to display as the portlet footer. It can be a JSP or HTML fragment.
   - *Titlebar URL*—enter a URL to display as the portlet titlebar. It can be a JSP or HTML fragment.
   - *Banner URL*—enter a URL to display as the portlet banner under the portlet titlebar. It can be a JSP or HTML fragment. The following shows a sample banner JSP page:

     ```
     <%@ page extends="com.beasys.portal.admin.PortalJspBase"%>

     <%@ page
     import="com.beasys.portal.tags.PortalTagConstants"%>
     ```

```
<center>

<font size=-1>To Do's for

<%@
(String)getSessionValue(PortalTagConstants.PORTAL_GROUP,requ
est)%></font>

</center>
```

- *Mandatory*—a portlet can now be personalized to be mandatory. A mandatory portlet is one that is always available and visible. The portlet can be made mandatory at the definition, portal personalization, and group personalization levels.
- *Alternate Header URL*—enter a URL to display as a Web page header when the portlet is floated or maximized. If no alternate header exists, the portal framework uses a default called `alternateheader.jsp`.
- *Alternate Footer URL*—enter a URL to display as a Web page footer when the portlet is floated or maximized. If no alternate footer exists, the portal framework uses a default called `alternatefooter.jsp`.
- *Editable*—select the check box to enable users to edit a portlet's content. An Edit icon displays in the portlet titlebar. The attribute default is deselected.
- *Edit URL*—if you selected the Editable check box, enter a URL that enables the user to edit the portlet content.
- *Maximizable*—select the check box to enable users to maximize the portlet in the current browser window. A Maximize icon displays in the portlet titlebar. The attribute default is deselected.
- *Maximized URL*—if you selected the Maximizable check box, enter a URL for the content area of the maximized page. The default URL is your portlet content area URL.
- *Helpable*—select the check box to enable users to access a help screen. A Help icon displays in the portlet titlebar. The attribute default is deselected.
- *Help URL*—if you selected the Helpable check box, enter a URL that opens a help topic related to the portlet.
- *Icon URL*—enter a URL to display an icon (GIF) on the left side of the portlet titlebar. This image should be 27 pixels wide by 20 pixels high with 2 pixels of transparency on the right.

- *Minimizable*—select the check box to enable users to minimize the portlet in the portal screen. A Minimize icon displays in the portlet titlebar. The attribute default is deselected.
- *Floatable*—select the check box to enable users to float the portlet in a new browser window. A Float icon displays in the portlet titlebar. The attribute default is deselected.
- *Login Required*—select the check box to require a user to be logged on to the portal to view the portlet. The attribute default is deselected.

4. Click Create.

   If the portlet was successfully created, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.

5. Click Back to return to the home page. The new portlet name displays under the Portlets banner.

## Editing Portlets

After creating a portlet, you can redefine it at any time by adding or removing attributes.

To edit a portlet:

1. On the home page, click a portlet title link to display the Edit Properties tool. The name of the portlet you selected to edit displays at the top of the screen.
2. Enter the appropriate changes.
3. Click Save.

   If the changes were successfully made, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.

4. Click Back to return to the home page.

## Deleting Portlets

You can delete portlets that you no longer need. However, you must first remove (mark as unavailable) the portlet from any portals it is associated with.

To delete a portlet:

1. On the home page, click Delete in the Portlets banner. The Delete a Portlet tool displays.
2. Select the portlet from the Portlet Name drop-down list.
3. Click Delete. A confirmation window displays.
4. Click Ok to confirm your deletion.
5. Click Back to return to the home page. The portlet name is no longer listed under the Portlets banner.

# Administering Portals

You can create, edit, or delete portals from the Portals section of the Portal Administration Tool Home page. All screens in the administration tool related to portal functions are color-coded with tan banners and command buttons. Screens related to portlet functions display teal banners and command buttons.

## Creating Portals

To create a new portal:

1. On the Portal Administration Tool Home page, click Create in the Portals banner to display the Create a New Portal tool.
2. Complete the following required fields:
   - *Portal Name*—any combination of numbers and letters will be accepted in this field.
   - *Content URL*—enter a portal content JSP relative to `workingdir`.
   - *Number of Content Columns*—enter 1, 2 or 3.
3. Customize your portal display by entering the optional URL files. Make all URLs relative to `workingdir`:
   - *Header URL*—enter a header JSP for the default header page.
   - *Footer URL*—enter a footer JSP for the default footer page.
   - *Suspended*—select the check box to suspend the portal application and replace the portal home page with an 'under maintenance' screen until service resumes. To resume service, deselect the Suspended check box on the Edit Portal Definition tool.
   - *Suspended URL*—enter the default `suspended.jsp` to display the 'under maintenance' URL to end users while the application is in Suspended mode.
4. Click Create to create the portal definition.

   If the portal was successfully created, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.

5. Click Back to return to the home page. The new portal name displays under the Portals banner.

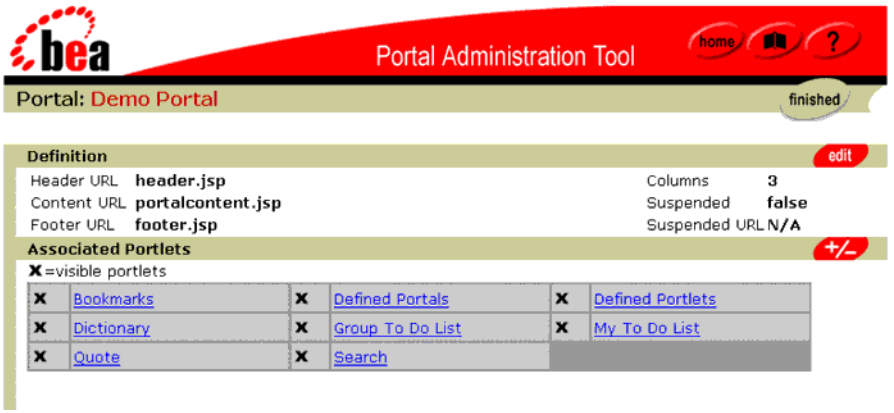
## Editing Portals

After creating a portal, you can edit it to associate portlets, groups, and users. You can also personalize the portal's layout and color scheme, and make changes to the definition.

To edit a portal:

1. On the Portal Administration Tool Home page, click a portal title link to see the portal-view page. The name of the portal you selected displays at the top of the screen. Colored banners separate each portal property and contain a command button for that property. The procedures in the next sections provide more information on editing group properties. Figure 3-4 shows the portal-view page.

**Figure 3-4 Portal-View Page**




2. When you are done viewing and editing the portal, click Finished at the top or bottom of the portal-view page to return to the home page.

## Editing Portal Definitions

You can edit the portal definition you created to reflect any changes to the associated URLs or number of portal columns.

To edit a portal definition:

1. On the portal-view page, click Edit in the Definition banner to display the Edit Portal Definition tool.
2. Enter the appropriate changes.
3. Click Save.

   If the changes were successfully made, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.
4. Click Back to return to the portal-view page.

## Adding and Removing Portlets

You can choose which portlets are available to a portal by adding and removing them from the system's list of all established portlets. From the narrowed list of portlets you associate with a portal, group, and end users further define which portlets they want available and visible on their personalized portal page.

To associate portlets with a portal:

1. On the portal-view page, click +/- in the Associated Portlets banner to display the Add or Remove Portlets tool.
2. To add a portlet to the portal, select Avail. The portlet is associated with the portal. It doesn't appear on the portal page until it is made visible by you, the Group Administrator or the end user.
3. To make a portlet visible, select Visible. The portlet is associated with the portal and now appears on the portal page.

4. To remove a portlet from the portal, select Unavail. The portlet becomes disassociated with the portal and unavailable to new groups and end users (including anonymous users). However, if the portlet has been personalized at a group or user level, it remains associated with those levels.
5. Click Save.

   If the changes were successfully made, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.
6. Click Back to return to the portal-view page. Available portlets appear in the Associated Portlets section of the screen with a gray background. Visible portlets are marked with an check mark.

## Editing Portlet Display Attributes

Portlet titles, associated with a portal, display as hot links in the Associated Portlets section of the portal-view page. These links open a tool that enables you to further specify how the portlet displays in the portal, overriding the display attributes established when the portlet was created. Group Administrators can further personalize these attributes.

To edit an associated portlet's display attributes:

1. Click the portlet title link in the Associated Portlets section of the portal-view page.
2. Enter the appropriate changes in the Edit Portlet Display Attributes tool.
3. Click Save.

   If the changes were successfully made, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.
4. Click Back to return to the portal-view page.

## Editing the Portal Layout

You can move a portal's associated portlets left and right between columns and up and down within columns depending on the column layout you selected when you created the portal. You can also change the percentage of the portal page that each column occupies. Group Administrators and end users can further personalize the portal layout.

To edit the layout of portlets in a portal:

1. On the portal-view page, click Edit in the Layout banner to display the Edit Portal Layout tool. This layout tool shows each portal column, its span percentage, and the portlets that display within those columns.
2. Select the portlet you want to move by clicking on it. The portlet name is highlighted.
3. Click an arrow to move the portlet up or down within a column, or right or left between columns.

To change the column spans of a portal layout:

1. Click in the percentage field associated with a column and enter a new percentage. The sum of all column spans should equal 100%. For single column portals, you may specify from 1% to 100%.
2. When you are done editing the portal layout, click Save.

   If the changes were successfully made, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.
3. Click Back to return to the portal-view page. A table in the Portal Layout section lists the portlets as you arranged them within each column.

## Editing the Portal Color Scheme

You can edit the overall appearance of a portal by changing its background color as well as the portlets' component colors, title colors, and border appearance.

To edit portal colors:

1. On the portal-view page, click Edit in the Colors banner to display the Edit Color Schemes tool. This tool provides five preset color schemes and a Custom Scheme tool.
2. In the Portlet Color Schemes section of the screen, select a preset color scheme or the custom color scheme. If you selected the custom color scheme, enter a hex color code in each text field, or click the color palette icon to select a color for each field from the Color Picker.
3. Select On to display portlet borders, or Off to omit portlet borders.
4. Select Black, White, or Other to choose the color of the text that displays in the portlet titlebar. If you selected Other, enter a hex color code in the text field, or click the color palette icon to select a color from the Color Picker.
5. In the Portal Background Color section of the screen, select Gray, White, or Other to choose a background color for the entire portal page. If you selected Other, enter a hex color code in the text field, or click the color palette icon to select a color from the Color Picker.
6. To preview your color selections, click the Click here to save changes and preview colors link. Your color changes are saved and the Edit Color Schemes tool redisplays the example portlet at the bottom of the screen to reflect your color preferences.
7. To save your color preferences without previewing them, click Save. The portal-view page displays the new colors associated with the portal in the Colors section of the screen.
8. To revert the portal appearance to its original color scheme, click Restore Defaults. The portal-view page displays the default colors associated with the portal in the Colors section of the screen.

## Associating Groups with a Portal

You can only associate portal groups from the portal-view page. For more information on associating users with a group and personalizing portal groups, see “Administering Portal Groups” on page 3-22.

To associate a group with a portal:

1. On the portal-view page, click +/- in the Associated Groups banner.
2. Expand the hierarchy (or search) to find the desired group.
3. Check the group.
4. Click Save to return to the portal-view page. The new group name displays in the Associated Groups section of the portal-view page.

To disassociated with a group:

1. On the portal-view page, click +/- in the Associated Groups banner.
2. Expand the hierarchy (or search) to find the desired group.
3. Uncheck the group.
4. Click Save to return to the portal-view page. The group name no longer displays in the Associated Groups section of the portal-view page.

## Deleting Portals

You can delete an existing portal from the Portal Administration Tool Home page. However, you must first disassociate all portal groups and users from that portal.

To delete a portal:

1. On the home page, click Delete in the Portals banner to display the Delete a Portal tool.
2. Select the portal from the Portal Name drop-down list.
3. Click Delete. A confirmation window displays.
4. Click OK to confirm the deletion.
5. Click Back to return to the home page. The portal name no longer displays in the Portals section of the Portal Administration Tool Home page.

# Administering Portal Groups

A portal group is a user group associated with a portal. The Portal Administration Tool enables you to personalize portal groups. You can personalize the layout, content, and color scheme.

Avoid creating portal groups for business-to-consumer portals with an unmanageable number of users.

You can edit portal groups from the portal-view page.

## Editing Portal Groups

Editing portal groups allows you to associate portlets with each group. You can also personalize the group's portal layout and color scheme.

To edit a portal group:

1. On the Portal Administration Tool Home page, click the portal title link the group is associated with. The portal-view page displays.
2. In the Associated Groups section of the screen, click the group title link you want to edit. The group-view page displays. The names of the portal and group you selected display at the top of the screen. Colored banners separate each group property and contain a command button for that property.The procedures in the next sections provide more information on editing group properties.

## Adding and Removing Portlets from a Portal Group

As the Group Administrator, you choose which portlets are available to a group by adding and removing them from the portal's list of all associated portlets. From the narrowed list of portlets you associate with a group, end users further define which portlets they want available and visible on their personalized portal page.

To associate portlets with a group:

1. On the group-view page, click +/- in the Associated Portlets banner to display the Add or Remove Portlets tool.

2. To add a portlet to the group, select Avail. The portlet is associated with the group. It does not display on the portal page until it is made visible by you or the end user.

3. To make a portlet visible, select Visible. The portlet is associated with the group and displays on the portal page.

4. To remove a portlet from the group, select Unavail. The portlet becomes disassociated from the group and unavailable to the group and end users. However, if the portlet has been personalized at the user level, it remains associated with those users.

5. Click Save.

   If the changes were successfully made, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.

6. Click Back to return to the group-view page. Available portlets appear in the Associated Portlets section of the screen with a gray background. Visible portlets are marked with an X.

## Editing the Portal Group Layout

You can move a group's associated portlets left and right between columns and up and down within columns. End users can further personalize the portal layout.

To edit the layout of portlets in a group:

1. On the group-view page, click Edit in the Layout banner to display the Edit Portal Layout tool. This layout tool shows each portal column and the portlets that display within those columns.

2. Select the portlet you want to move by clicking on it. The portlet name is highlighted.

3. Click an arrow to move the portlet up or down within a column, or right or left between columns.

4. When you are done editing the portal layout, click Save.

If the changes were successfully made, a confirmation message appears in red at the top of the screen. If not, an error message notifies you of the required changes.

5. Click Back to return to the portal-view page. A table in the Portal Layout section lists the portlets as you arranged them within each column.

## Editing the Portal Group Color Scheme

You can edit the overall appearance of a group by changing its portal background color as well as the portlets' component colors, title colors, and border appearance.

To edit group colors:

1. On the group-view page, click Edit in the Colors banner to display the Edit Color Schemes tool. This tool provides five preset color schemes and a Custom Scheme tool.
2. In the Portlet Color Schemes section of the screen, select a preset color scheme or the custom color scheme. If you selected the custom color scheme, enter a hex color code in each text field, or click the color palette icon to select a color for each field from the Color Picker.
3. Select On to display portlet borders, or Off to omit portlet borders.
4. Select Black, White, or Other to choose the color of the text that in the portlet titlebar. If you selected Other, enter a hex color code in the text field, or click the color palette icon to select a color from the Color Picker.
5. In the Portal Background Color section of the screen, select Gray, White, or Other to choose a background color for the entire portal page. If you selected Other, enter a hex color code in the text field, or click the color palette icon to select a color from the Color Picker.
6. To preview your color selections, click the Click here to save changes and preview colors link. Your color changes are saved and the Edit Color Schemes tool redisplays the example portlet at the bottom of the screen to reflect your color preferences.

7. To save your color preferences without previewing them, click Save. The group-view page displays the new colors associated with the portal in the Colors section of the screen.
8. To revert the portal appearance to its original color scheme, click Restore Defaults. The group-view page displays the default colors associated with the portal in the Colors section of the screen.

# Where to Get More Information

This chapter presents the basics of portal development using the WebLogic Commerce Server and WebLogic Personalization Server Administration Tools. For an in-depth look at portals and portlets, see the following chapters later in this document:

- Chapter 4, “Developing Portlets,” presents the application developer with essential information about creating custom portlets.
- Chapter 5, “Building a Custom Portal Step-by-Step,” presents a tutorial for customizing your own e-commerce portal.
- Chapter 6, “Advanced Portal Topics,” describes how to add some of the functionality of the WebLogic Commerce Server to your portal. This chapter also discusses deploying your application as a Web application.

You may need to consult the following documentation when using the WebLogic Commerce Server and WebLogic Personalization Server:

- Programming WebLogic JSP
- Programming WebLogic JDBC
- JSP documentation from JavaSoft at http://java.sun.com/products/jsp

CHAPTER

# 4 Developing Portlets

An integral part of any portal solution is the portlet application. This chapter explains what you need to know to create a portlet application.

This topic includes the following sections:

- Introduction
  - What Is a Portlet?
- Creating a Portlet Application
  - Defining the Portlet JSP
- Working Within the Portal Framework
  - Extending the PortalJspBase Class
  - Accessing Portal Session Information
  - Sending Requests Through the Flow Manager
  - Using URL Links in Your Portlet
  - HTML Form Processing
  - Retrieving the Home Page
  - Retrieving the Current Page
  - Setting the Request Destination
  - Tracking User Login Status
  - Loading Content from an External URL
  - Using Example Portlets
  - HTML Tables Versus HTML Frames

# Introduction

Generally, a main portal page is organized into smaller display areas. Using the WebLogic Personalization Server, the portal developer can create a main page layout, with flexible methods for determining custom headers, footers, look and feel elements, and the primary content areas.

The most information-rich part of the main page consists of a set of portlets, laid out in columns. Each portlet is a small content area, provided to display a particular type of information. These portlets are developed especially for each portal and are written in JSP, so there is great flexibility in what can be displayed. There is a standard set of development guidelines, coupled with portal services, to ensure portal and portlets are well-behaved.

The primary way dynamic functionality of the personalization components is made available to portlets is via custom JSP tags resident in tag libraries. These tags hide much of the internal run-time complexity of the Personalization Server, presenting a small, well-defined interface to its functions. Portlets may also access certain types of personalization EJBs directly, using embedded Java to access Personalization Server functionality.

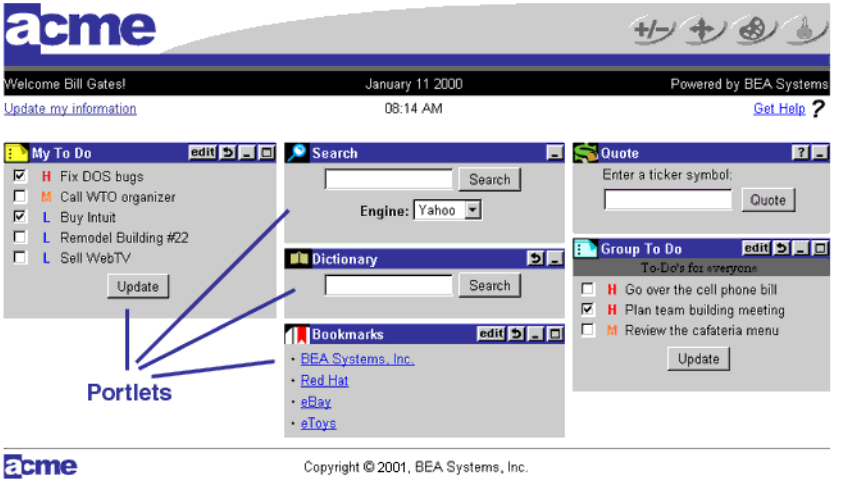
Each portlet may have a series of custom pages with specific functions associated with it, accessed via button clicks on the portlet. An edit page may make available to the user HTML input elements, in which the user can enter data on preferences specific to that portlet. A full page (or pages) version may be brought up to show an arbitrary amount of detail. A help page can be set up. The portlet may also be maximized, minimized, or floated in its own window.

## What Is a Portlet?

From the end user point-of-view, a portlet is a specialized content area that occupies a small “window” in the portal page. For example, a portlet can contain travel itineraries, business news, local weather, or sports scores. The user can personalize the content, appearance, and position of the portlet according to the profile preferences set by the administrator and group to which the user belongs. The user can also edit, maximize, minimize, or float the portlet window.

The following figure shows how portlets appear in a portal home page:

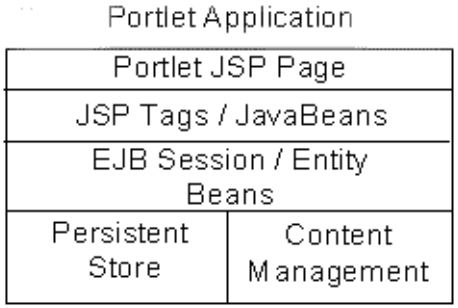
**Figure 4-1 Portlet Home Page View**




From a server application point-of-view, a portlet is a content component implemented as a JSP that defines the static and dynamic content for a specific content subject (weather, business news, etc.) in the portal page. The portlet JSP generates dynamic HTML content from the server by accessing data entities or content adapters implemented using the J2EE platform. The Portlet JSP then displays the content in the portal.

**Note:** All of the portlets in a portal are included in a *single* HTML page, through the use of the `<jsp:include>` action.

**Figure 4-2 Portal Application Programming Model**




The diagram shown above defines the portal application programming model. This programming model includes JSP, JSP tags, JavaBeans, EJBs, data stores, and content management stores. The portlet JSP contains static HTML and JSP code. This JSP code uses application or content specific JSP tags and/or JavaBeans to access dynamic application data through EJBs, content adapters, and legacy system interfaces. Once this data is retrieved, the portlet JSP applies HTML styling to it and the generated HTML is returned in the HTTP request to the client HTTP client.

# Creating a Portlet Application

To create a portlet application, you should be a J2EE developer with a background in JavaServer Pages (JSP), JavaScript and HTML, and have a knowledge of Enterprise Java Beans.

The portlet application is a JSP that contains code responsible for retrieving personalized content and rendering it as HTML.

Once you have created your portlets, you can associate them with one or more portals. Therefore, you must create your portlet applications before using the Portal Administration Tool to create and define your portal.

# Defining the Portlet JSP

The portal treats portlets as components or HTML fragments, not as entire HTML documents. The portal relies on the portlet application to create an HTML fragment for its portlet content. The portal renders the portlet's content in the portal page according to the personalization rules (the row and column position, colors, etc.) for the portal, group, and user levels.

When creating a portlet application, keep the following items in mind to ensure that your portlets run efficiently:

- Avoid using forms in a portlet that update the data within the portlet. This causes the entire portal to refresh its data which can be very time consuming. For more information on using an HTML form in a portlet, see"HTML Form Processing" on page 4-10.
- Place items that require heavy processing in an edit page or a maximized URL. Otherwise, the portal must wait for the portlet to process which considerably slows down the painting of the portal.

To define your portlet JSP:

1. Create a JSP for your portlet content.
2. Create JSPs for the portlet banner, header, footer, alternate header, alternate footer, help page, and edit URL as needed.

**Note:** You do not need to create a JSP for the portlet titlebar because it is included in the WebLogic Personalization Server (`public.html/portals/repository/titlebar.jsp`). The portlet titlebar displays the appropriate portlet titlebar icons and the name of the portlet you defined in the Portal Administration Tool.

**Note:** Avoid using the following HTML tags in your portlet content page. The HTML generated by the portlet content page is an HTML fragment contained in a larger portal HTML page, not a separate HTML document.

- `<html></html>`
- `<header></header>`
- `<body></body>`
- `<meta></meta>`

- `<title></title>`

3. Use the following portlet layout guidelines.

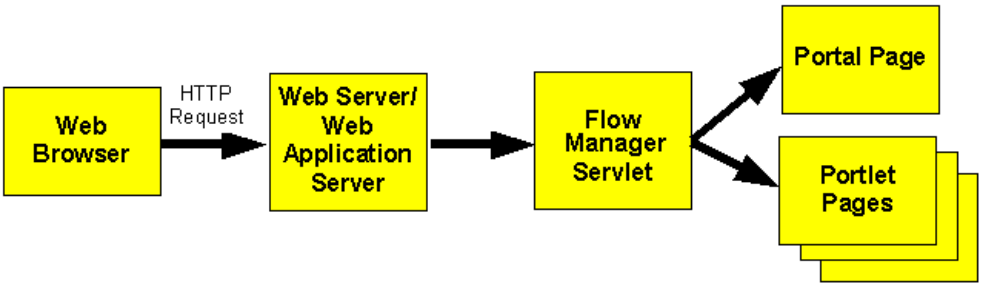
**Table 4-1 Portlet Layout Guidelines**

| Layout Attribute | Recommendation |
|---|---|
| Content Height | There are no restrictions on height as long as the content fits in your portal page. |
| Column Width | Take into account that the width of your portlet is controlled by the portal(s) it is associated with. A portal lays out your portlet content in a column based on portal, group, and user personalization rules. As a result, the width of your portlet should be well behaved. |
| Content Wrapping | Allow wrapping for all portlet content. Do **not** use the **NOWRAP** attribute in table cells. |
| Titlebar Icon Height | The image height attribute in titlebar.jsp is set to 20. |
| Titlebar Icon Width | The image width in titlebar.jsp is set to 27. |

# Working Within the Portal Framework

The portal framework consists of JavaServer Pages, JSP tag libraries, EJBs, Java servlets, and other supporting Java objects. The main Java servlet is the Flow Manager. The Flow Manager receives all incoming HTTP requests and dispatches each request to the appropriate destination URL. As a result, all access to your portal pages is controlled by the Flow Manager. The following diagram shows where the Flow Manager fits in the portal framework.

**Figure 4-3 Portal Framework**




# Extending the PortalJspBase Class

It is recommended that your portlet JSP extend the framework's `PortalJspBase` Java class. This class contains many convenience methods which perform general tasks for your portlet JSP page, such as accessing session information, the *traffic uri*, and user login information.

To extend the `PortalJspBase` class, include the following code at the top of your portlet JSP:

```
<%@ page
extends="com.beasys.commerce.portal.admin.PortalJspBase"%>
```

# Accessing Portal Session Information

The portal session information you can access from the `PortalJspBase` class are listed in the following table which lists the name, type, and description for each session value. For more information, see the *Portal Javadoc* API Documentation.

**Table 4-2 PortalJspBase Class Session Values**

| Session Value Name | Type | Description |
|---|---|---|
| `PortalAdminConstants.PORTAL_NAME` | `String` | The name of the portal associated with the current request. |

**Table 4-2 PortalJspBase Class Session Values (Continued)**

| Session Value Name | Type | Description |
|---|---|---|
| JspConstants.SERVICEMANAGER_SUCCESSOR | String | The name of the successor associated with the current session. The successor profile properties are used for those properties not specified by the user. |
| JspConstants.SERVICEMANAGER_USER | String | The name of the user associated with the current session. |
| UserManagmentConstants.PROFILE_USER | Configurable Entity | The user profile associated with the current request or the session. |
| UserManagmentConstants.PROFILE_SUCCESSOR | Configurable Entity | The group profile associated with the current request or the session. |
| UserManagmentConstants.PROFILE_SUCCESSOR_UID | Long | Unique IDs for the configurable entities. |
| UserManagmentConstants.PROFILE_USER_UID | Long | Unique IDs for the configurable entities. |

You can retrieve the portal session information described above through the following `PortalJspBase` methods:

- `public Object getSessionAttribute(String aName, HttpServletRequest aRequest)`
- `public void setSessionAttribute(String aName, Object aValue, HttpServletRequest aRequest)`
- `public void removeSession(String aName, HttpServletRequest aRequest)`

You can set the portal session's `SERVICEMANAGER_USER` and `SERVICEMANAGER_SUCCESSOR` through the following `JspBase` methods:

- `public static void setUser(String aUser, HttpServletRequest aRequest)`
- `public static void setSuccessor(String aSuccessor, HttpServletRequest aRequest)`
- `public static void setUserAndSuccessor(String aUser, String aSuccessor, HttpServletRequest aRequest)`

# Sending Requests Through the Flow Manager

Remember that all HTTP requests and responses are sent to the Flow Manager servlet. Therefore, your portlet HTML must refer to the Flow Manager's URL for URL links and HTML form processing.

# Using URL Links in Your Portlet

If your portlet contains links to a JSP page that is not a portlet, use the following `PortalJspBase` method to create your URL and to guarantee that the HTTP request is sent to the service manager URL:

```
public String createURL(HttpServletRequest aRequest, String
destination, String parameters)
```

The destination should be a relative or qualified file location in the form such as `example/mytodo.jsp`, or `/yourportal/example/mytodo.jsp`. The path is relative to the `documentRoot`, as specified in the WebLogic console. Parameters should be a string such as `column=4&row=5`.

**Note:** Parameter values should already be encoded as you would for any HTTP request. Example: `String parms = "column=" + java.net.URLEncoder.encode("4");`

Because of the way the JSP engine handles `jsp:forward` and `jsp:include`, you must fix up the relative URLs in your portlet, especially relative links to images. The Web browser thinks the root for relative links is the directory in which the Flow Manager resides and not your portlet's directory.

To fix up relative URLs, use the following `ToolsJspBase` method:

```
public static String ToolsJspBase fixupRelativeURL(String aURL,
HttpServletRequest aRequest)
```

where *`aURL`* is the destination URL to fix up and *`aRequest`* is the current HTTP request. In your JSP page, use the following method to code a `fixup`:

```
<img src="<%=fixupRelativeURL("images/quote.gif",
request)%>"width="50" height="35" border="0">
```

**Note:** For the repository feature to work with `jsp:include` and `jsp:forward`, use *reconcile file* to determine the correct location of the file that is included or forwarded. For example:

```
(jsp:forward page="<%=reconcileFile(request, "login.jsp")%>" />
```

# HTML Form Processing

If your portlet contains an HTML form, send all requests to the Flow Manager and set the destination request parameter.

To process HTML forms:

1. Set the form action to `action=getTrafficURI(request)`. This sends the form action request to the Flow Manager. This calls the `PortalJspBase` method:

   ```
   public String getTrafficURI(HttpServletRequest aRequest)
   ```

   The following example shows the use of the HTML form action to send a form request to the Flow Manager:

   ```
   <form method="post" action="<%=getTrafficURI(request)%>">
   ```

2. Set the destination request parameter in the HTTP post request. This tells the Flow Manager where to dispatch the request.

   To set the request destination for HTML forms, enter the following code within your form in your JSP page:

   ```
   <input type="hidden" name="<%=DESTINATION_TAG%>"
   value="example/mytodo.jsp">
   ```

   **Note:** Do not go through the Flow Manager for HTTP requests to other servers.

# Retrieving the Home Page

The Flow Manager sets the Home page for each portal in the Portal Framework session information. The Home page is registered as an initial argument for Flow Manager servlet in the portal's `APPLICATION_INIT` property set. Use the following `PortalJspBase` method call to retrieve the Home page:

```
public String getHomePage(HttpServletRequest aRequest)
```

# Retrieving the Current Page

You can also retrieve the current page from the Portal Framework session information by using the following `PortalJspBase` method:

```
public String getCurrentPage(HttpServletRequest aRequest)
```

**Note:** When you maximize a portlet, the current page changes to `fullscreenportlet.jsp`.

# Setting the Request Destination

When routing a request through the Flow Manager, you must specify the destination that should receive the request. The destination can be relative to the current page (portal.jsp, full-screen portlet.jsp, etc.) or a fully qualified path from the document root.

**Note:** The `DESTINATION_TAG` constant is available in `PortalJspBase`.

If your portlet contains links to other portal pages, use the following `PortalJspBase` method to create your URL and to guarantee that the HTTP request is sent to the service manager URL:

```
public String createURL(HttpServletRequest aRequest, String
destination, String parameters)
```

The destination should be a relative or qualified file location in the form such as `example/mytodo.jsp`, or `/yourportal/example/mytodo.jsp`.

In some cases, you may need to override the request parameter used by the Flow Manager. For example, use an override destination if your page contains a form that needs to be validated and forwarded elsewhere after validation. Use the following `PortalJspBase` method in your JSP page:

```
public void setOverrideDestination(HttpServletRequest req, String
dest)
```

To set the request destination for HTML forms, enter the following code within your form in your JSP page:

```
<input type="hidden" name="<%=DESTINATION_TAG%>"
value="example/mytodo.jsp">
```

# Tracking User Login Status

You can log the user in or out and track whether a user is currently logged in.

Use the following `PortalJspBase` method to track the user login status of a portal session:

```
public void setLoggedIn(HttpServletRequest aRequest,
HttpServletResponse aResponse, boolean aBool)

public Boolean getLoggedIn(HttpServletRequest aRequest)
```

# Loading Content from an External URL

According to the JSP specification, a JSP processed by a JSP engine must be relative to the server in which the JSP engine is running, requiring that all of your portlets reside in your portal server and not on an external Web site. However, you can use the `uricontent` tag to download the contents of an external URL into your portlet. If you download the contents of a URL into your portlet, you need to fully qualify the images located on the remote server because the relative links contained within the remote URL will not be found unless fully qualified.

Use the following method to load content from an external URL:

```
<es:uricontent id="uriContent"

   uri="http://www.beasys.com/index.html">
<%
out.print(uriContent);
%>

</es:uricontent>

The sample <es:uricontent> tag is available in
application/portals/repository/portlets/_uri_example.jsp
```

# Using Example Portlets

The /server/application/portals/repository/portlets directory of the WebLogic Personalization Server contains example portlets. The following table lists the name of each example portlet, its description, and its associated files.

**Caution:** The example portlets are intended for illustration purposes only and should not be used for production code.

**Table 4-3 Example Portlet Descriptions and Associated Files**

| Example Portlet | Description |
|---|---|
| _uri_example.jsp | Demonstrates how to implement the uricontent tag to import contents from another URL on the Internet. |
| bookmarks.jsp | Displays the bookmarks associated to the current user.<br>■ bookmarks_edit.jsp–edit screen for the bookmarks.<br>■ images/pt_bookmark.gif–bookmark icon for the portlet titlebar. |
| definedportals.jsp | Displays the portals defined in the system. Uses the <es:foreachinarray>, <es:simplereport>, and <wl:sqlquery tags>. |
| definedportlets.jsp | Displays the portlets defined in the system. Uses the <es:foreachinarray>, <es:simplereport>, and <wl:sqlquery tags>. |
| dictionary.jsp | Demonstrates how to redirect a portlet to an external site.<br>■ images/pt_dictionary.gif –dictionary icon for the portlet titlebar. |
| generic_todo.jsp | For a complete generic_todo.jsp example, see Using the Default Implementation. |
| news_index.jsp | Demonstrate use of <cm:> tags. |
| news_viewer.jsp | Display content driven from content_index.jsp. (Use in conjunction with content_index.jsp.) |

**Table 4-3 Example Portlet Descriptions and Associated Files (Continued)**

| Example Portlet | Description |
|---|---|
| `grouptodo.jsp` | Displays a `Group To Do List`.<br>■ `todo.js` –statically included file that does not run by itself. It requires user information from `grouptodo.jsp`.<br>■ `grouptodo_edit.jsp`–edit URL for `grouptodo.jsp`.<br>● `todo_edit.jsp`–statically included file that does not run by itself. It requires user information from `grouptodo_edit.jsp`.<br>■ `grouptodobanner.jsp`–banner for the `grouptodo.jsp`.<br>■ `images/pt_group_list.gif`–Group To Do List icon for the portlet titlebar. |
| `mytodo.jsp` | Displays a `My To Do List`.<br>■ `todo.jsp`–statically included file that does not run by itself. It requires user information from `mytodo.jsp`.<br>■ `mytodo_edit.jsp`–edit URL for `mytodo.jsp`.<br>● `todo_edit.jsp`–statically included file that does not run by itself. It requires user information from `mytodo_edit.jsp`.<br>■ `images/pt_my_list.gif`–My To Do List icon for the portlet titlebar. |
| `quote.jsp` | Demonstrates how to redirect a portlet to an external site.<br>■ `images/pt_quote.gif`–Quote icon for the portlet titlebar. |
| `search.jsp` | Demonstrates how to redirect a portlet to an external site.<br>■ `images/pt_search.gif`–Search icon for the portlet titlebar. |

# HTML Tables Versus HTML Frames

BEA WebLogic Commerce Server does not prevent the use of any kind of HTML, including HTML frames. You will see that the demos and examples that ship with the product are all reference implementations which use HTML tables.This tabular style is not a requirement of the product—you can write HTML however you like. If you choose to use HTML frames, keep the following considerations in mind:

- When using frames, performance may be an issue, as each HTML frame is a separate page request.
- Since HTTP is stateless, managing state between HTML frames is difficult. In a single application, using a table allows a single HTTP request to be made with all the portlets gathered in one request.

If you choose to use frames, you will need to write HTML code to layout the portlets.

CHAPTER

# 5 Building a Custom Portal Step-by-Step

This chapter is a tutorial for building your own custom e-commerce portal. It assumes minimal knowledge of BEA products, and some knowledge of HTML and JSP. If you are new to WebLogic Server, WebLogic Commerce Server and WebLogic Personalization Server, and want to get up to speed quickly, this chapter is for you.

It is recommended, but not required, that you review the Personalization Tour before proceeding with this chapter.

This topic has the following sections:

- Introduction
  - Terminology
  - How to Use This Chapter
- Creating the Framework for Your Custom Portal
  - Installing WebLogic Personalization Server
  - Setting Up the Portal Framework
  - Repository Directory
- Simple Customizations
  - Project 1: Customizing the Acme Logos
  - Project 2: Customizing the Choice of Portlets
  - Project 3: Customizing the Layout of Portlets
  - Project 4: Describing Your Users

- Writing Your Own Portlets
  - Project 5: Building a Static Portlet
  - Project 6: Building a Simple Dynamic Portlet
  - Project 7: Building a Dynamic Portlet Using JSP Tags
- Advanced Portlet Functionality
  - Project 8: Adding a Maximized URL
  - Project 9: Changing the Look of a Maximized Portlet
  - Project 10: Inter-portlet Communication
  - Using the HTTP Request Method to Communicate Between Portlets
- Other Customization Techniques
  - More Portlet Customization
  - Database Interaction
  - Java Beans Interaction
  - Personalization Advisor Functionality
  - Internationalization
  - Using Webflow
  - Commerce Functionality
  - Modifying the Portal Framework
  - Building Your Site Without the Portal Framework
- Framework Files

**Note:** Throughout this chapter, the environment variable `WL_COMMERCE_HOME` is used to indicate the directory in which you installed the WebLogic Commerce Server and WebLogic Personalization Server software.

# Introduction

Internet portals are a key part of many e-commerce applications. Portals provide an entry point to the Internet as well as value-added services such as searching and application integration. The WebLogic Personalization Server allows you to quickly assemble both Business-to-Consumer and Business-to-Business portals that require personalized application content on the Internet.

The WebLogic Personalization Server enables Web developers to create portal Web pages and personalized application content for each portal user. The WebLogic Personalization Server uses JSPs, a part of the J2EE specification, in conjunction with a special library of JSP tags, standard HTML, Enterprise Java Beans (EJB), portal end user and the Portal Administration Tools, and a preconfigured database to store portal component entities.

## Terminology

Before you can begin building your portal, familiarize yourself with the following terminology.

**%WL_COMMERCE_HOME%**
: The folder in which you installed WebLogic Personalization Server.

**portal**
: This word has a specific meaning when working with the WebLogic Personalization Server product. A portal is a page that is intended to be the starting point for a user on a site. Furthermore, this chapter assumes that you will be using the Portal Framework included with WebLogic Personalization Server to build your portal.

**Portal Framework**
: A collection of prebuilt JSP pages included with the WebLogic Personalization Server distribution that provide the core functionality for portals. They are located in `%WL_COMMERCE_HOME%/config/wlcsDomain/applications/wlcsApp/exampleportal/portals/repository`.

When using the framework, your pages will have a common layout. In this layout, a page's real estate will be divided into three main areas: a header, a content area, and a footer. The header resides at the top of the page and typically contains a full-sized logo for the site plus some navigation features. The footer resides at the bottom of the page and typically contains legal notices, copyright information, and a small logo. The middle section, the content area, contains any number of small independent components called portlets. The JSPs included in the Portal Framework manage the layout of these portlets on the page.

**Note:** You are not required to use the Portal Framework. You may build your site from scratch, although, it is not recommended for new users of the system.

**portlet**

A JSP page that is displayed within a portal page. There is a one-to-many relationship between a portal and its portlets. Each portlet should provide a limited piece of functionality. For example, imagine an information portal where one portlet gives the weather report, another provides a stock ticker, another the top news stories, and another that shows yesterday's sports scores.

**administration tool**

WebLogic Personalization Server ships with the WebLogic Commerce Server Administration Tool. The focus of this chapter is on development; it will not provide detailed instructions on how to use the tool. If you have questions on how to use the Administration Tool, refer to Chapter 3, "Using the Portal Management Administration Tool."

**example portal**

The name given to a sample portal implementation included with the WebLogic Personalization Server distribution. This example is built on top of the Portal Framework. If you took the Personalization Tour, you worked with the example portal. It is branded with an "Acme" logo. The files for this example portal coexist in the same folder with the files used for the Portal Framework. The difference is the Portal Framework files are generic, while the example portal files are specific to the example.

**property set**

WebLogic Personalization Server supports the storage of collections of data called property sets. These sets may be associated with users, groups, or sites. In this chapter, you will need to create and edit a property set that describes your portal. This kind of property set is called an `APPLICATION_INIT`

property set and describes properties such as your portal's name, its working directory, and the home page.

# How to Use This Chapter

Within the WebLogic Commerce Server Administration Tool is a Portal Management Administration Tool (the "Portal Manager") that allows you to quickly build a basic portal using the Portal Framework. Chapter 2, "Creating a Portal Using the Example Portal," includes step-by-step instructions for building the Acme Demo Portal. Techniques for using the Portal Manager are documented in Chapter 3, "Using the Portal Management Administration Tool." This tutorial will not repeat the information presented in earlier chapters.

The goal of this chapter is to get you started building your own custom portal. It will cover many techniques for customizing the Portal Framework. This chapter also provides many small projects which will demonstrate how to use these techniques. The code fragments used to build these projects are included.

However, this chapter does not explain what every line of code does in these samples. It provides general guidance in understanding how an example works, but the details are left as an exercise to the user. The reason for this is that the best way to learn how to develop with WebLogic Personalization Server is to reverse engineer code written by others. Once you get each example working, spend some time experimenting with the code. A good rule of thumb is to not proceed to the next example until you know what each line of code does in the previous example.

# Creating the Framework for Your Custom Portal

This section describes how to build a custom portal. At the end of this section, you will have created a copy of the example portal (which uses the Portal Framework) which you will alter as you build your custom site. It is important that you use the example portal as a base since it does provide extensive functionality. Later, when you gain familiarity with the product, you can re-engineer your custom site one piece at a time.

**Note:** It is not recommended for new WebLogic Personalization Server developers to attempt to build a portal from scratch.

This section will walk you through the process one step at a time. It is primarily intended to help you get the framework of your custom portal up and running and does not attempt to explain the details of this process. In later sections, you will be introduced to the details in a more rigorous way.

## Installing WebLogic Personalization Server

If you have already installed WebLogic Personalization Server, begin this procedure with "Setting Up the Portal Framework" below.

To install WebLogic Personalization Server, refer to the *Installation Guide*.

Test WebLogic Personalization Server by clicking on `StartCommerce.bat` (`StartCommerce.sh` for UNIX users). It should start up and print "WebLogic started." **Do not** shut down the server.

**Note:** It is very important to look at the console window and inspect the output for Java exceptions. If any exceptions occurred during startup, you will need to resolve the problem before WebLogic Personalization Server will work properly.

When you have completed the installation of WebLogic Personalization Server, you can proceed with setting up the framework for your custom portal.

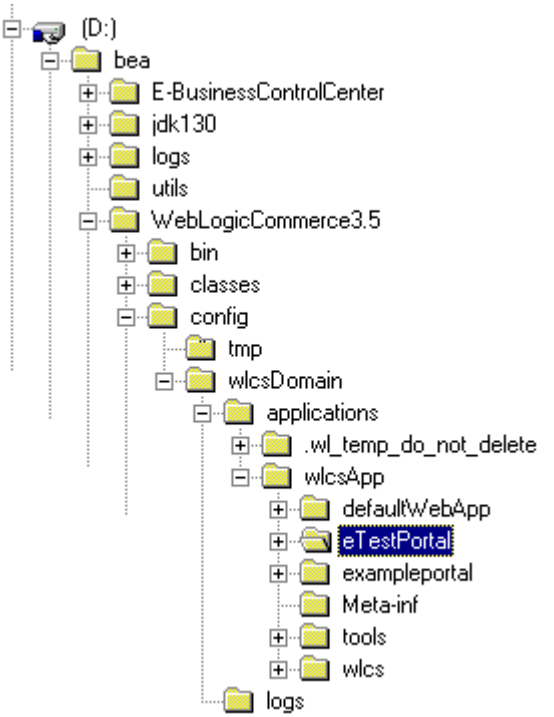
# Setting Up the Portal Framework

To set up the framework for your custom portal, follow the steps listed below.

## Create a New Web Application Directory

1. Create a new Web application directory named `eTestPortal` by copying and pasting the `exampleportal` directory and renaming it `eTestPortal` in the `wlcsApp` directory, as shown in Figure 5-1.

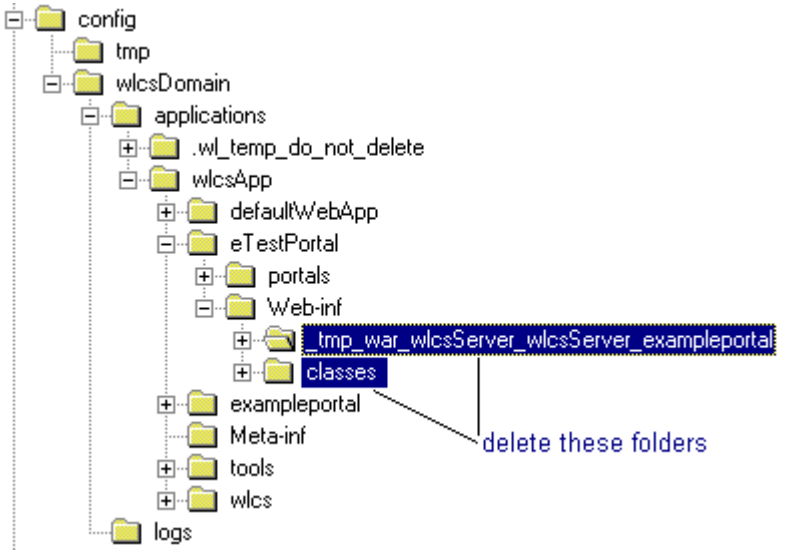
**Figure 5-1 Create a New Web Application Directory**




**Note:** Do not use spaces in this folder name or you will have complications in step 21.

2. Delete any `_tmp*` directories and the `classes` directory in the `eTestPortal/Web-inf` directory, as shown in Figure 5-2. (This step is optional.)

**Figure 5-2 Optionally Delete the _tmp* and Classes Directories.**




3. In the WebLogic Server console, edit the `WL_COMMERCE_HOME/config/wlcsDomain/applications/wlcsApp/META-INF/application.xml` file, and place a reference to the new module there. For information about creating references to modules, refer to "Module Elements" in the "Reference Domain chapter of the *Deployment Guide*.

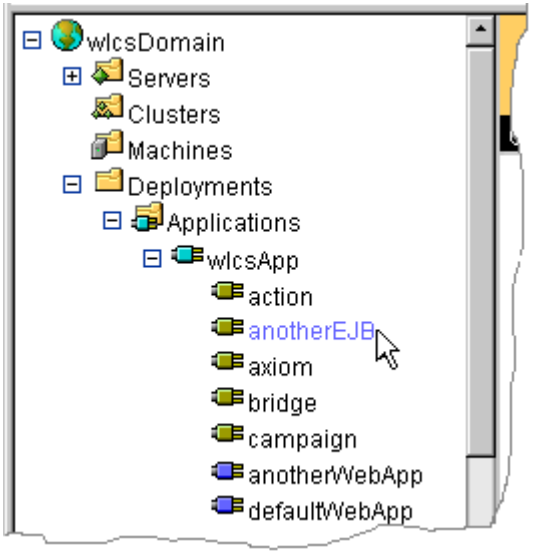
4. If the server is running, restart it.

5. Start the WebLogic Administration Console for your domain. For more information about starting the console, refer to the section "Starting the Administration Console for the Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server Domain" in the "Server Configuration chapter of the *Deployment Guide*.

6. In the Administration Console, in the left pane, click Deployments → Applications → wlcsApp.

7. Under wlcsApp, click the EJB or web application that you added to `application.xml`. (See Figure 5-3.)

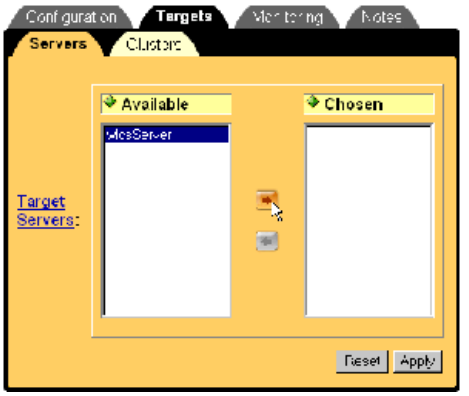
**Figure 5-3 Click the EJB or Web Application That You Added**




The right pane of the Administration Console displays the module deployment properties.

8. In the right pane, click the Targets tab. Then click the Servers subtab.
9. On the Servers tab, move wlcsServer from the Available list to the Chosen list.
10. Click Apply.

**Figure 5-4 Deploy the Module on wlcsServer**



## Use the Portal Management Tool to Create a New Portal "eTestPortal"

11. Start the WebLogic Commerce Server.
12. Log into the WebLogic Commerce Server Administration Tool. If you installed WebLogic Personalization Server with the default settings, you can use this URL in a browser that is invoked on the same machine as the server: `http://localhost:7501/tools`. The default username is `administrator` and the default password is `password`.
13. Click the blue and red monitor icon on the Portal Management banner. This will take you into the Portal Manager.
14. Click the Create button on the Portals banner. This will allow you to register your new portal with the server.
15. A form will appear.
    a. In the Portal Name form control, enter `eTestPortal` exactly as you did while editing the property set. Leave the rest of the controls as they are by default.
    b. Click Create. This should succeed. You have now successfully registered your new portal.
    c. Click Back to return to the Portal Manager page.

## Make Desired Portlets Visible

16. Click your portal's name underneath the Portal banner. This will take you to the portal editor page.
17. By default, no portlets will be included in your portal. You should add a few portlets. Click the +/- icon on the Associated Portlets banner. You will now be directed to the Portlet Association page.
18. Make a few portlets available, and at least a couple of portlets visible. Click Save and then Back when finished.

    **Note:** Not all of the displayed portlets may be valid for your new portal. If any portlets were added for another portal on the server, these portlets may have a different relative path from `%WL_COMMERCE_HOME%/config/wlcsDomain/applications/wlcsApp/exampleportal/portals`. Adding a portlet located in another portal will give your users a run-time error. To avoid this problem, use only the

portlets located in the `%WL_COMMERCE_HOME%/server/application/portals/repository/portlets` subfolder (consult the list below).

The following list describes the available portlets. For more information about these portlets, see "Creating Portlets for Your Demo Portal" on page 2-4 in this guide.

- Defined Portals—displays the portals defined in the system. Uses the `<es:forEachInArray>` and `<es:simpleReport>` tags.
- Defined Portlets—displays the portlets defined in the system. Uses the `<es:forEachInArray>` and `<es:simpleReport>` tags.
- News Index—demonstrates the use of Content Management tags.
- News Viewer—displays content driven from `content_index.jsp`. Use in conjunction with News Index.
- Quote—displays stock quotes. Demonstrates how to redirect a portlet to an external site.
- Dictionary—demonstrates how to redirect a portal to an external site.
- Search—demonstrates how to redirect a portal to an extend site.
- Group To Do List*—displays a Group To Do list. Requires user to be logged in.
- My To Do List*—displays a My To Do list. Requires user to be logged in.
- Bookmarks*—displays the bookmarks associated with the current user. Requires user to be logged in.

**Note:** Portlets marked with an asterisk (*) require a user to be logged on to be visible.

## Use the Property Set Management Tool to Create a New Application Init Property Set "eTestPortal"

19. Click the triple-can icon on the Property Set Management banner. This will take you into the Property Set Manager.
20. To register your eTestPortal, you need to create a new property set. Click the Create button on the banner.

21. You are presented with a form.

    a. In the Name field, enter `eTestPortal`.

    b. In the Description field, enter something like `My test portal`.

    c. In the Copy Properties From drop-down list, select `APPLICATION_INIT._DEFAULT_PORTAL_INIT`.

    d. Finally, in the Property Set Type drop-down list, enter `APPLICATION_INIT`.

    e. Once you have completed the form, click the Create button.

    **Note:** Make sure you type this exactly as you see it. It is case sensitive and spaces should not be used.

22. You have just created a property set that will be used to register your application. Click the Back button.

    **Note:** Clicking Back will fail with an "Authorization Failed" message if your browser does not allow cookies. In this case, you must change your browser settings to allow cookies for the Administration Tool to function properly.

23. You now need to edit your new property set. Click the eTestPortal name in the Application Initialization property set list. This will invoke the property set editor for the eTestPortal property set.

24. Edit the portalName property. This must exactly match the name of the portal you are creating; in this example it should be eTestPortal.

25. Return to the home page. Click Finished, then click Home.

## Restart WebLogic Server

26. It is not possible to hot deploy a Web application when it is being added to an existing application, as we are doing here in `wlcsApp`. You will need to restart the server in order for the changes you made to `config.xml` to take effect.

## View Your Portal Site in a Browser

27. Test your portal site as follows:

    Open a browser window and type in `http://localhost:7501/eTestPortal`.

a. Replace `localhost` in this URL with the name of the machine the server is running on if it is different from the browser's machine.

b. Replace `7501` if you changed the listening port of WebLogic Server during the WebLogic Personalization Server installation.

c. If you created a portal and a property set with a different name, replace `eTestPortal` with the name of the property set.

In your browser, you should see a Web page with an Acme logo. You should see all the portlets which you defined as being visible in the Portlet Administration page. Portlets that require a user login (marked with an ***** above) will not display until you are logged in.

**Note:** It is important to look at the console window to make sure exceptions are not being thrown.

## Refreshing the Browser

Your browser may cache pages. If you make modifications to your site and they do not appear in your browser, click Refresh. This will circumvent the cache and get an updated page from the WebLogic Personalization Server. If this fails to work, flush your browser's disk cache, as follows:

- **Internet Explorer users:** from the Tools menu, choose Internet Options. From the General tab of the dialog box, click the Delete Files button.
- **Netscape users:** from the Edit menu, choose Preferences. From the browse tree, click Advanced, then Cache. Click both the Clear Memory Cache button and the Clear Disk Cache button.

If you are still not able to see your latest changes, try switching to a different browser (Internet Explorer or Netscape) to view the page.

## Troubleshooting

If you do not see the portal page or have exceptions, make sure you have not made the following common mistakes:

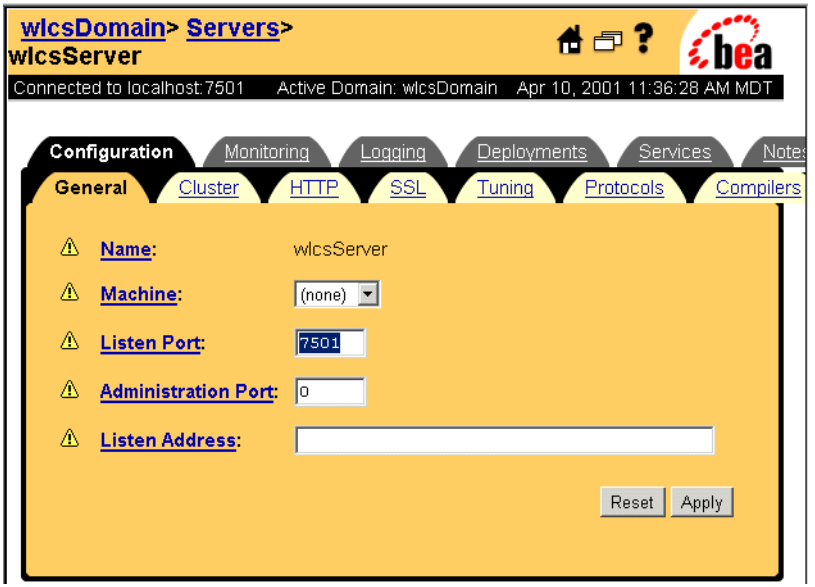
**Problem: Server not responding**

a. Server is not running—make sure you have a console window with the Commerce Server running. It must output "WebLogic Server started." before it will accept connections.

b. Server is running on different machine—this chapter assumes that your server is running on the same machine as your browser. If not, replace the name `localhost` in your browser URLs with the name of the machine on which the server is running.

c. Server is running on non-default port—this chapter assumes that your server is running on port 7501. Check the Listening Port field in the WebLogic Administration Console, as illustrated in Figure 5-5. The number assigned to this property is the server's port number.

**Figure 5-5 Check Your Server's Listening Port in the WLS Console.**




**Problem: Server returns error**

a. `PORTAL_NAME` not defined—when creating your property set, you must replace the default value of the `PORTAL_NAME` property.

b. PORTAL_NAME does not match portal name—the PORTAL_NAME property in your property set (step 21.) must match exactly the name you give your portal when creating the portal in the Portal Manager (step 15.)

c. repositorydir incorrect—you should have modified several paths in your property set (step 24.) but not repositorydir. The repositorydir property should be /portals/repository.

28. Explore your new site. Create a new user account by clicking on the key icon in the top right-hand side of the page. Build a personalized Web page for your new user.

You have completed the first step in building your own custom portal.

# Repository Directory

An important concept to understand is the repository directory. The repository directory (specified by your repositorydir property in your property set) is the location where the server looks to find a resource if it cannot find it in your working directory (specified by your workingdir property in your property set). If you followed the previous instructions, you did not populate your working directory with any files. Therefore, when you navigated to your site in "Setting up the Portal Framework," (step 12.) the server failed to find the files (specifically, your home page portal.jsp) in your working directory and so it found them in your repository directory instead. It is important that you understand how this works.

Before you proceed, take a few moments now to read back through this section and review the steps you followed. Although you may not understand why each step was necessary, it is helpful to have a clear understanding of what the steps were. You can look in the repository directory to see what resources are provided by default:
%WL_COMMERCE_HOME%/config/wlcsDomain/applications/wlcsApps/exampleportal/portals/repository
The repository directory contains the Portal Framework and resources specific to the Acme portal and its portlets.

The rest of this chapter is devoted to making incremental changes to your copy of the example portal so that it is transformed into your own custom portal. In the process, you will replace many pieces of the example portal, though your site will still be based on the Portal Framework. You will *not* make changes to repositorydir.

# Simple Customizations

The previous section described how to establish a platform for your custom portal. This section assumes that you have successfully completed that process. Whereas the last section had a strictly defined process, the following sections are project based. It is recommended that you do these projects in order, although it is not required. Each project will list any prerequisite projects.

## Project 1: Customizing the Acme Logos

Start by removing the Acme logo and replacing it with your own brand image, as follows:

1. Open the `%WL_COMMERCE_HOME%/config/wlcsDomain/applications/wlcsApp/examp leportal/portals/repository/images` folder using your preferred filesystem navigator. The file listing will show all of the graphics used in your custom portal. Spend some time opening these graphics so you become familiar with the graphical components of your site.

2. Copy the files `logo.gif` and `logo_small.gif` to a new `images` subfolder in your working directory. For example: `../portals/example/images`

   By doing this, you are creating your own copy of these images outside of the repository. Therefore, when these images are used in your portal, they come from your working directory and not the repository.

3. Launch your preferred image editing application. This application must support reading and writing the GIF file format.

4. Using your image editor application, open the file called `logo.gif` located in your `../example/images` subfolder.

5. When the image opens, you will notice that it is the Acme logo image that appears at the top of your custom portal page.

6. Replace the Acme logo with your brand image. You may do this either by authoring a new image in your image editing application or using an existing GIF file with the same approximate dimensions as the original Acme logo.

7. Rename the Acme logo (`logo.gif`) to `acmelogo.gif`. Now save your own GIF format image as `logo.gif` in the `../example/images` subfolder.

8. There is another image called `logo_small.gif` located in your `images` subfolder that also needs to be updated. This is simply a smaller version of the main logo and is used on the bottom of your portal pages. Update this image as you did the first image.

9. Be sure you have updated the files in the `../example/images` folder and that the new GIFs have exactly the same name as the original Acme GIFs.

10. Use your browser to display the first page of your custom portal, following the procedure described in "View Your Portal Site in a Browser" on page 5-12.

    **Note:** WebLogic Personalization Server will automatically detect the new images and load them in, so you do not need to restart the server. Remember to click Refresh in your browser. If after clicking Refresh, you do still do not see your new image, you may have to flush your browser's disk cache. For more information, see the section "Refreshing the Browser" on page 5-13.

# Project 2: Customizing the Choice of Portlets

In the previous section, you were asked to randomly choose a set of portlets to assign to your portal. Now that you are up and running, it is time to revisit the choice of portlets you made.

## Use the Portal Management tool to select portlets

1. Log on to the WebLogic Personalization Server Administration Tool. (For more information on completing this step, see "Logging On to the Administration Tool" on page 3-2.)

2. Go to the Portal Manager and double-click the name of your portal under the Portals banner. You will see the portal editor page. Click the +/- icon on the Portlets banner. The portlet selection editor will appear.

3. Experiment with the controls and choose the portlets that you want to include. At this time, you can choose from only the prebuilt portlets. In later projects you will be building your own portlets.

When choosing a portlet for your site, you may also specify whether it is visible by default, not visible but available, or not available at all.

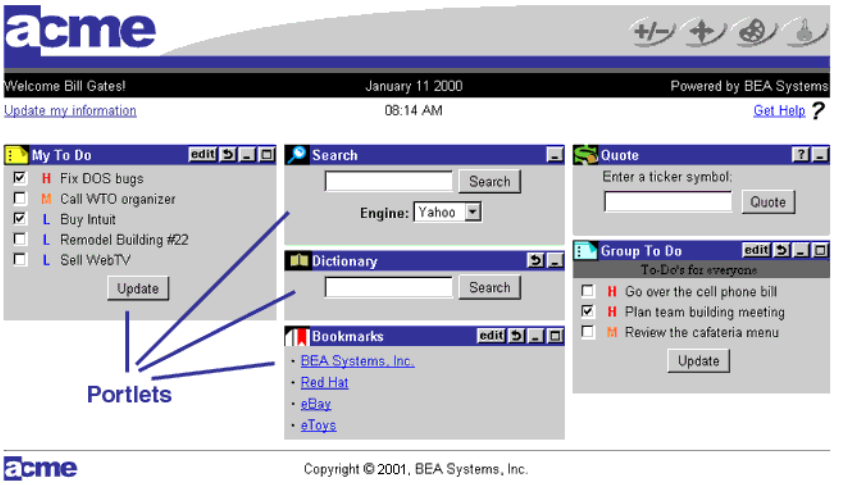
**Note:** Remember that some of the pre-built portlets will not be visible in the portal until the user is logged in. (See the section "Make Desired Portlets Visible" on page 5-10.)

For more information about using the Administration Tool, see the section Administering Portlets in Chapter 3, "Using the Portal Management Administration Tool," in this guide.

# Project 3: Customizing the Layout of Portlets

If you look carefully at your portal, you will notice that your portal has three main sections: a titlebar, a container in the middle for a number of smaller components, and a footer. The smaller components in the middle space are called "portlets," as shown in Figure 5-6.

**Figure 5-6 Portlets Laid Out in Columns in a Portal Page**




Each portlet is written as an independent JSP or HTML file. Each portlet is responsible for its own contents, while the portal page is responsible for laying the portlets out in columns.

You can customize this layout in two ways:

1. You can specify how many columns the portal uses to arrange portlets (1, 2 or 3 columns are valid).
2. You can choose which portlets appear in each of the columns.

**Note:** As an administrator you may define how the layout appears by default, but the user may override your choices.

### Use the Portal Management tool to rearrange portlets

1. Log on to the WebLogic Personalization Server Administration Tool. (For more information on completing this step, see "Logging On to the Administration Tool" on page 3-2.)
2. Go to the Portal Manager and then click the name of your portal under the Portals banner. You will see the portal edit page. By editing the portal definition, you may change the number of columns used to display your portal.
3. To change which portlets are in which column, use the Layout editor. (For more information on completing this step, see "Editing the Portal Layout" on page 3-19.)

For more information about using the Administration Tool, see Chapter 3, "Using the Portal Management Administration Tool," in this guide.

## Project 4: Describing Your Users

Your portal site will most likely have a number of categories of users. WebLogic Personalization Server recognizes this by supporting a feature called User Groups. With this feature, you can define a hierarchical set of groups to which you can assign users.

Initially, you may have a number of predefined users that you would like to set up. As part of this process, you will want to assign these people to one or more of the groups you have set up. This project details how to build groups and users into your portal.

1. Log on to the WebLogic Commerce Server Administration Tool.

2. Navigate into the User Manager.
3. Click Create on the Users banner to create a number of new users.
4. Return to the User Manager and create a number of new groups by clicking Create from the Groups banner.
5. Return to the User Manager and click the text "Groups" from the Groups banner, which will take you to the Group editor. Edit the groups you have created, and in doing so assign some users to each of the groups.

   **Note:** Before going to the next step, add a reference to your portal group in the deployment description.

   Do this by adding a `<principal-name>new_group</principal-name>` entry to the `<security-role-assignment>` segment of the `weblogic.xml` file in the following directory:

   ```
   WL_COMMERCE_HOME/config/wlcsDomain/applications/wlcsApp/examp
   leportal/WEB-INF
   ```

   You will need to restart the server and logon as a new user before continuing.

   For more information, see the chapter "Deployment Descriptors and Security Roles" in the *Security Guide*.
6. Finally, go to the Portal Manager and edit your custom portal. In the Associated Groups section, add your new groups to your portal.

For more information, see Chapter 3, "Using the Portal Management Administration Tool," in this guide.

# Writing Your Own Portlets

"Creating the Framework for Your Custom Portal" on page 5-6 showed you how to get up and running with your custom portal by copying the example portal. "Simple Customizations" on page 5-16 showed you how to alter the logo images and use the Administration Tool to customize your portal  Now, this section will show you how to build your own functionality into your custom portal.

In this section, you will see how to build static portlets and portlets that change based on state information retrieved from WebLogic Personalization Server.

Once you have mastered the essence of portlet writing, you will learn about advanced functionality. Topics such as maximized portlets and inter-portlet communication will be presented in the next section, "Advanced Portlet Functionality" on page 5-30.

**Note:** In the following projects, you will be registering your portlets and adding them to your portal using the Administration Tool. When doing these activities or when editing the portlet definition after creation, you may not see the changes to your portal in your browser even after flushing the browser cache. This is due to server-side caching. In order to force a rebuild of your portal page, you must log in if you are logged out, or log out if already logged in to see your changes take effect.

# Project 5: Building a Static Portlet

The fundamentals of portlet writing are not difficult. As you will see, the construction of a static portlet is quite easy. The complexity of portlets come when they become dynamic. The first portlet project is a static portlet.

When you view your portal, your browser is displaying an HTML page to you. If you "View Source" on your portal, you will not see any JavaServer Page tags, although JSP was used to author the document. This is because the server processed the JSP input and output the HTML to your browser. In exactly the same way, before your portlet is placed on the portal page, it is processed and converted into HTML if it was not already. With this in mind, think of a portlet as just a small Web page.

For this static portlet project, you are not going to use a JSP. You will build just an HTML fragment that will get included into the portal page. What follows is your first portlet.

## welcome.html

```
<p align=center>
<h1>Welcome!</h1><br>

This portal contains the projects built by following the WLCS
tutorial.
</p>
```

Students of HTML will recognize this as just a simple static HTML fragment. There really is nothing special about this HTML.

To make the HTML fragment above into a portlet, follow these steps:

1. With your filesystem navigation tool, navigate to your custom portal folder. Then, enter the `portlets` subfolder. Here, create a new file called `welcome.html` with a text editor. This HTML file you just created will hold your first portlet.

   **Note:** Windows users using WordPad or Microsoft Word should Save As a text document; otherwise, extra unwanted characters will be dumped into the text stream.

2. Copy the HTML fragment above into `welcome.html`. This file should contain no other text. Save this file to disk.

3. Log into the WebLogic Commerce Server Administration Tool. (For more information on completing this step, see "Logging On to the Administration Tool" on page 3-2.)

4. Click the blue and red monitor icon on the Portal Management banner. This will take you into the Portal Manager.

5. Click the Create button on the Portlets banner. This will allow you to register your new portlet with the server.

   **Note:** Be sure to click the Create button on the Portlet banner and not the one on the Portal banner.

6. You are presented with a form. There are two required fields that you must fill in, the rest you should leave with the defaults.

   a. For Portlet Name, enter `Welcome`.

   b. For Content URL, enter `portlets/welcome.html`.

   Click Create, and then click Back. You have now successfully registered your new portlet with the server.

7. Add the portlet to your custom portal and make it visible. For more information about this step, see "Adding and Removing Portlets" on page 3-17.

8. Use your browser to display the first page of your custom portal, following the procedure described in "View Your Portal Site in a Browser" on page 5-12. You should see your Welcome portlet.

You have now added new functionality to WebLogic Personalization Server.

# Project 6: Building a Simple Dynamic Portlet

Now that you see how easy it is to build portlets, your next step is to add some dynamic behavior to your portlet. For this, you will need to create a JavaServer Page file, not HTML. With the power of JSP, you can query the WebLogic Personalization Server for information, and vary the output depending on the results of your queries.

In this project, you will build a portlet that will detect if the user of the browser is logged on to the portal. If not, this portlet will display a message to the user, asking the user to log on. If the user is logged on, the portlet will instead display the user's login name.

You will also accomplish the dynamic functionality by using methods included in the portlet JSP base class. All portlet JSPs should extend `com.beasys.commerce.portal.admin.PortalJspBase`.
This class contains many convenience methods which perform general tasks for your portlet JSP page, such as accessing session information and user login information. To achieve this, begin your portlet JSP files with the following line:

```
<%@ page
extends="com.beasys.commerce.portal.admin.PortalJspBase"%>
```

Once you have extended `PortalJSPBase`, you have access to many methods from your JSP file, including `getLoggedIn()` and `getSessionValue()`.To understand what these methods do, look at the following JSP fragment.

## isloggedon.jsp

```
<%@ page extends="com.beasys.commerce.portal.admin.PortalJspBase"
%>

<p>
<%
    // getLoggedIn() returns true if the user is logged in
    if (getLoggedIn(request))
    {
%>
       You are currently logged in as
       <%= getSessionValue(
com.beasys.commerce.axiom.jsp.JspConstants.SERVICEMANAGER_USER,
```

```
                request)
        %>
        . Please make yourself at home.
<%
    }
    else
    {
%>
        You are not currently logged on. Please click the
        key icon at the top right-hand corner of the page to
        log onto this site.
<%
    }
%>
</p>
```

This code is the complete text for your first dynamic portlet.

To implement this project, follow these steps:

1. With your filesystem navigation tool, navigate to your custom portal folder. Then, enter the `portlets` subfolder. Here, create a new file called `isloggedon.jsp` with a text editor. This JSP file you just created will hold your first dynamic portlet.
2. Copy the JSP fragment above into `isloggedon.jsp`. This file should contain no other text. Save this file to disk.
3. Log into the WebLogic Commerce Server Administration Tool. For more information about this step, see "Logging On to the Administration Tool" on page 3-2.
4. Click the blue and red monitor icon on the Portal Management banner. This will take you into the Portal Manager.
5. Click the Create button on the Portlets banner. This will allow you to register your new portlet with the server.
6. You are presented with a form. There are two required fields that you should fill in, the rest you should leave with the defaults.

   a. For Portlet Name, enter `Logged on?`

   b. For Content URL, enter `portlets/isloggedon.jsp`.

   Click Create, then click Back. You have now successfully registered your new portlet with the server.

7. Add the portlet to your custom portal and make it visible. For more information about this step, see “Adding and Removing Portlets” on page 3-17.
8. Use your browser to display the first page of your custom portal, following the procedure described in “View Your Portal Site in a Browser” on page 5-12. You should see your “Logged on?” portlet. See how the text in the portlet changes depending on whether you are logged into the portal or not.

You have now finished your first dynamic portlet.

# Project 7: Building a Dynamic Portlet Using JSP Tags

In Project 6, you built a simple dynamic portlet using functionality provided by extending PortalJSPBase. Take some time to look at other functionality provided by PortalJSPBase. Once you have done this, you are ready to begin working with another dynamic technique available to your JavaServer Page portlets, JSP tags.

Included with the WebLogic Personalization Server is a set of tag libraries that enable your JSPs access to the full power of the personalization engine. The tag libraries for personalization are as follows:

- Personalization: uri = “pz.tld”
- Content Management: uri = “cm.tld”
- User Management: uri = “um.tld“
- Property Sets: uri = “ps.tld”
- Internationalization: uri = “i18n”
- Personalization Utilities: uri = “es.tld”

For more information, see the JSP Tag Library Reference chapter in the *Guide to Building Personalized Applications*.

Additionally, a JSP tags for managing portals can be found in the tag library:

- Portal Management: uri = “esp.tld“

For more information, see Chapter 8, “Portal Management JSP Tag Library Reference,” in this guide.

For full details on how tag libraries work in the JSP language, consult a JavaServer Page handbook. After viewing this sample tag library portlet, you will see that tag libraries are quite easy to use.

Each one of these tag libraries supports a number of tags. In this project, you will use tags from the User Management <um:> and Personalization Utilities <es:> tag libraries. This portlet will output the name of all users of your portal. Next to each username, it will output the e-mail address of that user. A detailed description of how this code works is not provided here. Hopefully, reading the code and consulting the *Guide to Building Personalized Applications* is sufficient.

One point about the code should be made. You will notice that for every username retrieved by calling <um:getUsersnames>, there is a call to <um:getProfile>. It is necessary to explain why this line of code is needed. At any time during the processing of a portlet, exactly one user profile is in scope. Calls like <um:getProperty> and <um:setProperty> refer to the user profile in scope. In this project the code must iterate through the list of usernames and query the profile associated with each user. Therefore, before a call to <um:getProperty> is made, the profile for the user must be loaded into scope by calling <um:getProfile>. And at the end of this JSP, the original user profile must be loaded back into scope to avoid causing problems with other portlets.

## EmailList.jsp

```
<%-- include the tag libraries we need --%>
<%@ taglib uri="es.tld" prefix="es" %>
<%@ taglib uri="um.tld" prefix="um" %>

<%-- extend PortalJSPBase to get some base functionality --%>
<%@ page
       extends="com.beasys.commerce.portal.admin.PortalJspBase"
%>

<%
    // get the name of the current user
    String originalUserName = (String)getSessionValue(
com.beasys.commerce.axiom.jsp.JspConstants.SERVICEMANAGER_USER,
        request);
    if ( originalUserName == null ) originalUserName = "";

    // get the name of the portal
    String portalName =
       (String)getSessionValue(PORTAL_NAME, request);
    if ( portalName == null ) portalName = "";
%>
```

```
<%-- ask WLCS to put a list of the user names in string array
"userNameList" --%>
<um:getUsernames id="userNameList" result="namesResult"/>

<table border=1 cellspacing=1 align="center">

<tr>
       <th colspan=2>Portal Users</th>
</tr>

<es:forEachInArray id="curUser" type="String"
              array="<%=userNameList%>"
              counterId="curIndex">

<tr>
       <%-- This section is evaluated once for
              every user in userNameList --%>

       <%-- Output the name of the user for this row --%>
       <td><%=curUser%></td>

       <%-- Output the email address of the curUser --%>
       <td>

              <um:getProfile profileKey="<%=curUser%>"
                      scope="request"
              />
              <um:getProperty id="email"
                     propertySet="<%=portalName%>"
                     propertyName="<%=PROFILE_EMAIL%>"
              />
              <%=email%>
       </td>
</tr>
</es:forEachInArray>

<%
if (getLoggedIn(request)) {
%>

       <um:getProfile
       profileKey="<%=originalUserName%>" scope="request" />

<%
}
%>

</table>
```

This code is the complete text for your second dynamic portlet.

To implement this project, follow these steps:

1. With your filesystem navigation tool, navigate to your `custom portal` folder. Then, enter the `portlets` subfolder. Here, create a new file called `EmailList.jsp` with a text editor. This JSP file you just created will hold your first dynamic portlet.
2. Copy the JSP fragment above into `EmailList.jsp`. This file should contain no other text. Save this file to disk.
3. Log into the WebLogic Commerce Server Administration Tool. For more information about this step, see "Logging On to the Administration Tool" on page 3-2.
4. Click the blue and red monitor icon on the Portal Management banner. This will take you into the Portal Manager.
5. Click the Create button on the Portlets banner. This will allow you to register your new portlet with the server.
6. You are presented with a form. There are two required fields which you should fill in, the rest you should leave with the defaults.

   a. For Portlet Name, enter `Email List`.

   b. For Content URL, enter `portlets/EmailList.jsp`.

   Click Create, then click Back. You have now successfully registered your new portlet with the server.
7. Add the portlet to your custom portal and make it visible. For more information about this step, see "Adding and Removing Portlets" on page 3-17.
8. Use your browser to display the first page of your custom portal, following the procedure described in "View Your Portal Site in a Browser" on page 5-12. You should see your "Email List" portlet.
9. Now add some more users to your portal. Do this by clicking the key icon in the upper right-hand corner of your portal. This will bring up the sign-on page. Under the New User banner, click Create. Follow the instructions to add new users.
10. Return to your portal page and see the new users appear in the portlet.

You have now seen the three major techniques for building a portlet: static HTML, dynamic behavior based on extending PortalJSPBase, and dynamic behavior based on the WebLogic Personalization Server tag libraries.

# Advanced Portlet Functionality

In the previous section, you learned how to build portlets. This section continues with portlets and demonstrates how to use more portlet features.

## Project 8: Adding a Maximized URL

This project will walk you through how to build a maximized version of your portlet. In the default case, your portlet cannot be maximized. If you allow your portlet to be maximized but do not provide a maximized URL, WebLogic Personalization Server will simply use your normal-sized portlet content when maximized. In most cases, you will want to take advantage of the extra space afforded by being maximized and alter your portlet content to include more information. This project shows how to do this. It assumes you completed "Project 7: Building a Second Dynamic Portlet". You will not change the portlet created in that project, but you will add a new maximized JSP.

At the portal definition level, setting the maximizable attribute establishes the initial value for the portal. If you set max/min on a portlet, it overrides the max/min setting for the portal.

The first step of this project is to create the new portlet content JSP for the maximized state. Since in the maximized state your portlet has more screen real estate, you can display more information. In this case, for each user displayed, you will show more columns of information. The following is the maximized JSP:

### EmailListMax.jsp

```
<%-- include the tag libraries we need --%>
<%@ taglib uri="es.tld" prefix="es" %>

<%@ taglib uri="um.tld" prefix="um" %>

<%-- extend PortalJSPBase to get some base functionality --%>
<%@ page
extends="com.beasys.commerce.portal.admin.PortalJspBase"
%>

<%
       // get the name of the current user
```

```
        String originalUserName = (String)getSessionValue(
com.beasys.commerce.axiom.jsp.JspConstants.SERVICEMANAGER_USER,
                request);
        if ( originalUserName == null ) originalUserName = "";

        // get the name of the portal
        String portalName =
        (String)getSessionValue(PORTAL_NAME,request);
        if ( portalName == null ) portalName = "";
%>

<%-- ask WLCS to put a list of the user names in
        string array "userNameList" --%>
<um:getUsernames id="userNameList" result="listresult" />

<table border=1 cellspacing=1 align="center">

<tr>
        <th colspan=8>Portal Users</th>
</tr>

<es:forEachInArray id="curUser" type="String"
        array="<%=userNameList%>" counterId="curIndex">

<tr>
        <%-- This section is evaluated once for
                every user in userNameList --%>

        <%-- Output the name of the user for this row --%>
        <td><%=curUser%></td>

        <um:getProfile profileKey="<%=curUser%>" scope="request" />


        <%-- Output the email address of the curUser --%>

        <td>
                <%-- Load curUser's profile into scope --%>
                <um:getProperty id="email"
                        propertySet="<%=portalName%>"
                        propertyName="<%=PROFILE_EMAIL%>"/>
                <%=email%>
        </td>

        <%-- Output the first name of the curUser --%>
        <td>
                <um:getProperty id="first"
                        propertySet="<%=portalName%>"
                        propertyName="<%=PROFILE_FIRST%>"/>
```

```
                <%=first%>
            </td>

            <%-- Output the last name of the curUser --%>
            <td>
                <um:getProperty id="last"
                    propertySet="<%=portalName%>"
                    propertyName="<%=PROFILE_LAST%>"/>
                <%=last%>
            </td>

            <%-- Output the address of the curUser --%>
            <td>
                <um:getProperty id="address"
                    propertySet="<%=portalName%>"
                    propertyName="<%=PROFILE_ADDRESS%>"/>
                <%=address%>
            </td>

            <%-- Output the city of the curUser --%>
            <td>
                <um:getProperty id="city"
                    propertySet="<%=portalName%>"
                    propertyName="<%=PROFILE_CITY%>"/>
                <%=city%>
            </td>

            <%-- Output the state of the curUser --%>
            <td>
                <um:getProperty id="state"
                    propertySet="<%=portalName%>"
                    propertyName="<%=PROFILE_STATE%>"/>
                <%=state%>
            </td>

            <%-- Output the zip code of the curUser --%>
            <td>
                <um:getProperty id="zip"
                    propertySet="<%=portalName%>"
                    propertyName="<%=PROFILE_ZIP%>"/>
                <%=zip%>
            </td>
        </tr>
        </es:forEachInArray>

        <%
        if (getLoggedIn(request)) {
        %>
            <um:getProfile profileKey="<%=originalUserName%>"
                scope="request" />
```

```
<%
}
%>
</table>
```

To specify the above code as your maximized portlet content, follow these steps:

1. With your filesystem navigation tool, navigate to your custom portal folder. Then, enter the `portlets` subfolder. Here, create a new file called `EmailListMax.jsp` with a text editor. This JSP file you just created will hold your maximized portlet code.

2. Copy the JSP fragment above into `EmailListMax.jsp`. This file should contain no other text. Save this file to disk.

3. Log into the WebLogic Commerce Server Administration Tool. For more information about this step, see "Logging On to the Administration Tool" on page 3-2.

4. Click the blue and red monitor icon on the Portal Management banner. This will take you into the Portal Manager.

5. Under the Portlets banner, click the EmailList portlet you created in "Project 7: Building a Dynamic Portlet Using JSP Tags" on page 5-25.

6. You will be presented with a form.

   a. Change the state of the "Maximizable" check box so that it is checked.

   b. In the "Maximized URL" edit field, enter the `EmailListMax.jsp` file you created in step 1 (in this case: `portlets/EmailListMax.jsp`).

   Click Save, then click Back.

7. Open another browser window and navigate to your portal, following the procedure described in "Setting Up the Portal Framework" on page 5-7. Your portlet should be visible, and will have a square-like icon on its titlebar. This indicates that it is maximizable.

   **Note:** Remember to log out and log in to force the server to rebuild the portal page.

8. Click the maximize icon, which will cause the portlet to be maximized. The `EmailListMax.jsp` page will load, listing each user, plus more information per user than what is displayed when the portlet is of normal size.

# Project 9: Changing the Look of a Maximized Portlet

Project 9 assumes that you have completed "Project 8: Adding a Maximized URL." The goal of this example is to specify a non-default header and footer to be used when your portlet is maximized. In Project 8, the portlet used the provided `alternateheader.jsp` and `alternatefooter.jsp` since you did not override the defaults. In some cases the defaults are sufficient, but for the purposes of demonstration this project will define its own.

## EmailListMaxHeader.jsp

```
<p>
<h1>Email List Portlet</h1>
<hr/>
</p>
```

## EmailListMaxFooter.jsp

```
<p>
<hr/>
<i>Creating a Custom Portal</i> Tutorial
</p>
```

These code examples are just simple samples. You may elaborate on the design if you wish.

To replace the alternate header and footer with the code above, follow these steps:

1. With your filesystem navigation tool, navigate to your custom portal folder. Create two new files called `EmailListMaxHeader.jsp` and `EmailListMaxFooter.jsp` with a text editor. This JSP files you just created will hold the header and footer for your maximized portlet.

   **Note:** Be sure to create these files in your portal's root folder and not in your portlets subfolder.

2. Copy the first JSP fragment above into `EmailListMaxHeader.jsp`, and the second into `EmailListMaxFooter.jsp`. These files should contain no other text. Save these files to disk.

3. Log into the WebLogic Commerce Server Administration Tool. For more information about this step, "Logging On to the Administration Tool" on page 3-2.

4. Click the blue and red monitor icon on the Portal Management banner. This will take you into the Portal Manager.

5. Under the Portlets banner, find the EmailList portlet you created in Project 7 and click on it.

6. You will presented with a form.

   a. In the Alternate Header URL, enter `EmailListMaxHeader.jsp`.

   b. In the Alternate Footer URL, enter `EmailListMaxFooter.jsp`.

   Click Save, then click Back.

7. Open another browser window and navigate to your portal. Your portlet should be visible, and will have a square-like icon on its titlebar. Click the icon, which will cause the portlet to be maximized.

8. `EmailListMax.jsp` will load, along with `EmailListMaxHeader.jsp` and `EmailListMaxFooter.jsp`.

   **Note:** Remember to log in and log out to force the server to rebuild the portal page.

This concludes the customization of your portlet's maximized state. You may also experiment with creating an editable version of your portlet, and even a help window associated with your portlet.

# Project 10: Inter-portlet Communication

In some instances, you will want to have actions in one portlet affect the display of another portlet. This is called inter-portlet communication. To implement this functionality, imagine writing Java Beans or database calls where one portlet persists data and another reads the data. This would work, but requires more effort than necessary. Unless you need to pass large amounts of data between portlets, you should follow a simpler approach. This approach is demonstrated here.

Before you read the code, you must understand that there is an object that is shared between the Portal Framework and all of the portlets. This object can be used to pass data between all of these entities. This object is the HTTP request. You may set parameters in the request in one portlet, then forward the request back to the portal, which will ultimately forward the request to all portlets. The end result is that the HTTP request can serve as a message passing mechanism for portlets. This project shows you how to exploit this.

The goal of this project is to create two portlets. The first portlet, called "User Index," displays a ranges of usernames in the system. For example, it will show "Allen to Carl, Chuck to Elmer, Francis to Irene,..." If the user clicks on a name range, the second portlet will refresh and show detailed information about each username in that range. The second portlet is called "User Index Details."

The following is the code for both portlets. Take special note of how the "href" is constructed in `UserIndex.jsp`, and how the parameters are retrieved in `UserIndexDetails.jsp`.

## UserIndex.jsp

```
<%-- include the tag libraries we need --%>
<%@ taglib uri="es.tld" prefix="es" %>
<%@ taglib uri="um.tld" prefix="um" %>

<%-- extend PortalJSPBase to get some base functionality --%>
<%@ page
 extends="com.beasys.commerce.portal.admin.PortalJspBase"%>

<%
    // get the name of the current user
    String originalUserName = (String)getSessionValue(
com.beasys.commerce.axiom.jsp.JspConstants.SERVICEMANAGER_USER,
        request);
    if ( originalUserName == null ) originalUserName = "";

    // get the name of the portal
    String portalName = (String)getSessionValue(PORTAL_NAME,
        request);
    if ( portalName == null ) portalName = "";
%>

<%-- ask WLCS to put a list of the user names in string
        array "userNameList" --%>
<um:getUsernames id="userNameList" result="listresult" />
```

```
<table border=1 cellspacing=1 align="center">

<tr>
       <th>Portal Users Index</th>
</tr>

<%
int divisor = 5;
boolean isRowTerminated = true;
String persistCurUser = null;
%>

<es:forEachInArray id="curUser" type="String"
       array="<%=userNameList%>" counterId="curIndex">
       <%-- This section is evaluated once for every user in
              userNameList --%>

<%
       persistCurUser = curUser;

       // start the cell if this is the first user in a range
       if (curIndex.intValue()%divisor == 0)
       {
              // beginning of range
              isRowTerminated = false;
%>
              <%-- THIS IS WHERE THE DATA IS PASSED --%>
              <tr><td>
              <a href="<%=response.encodeURL(createURL(request,
                     getHomePage(request),
                     ("userIndexStartIndex=" + curIndex
                     + "&userIndexDivisor=" + divisor
                     )))%>"
              >
              <%=curUser%> to
<%
       }

       // finish cell if this is the last user in range or
       // the last user in the list
       if (curIndex.intValue()%divisor == divisor-1)
       {
              // end of range
              isRowTerminated = true;
%>
              <%-- Output the name of the user for this row --%>
              <%=curUser%></a></td></tr>
<%
       }
```

```
%>
</es:forEachInArray>
<%
       if (!isRowTerminated)
       {
              // terminate the row if it ended on a
              // non-divisor boundary
%>
              <%-- Output the name of the user for this row --%>
              <%=persistCurUser%></a></td></tr>
<%
       }
%>
</table>
```

## UserIndexDetails.jsp

```
<%-- include the tag libraries we need --%>
<%@ taglib uri="es.tld" prefix="es" %>
<%@ taglib uri="um.tld" prefix="um" %>

<%-- extend PortalJSPBase to get some base functionality --%>
<%@ page
extends="com.beasys.commerce.portal.admin.PortalJspBase"%>

<%
    // get the name of the current user
    String originalUserName = (String)getSessionValue(
com.beasys.commerce.axiom.jsp.JspConstants.SERVICEMANAGER_USER,
                request);
    if ( originalUserName == null ) originalUserName = "";

    // get the name of the portal
    String portalName =
              (String)getSessionValue(PORTAL_NAME, request);
    if ( portalName == null ) portalName = "";
%>

<%
// GET THE PARAMETERS PASSED IN, if they exist

       int startIndex = 0;
       String startIndexString =
                     request.getParameter("userIndexStartIndex");
       if (startIndexString != null)
       {
              try
              {
```

```
                        startIndex =
                                Integer.parseInt(startIndexString);
                }
                catch (Exception e) {
                        System.out.println("UserIndexDetail.jsp - "+
                        "startIndex parse error: "+startIndexString);
                }
        }

        int divisor = 0;
        String divisorString =
                         request.getParameter("userIndexDivisor");
        if (divisorString != null)
        {
                try
                {
                        divisor = Integer.parseInt(divisorString);
                }
                catch (Exception e) {
                        System.out.println("UserIndexDetail.jsp - "+
                        " divisor parse error: "+divisorString);
                }
        }

        if (divisor == 0)
        {
%>
        Click a name range in the User Index portlet to display
        information about each user in the range.
<%
        }
        else // divisor !=0
        {
%>

<%-- ask WLCS to put a list of the user names in string array
        "userNameList" --%>
        <um:getUsernames id="userNameList" result="listresult" />

        <table border=1 cellspacing=1 align="center">

        <tr>
        <th colspan=2>Portal Users</th>
        </tr>

        <es:forEachInArray id="curUser" type="String"
                array="<%=userNameList%>" counterId="curIndex">
<%
        if ((curIndex.intValue() >= startIndex) &&
```

```
         (curIndex.intValue() < startIndex+divisor))
        {
%>
        <tr>
        <%-- This section is evaluated once for every user in
               userNameList --%>

        <%-- Output the name of the user for this row --%>
        <td><%=curUser%></td>

        <%-- Output the email address of the curUser --%>
        <td>
               <um:getProfile profileKey="<%=curUser%>"
                      scope="request" />
               <um:getProperty id="email"
                propertySet="<%=portalName%>"
                propertyName="<%=PROFILE_EMAIL%>"
               />
               <%=email%>
        </td>
        </tr>
<%
        }
%>
        </es:forEachInArray>

<%
        if (getLoggedIn(request)) {
%>
               <um:getProfile profileKey="<%=originalUserName%>"
                      scope="request" />
<%
        }

        } // from else clause of if (divisor == 0)
%>

</table>
```

To use the above code as your portlets, follow these steps:

1. With the filesystem navigation tool, navigate to your custom portal folder. Navigate into your `portlets` subfolder. Here, create two new files called `UserIndex.jsp` and `UserIndexDetails.jsp` with a text editor. These JSP files you just created will hold the two portlets that will communicate with each other.

2. Copy the first JSP fragment above into `UserIndex.jsp`, and the second into `UserIndexDetails.jsp`. These files should contain no other text. Save these files to disk.

3. Log into the WebLogic Commerce Server Administration Tool. For more information about this step, see "Logging On to the Administration Tool" on page 3-2.

4. Click the blue and red monitor icon on the Portal Management banner. This will take you into the Portal Manager.

5. Click the Create button on the Portlets banner. This will allow you to register your new portlets with the server.

6. You are presented with a form. There are two required fields which you should fill in, the rest you should leave with the defaults.

   a. For Portlet Name, enter `User Index`.

   b. For Content URL, enter `portlets/UserIndex.jsp`.

   Click Create. You have now successfully registered your first new portlet with the server.

7. You still are presented with the form. You should now overwrite the values you entered in step 6.

   a. For Portlet Name, enter `User Index Details`.

   b. For Content URL, enter `portlets/UserIndexDetails.jsp`.

   Click Create. You have now successfully registered your second new portlet with the server.

8. Add the portlets to your custom portal and make them visible. For more information about this step, see "Adding and Removing Portlets" on page 3-17.

9. Test your portlets by directing your browser to your portal. Use your browser to display the first page of your custom portal, following the procedure described in "View Your Portal Site in a Browser" on page 5-12. You should see your "User Index" and "User Index Details" portlets.

10. Now add some users to your portal. Do this by clicking on the key icon in the upper right-hand corner of your portal. This will bring up the sign-on page. Under the New User banner, click Create. Follow the instructions to add new users.

11. Go back to your main portal page. Notice that username ranges are displayed in the User Index portlet. Click on one of the ranges. Notice how the User Index Details portlet now displays detailed information on the users in the name range you selected.

You have now completed the advanced portlet creation projects.

# Using the HTTP Request Method to Communicate Between Portlets

There are two issues that you should be aware of when using the HTTP request method to communicate between portlets.

## Parameter Name Collisions Between Portlets

Because the HTTP request is broadcast to all portlets within the portal, you must be careful to avoid a parameter name collision between portlets. For example, suppose a portlet appends a URL parameter such as `src=/usr/local/src` to the anchor tags that it generates. If other portlets look for such a parameter, then they will all find it when the user clicks the link. Unless all portlets looking for that parameter are interpreting it the same way, confusing or bad parameters can be passed to the receiving portlets.

To avoid name collisions, remember that the URL parameter names that get encoded by using the URL string are global in nature.

## Several Sets of Portlets Using the Http Request Method at Once

When a set of portlets communicates, the code does the following:

```
<a href="<%response.encodeURL(createURL(request,
getHomePage(request),...and so on.
```

When the `getHomePage()`method is called, it strips off any parameters that were passed in from the last request. This is desirable when a single set of portlets communicates using this method. However, if an unrelated set of portlets is employing the same technique, then only one set can be “active” at any one time.

For example, one set of portlets (group A) includes a headline browser portlet and a news story display portlet. When a user clicks a headline, the headline browser uses the code above to generate a URL. The URL includes a parameter that tells the story display portlet where to find the story. When these are the only two portlets using this method, it works correctly.

Now lets add a second set of portlets to the example. Portlet group B includes a stock quote portlet and a stock detail portlet.When a user clicks a stock quote, the portlet uses the code above to generate a URL. The URL includes a parameter that tells its partner portlet where to find the stock details. Again, this method works correctly when just one set of portlets is communicating. However, what happens when both sets of portlets are using the `getHomePage()`method?

In this scenario, first the user clicks a story headline to see the story. The story portlet reads the parameter from the URL and displays the story. Then the user clicks a stock symbol to get detailed stock information, and the stock quote portlet generates a new anchor tag using a stock related parameter. The stock details display in the partner stock portlet just fine, but what happens in the story portlet?

When the user clicks the stock symbol, this second event notifies all the portlets to poll again. The URL generated by the stock quote portlet overwrote the news story parameter with its own stock related parameter, and so the new URL contains no parameters related to the story. If the story portlet does not find the parameter it expects, it may just display a blank portlet page. You can easily avoid this problem using the session cache. If the story portlet has already seen a parameter telling it where to find the story, it could use the session cache to default to the previous story location if there is no new information.

# Other Customization Techniques

This tutorial has introduced you to the power of WebLogic Personalization Server and the Portal Framework. With the techniques described in the previous sections, you can now build a sophisticated portal.

However, there are more ways in which you can build your customized application. Although this chapter will not cover these additional techniques in detail, this section discusses each technique and provides pointers to other documentation that contains more information.

## More Portlet Customization

In Projects 8 and 9, you created a maximized version of your portlet. In the same manner, you may create an editable version of your portlet. You may also specify a help page for your portlet, mandate that it appear for all users, allow/disallow it to be minimized, make it floatable, and make changes to the titlebar appearance.

Editing the portlet's properties via the Associated Portlets link is only necessary if you wish to override the default values used to create the portlet.

For more information about the Portal Management Administration Tools, see Chapter 3, “Using the Portal Management Administration Tool.”

## Database Interaction

In many cases you will want to persist data or retrieve persisted data from within your portal and portlets. In most cases, you will be using a database for this purpose. WebLogic Personalization Server provides simple connectivity with your database via Personalization Utility tags. Specifically, refer to the `<es:simpleReport>` tag in the JSP Tag Library Reference chapter of the *Guide to Building Personalized Applications*.

Alternatively, enterprise applications will likely want to use Enterprise Java Beans for data persistence. WebLogic Commerce Server provides full support for the EJB specification.

For more information, consult the WebLogic Server documentation.

## Java Beans Interaction

The Java Beans technology provides an easy way to remove the bulk of your code from your JavaServer Page files. This will free your JSP files from clutter, and make that code more maintainable and reusable. WebLogic Commerce Server provides full support for calling Java Beans from your JSP files.

For more information, consult a JavaServer Page handbook.

## Personalization Advisor Functionality

The Advisor delivers content to a personalized application based on a set of rules and user profile information. It can retrieve any type of content from a Document Management system and display it in a JSP or use it in a servlet. The Advisor component uses the Personalization JSP tag library and an Advisor EJB (stateless session bean) to access the WebLogic Personalization Server's core personalization services and deliver personalized content.

For more information, consult the chapter Creating Personalized Applications with the Advisor in the *Guide to Building Personalized Applications*.

## Internationalization

WebLogic Personalization Server provides a simple framework that allows access to localized text labels and messages. The internationalization (I18N) framework is accessible from JSP files through a small I18N tag library.

For more information, consult the chapter Creating Localized Applications with the Internationalization Tags in the *Guide to Building Personalized Applications*.

# Using Webflow

In WebLogic Personalization Server, Webflow is a feature that allows you to string together JSP files, input processors (IPs) and pipeline processors (PPs) without hard coding the linkage between them. Instead, the linkage is defined in an external Webflow properties file.

If you are considering using Webflow within the Portal Framework, see "Using Webflow Within a Portal" on page 6-2.

# Commerce Functionality

The process customers go through when making a purchase from your Web site is one of the most common but complex aspects of an e-business. To help you get to market faster than your competitors, the WebLogic Commerce Server product provides you with an Order Processing package. This package contains default implementations for the most common e-business order-related services (such as shopping cart management, taxation, payment, and so on). Designed to be used out-of-the-box, the Order Processing package allows your site designers to customize the order process without the need for advanced programming skills. Additionally, it is easily extensible for those with advanced technical knowledge.

For more information, consult the *Guide to Processing Orders and Managing Purchases.*

# Modifying the Portal Framework

In some cases,you may find that the Portal Framework is generally suitable for your needs, but some aspects of it need to be modified. A valid option in this case is to actually modify the JSP files in the repository folder. You are encouraged to use the Portal Framework files in any way to help you get to market quickly. Be aware that some changes you make may be inconsistent with the Administration Tool; you will need to implement your own administration functionality in those cases.

To do this, you will need to read and understand the contents of the JSP files located in the repository folder. Therefore, you will need to be comfortable with JSP. Also, you will need to be comfortable with the WebLogic Personalization Server custom JSP tags and provided classes such as `PortalJSPBase`.

For more information, see the JSP Tag Library Reference in the *Guide to Building Personalized Applications*

## Building Your Site Without the Portal Framework

You may not want a portal metaphor when creating your site. Or, you may find that your site design would require extensive changes to the Portal Framework. In both cases, you may choose to build your site from scratch. In this case, you have the full power of JSP and HTML at your disposal, as well as the commerce components (shopping cart management, taxation, payment) and the personalization components (rules, property sets, content management, user and group management) via the JSP tag libraries.

For more information, see the JSP Tag Library Reference in the *Guide to Building Personalized Applications* and the WebLogic Commerce Server documentation.

# Framework Files

Table 5-1 displays the names and functions of the template JSP files provided with the WebLogic Personalization Server framework. Each of these files is located in the root directory of the portal which it serves, such as `/portals/repository`.

**Table 5-1 Framework Templates**

| JSP Filename | Function |
|---|---|
| `_user_add_portlets.jsp` | The tool employed by the end user to add/remove portlets. |
| `_user_layout.jsp` | The tool employed by the end user to update portlet layout. |
| `_userlogin.jsp` | The user login page. |

**Table 5-1 Framework Templates (Continued)**

| JSP Filename | Function |
|---|---|
| `_userreg.jsp` | The new user registration page. |
| `_userreg_summary.jsp` | The user profile summary page. |
| `alternatefooter.jsp` | The footer displayed when a portlet is maximized or detached. |
| `alternateheader.jsp` | The header displayed when a portlet is maximized or detached. |
| `baseheader.jsp` | A stripped version `header.jsp`, intended for general use beyond the portal home page. |
| `color_picker.jsp` | The color palette employed by the user color preferences tool. |
| `error.jsp` | A general-purpose page used for displaying run-time errors. |
| `error_footer.jsp` | The footer displayed with `error.jsp`. |
| `error_header.jsp` | The header displayed with `error.jsp`. |
| `footer.jsp` | The footer displayed with the main portal page. |
| `fullscreenportlet.jsp` | The page used to display a maximized or detached portlet. |
| `gen_prefs.jsp` | The tool employed by the end user to update general user profile information. |
| `header.jsp` | The header displayed with the main portal page. |
| `help.jsp` | The end user help page. |
| `layout_script.jsp` | The JavaScript used by the end user layout tool. |
| `portal.jsp` | The main portal page. |
| `portalcontent.jsp` | The page which prescribes portlet layout within the main portal page. |
| `portalerror.jsp` | The default error page displayed when an access attempt to a portal page fails. |
| `portalnotexist.jsp` | The page which displays a general message indicated that the requested portal does not exist. |
| `portlet.jsp` | The page which constructs a portlet, combining portlet titlebar, banner, header, content, and footer. |

**Table 5-1 Framework Templates (Continued)**

| JSP Filename | Function |
|---|---|
| `privacy_policy.jsp` | A placeholder for a company privacy policy statement. |
| `status.jsp` | The page used to display end user status messages. |
| `suspended.jsp` | The page which provides a message indicating that the requested portal is currently non-operational, typically for maintenance reasons. |
| `titlebar.jsp` | The portlet titlebar. Contains appropriate portlet icons and portlet name. |
| `user_colors.jsp` | The end user color preferences tool. |

# 6 Advanced Portal Topics

This chapter describes how to deploy your portal as a Web application and use Webflow within a portal.

This topic includes the following sections:

- Deploying New Portals as Web Applications
- Using Webflow Within a Portal
- Cache Control in the Portal

**Note:** Throughout this chapter, the environment variable `WL_COMMERCE_HOME` is used to indicate the directory in which you installed the WebLogic Commerce Server and WebLogic Personalization Server software.

# Deploying New Portals as Web Applications

Beginning with WebLogic Server version 6.0, all public HTML-based applications must be deployed as Web applications. The sample applications shipped with this product are all Web applications and serve as examples to follow.

## Building a New Portal Web Application Using the Portal Framework

In the following steps, you will create a new portal called 'myAcme' within the `wlcsApp` application. You can use this procedure to create new portals with any name you choose:

1. Create a new Web application directory named `myAcme` by copying and pasting the `exampleportal` directory and renaming it `myAcme` in the `wlcsApp` directory.

2. Delete any `_tmp*` directories and the `classes` directory in the `myAcme/WEB-INF` directory. (Optional)
3. Edit the config.xml file in the `wlcsDomain` directory and add the following entry within the `<Application Deployed="true" Name="wlcsApp"....>` element:

```
<WebAppComponent Name="myAcme"
    ServletReloadCheckSecs="300" Targets="wlcsServer"
    URI="myAcme"/>
```

4. Start the WebLogic Commerce Server.
5. Use the Portal Administration tool to create a new portal `myAcme`; make desired portlets `visible`.
6. Use the Property Set Administration tool to create a new Application Init property set named `myAcme`. Change the `portalName` property value to `myAcme` and save the property set.
7. Enter the following URL in your browser: http://localhost:7501/myAcme

You have successfully deployed your portal as a Web application.

For a detailed discussion of these steps, see "Creating the Framework for Your Custom Portal" on page 5-6 in Chapter 5, "Building a Custom Portal Step-by-Step."

# Using Webflow Within a Portal

In WebLogic Commerce Server, Webflow is a feature that allows you to string together JavaServer Page (JSP) files, input processors (IPs) and pipelines (PLs) without hard coding the linkage between them. Instead, the linkage is defined in an external Webflow properties file.

Something to consider is how Webflow works with portals. When using Webflow, your users are conducted through a number of complete JSP pages as they work through a process. The portal, on the other hand, generally keeps the user on a single portal page, while the contents of that page (the portlets) change state. Due to this difference, in the current implementation, you cannot use the Webflow feature of page

transitions while you employ the Portal Framework. But you can utilize the power of input processors and pipeline processors from within a portlet. This section details how to do this.

## Using Input Processors and Pipeline Processors from Within a Portlet

The first requirement of Webflow is that your site must be deployed as a Web application. For more information, refer to the section "Deploying New Portals as Web Applications" on page 6-1. Once you have deployed your portal as a Web application, follow these steps:

1. Start your server and log on to the administration client.
2. Navigate to the Property Set Manager. Click on your property set to edit it.
3. You need to use a special destination determiner. Edit your `destinationdeterminer` property, and set it to be: `com.beasys.commerce.Webflow.WLCSPortalDestinationDeterminer.`
4. In a text editor, open or create your Webflow.properties located in your `%WL_COMMERCE_HOME%` folder.

   In this folder, you need to create transitions from your portlet to input processors and pipeline processors. It is important to understand that the destination determiner will route the flow from your portlet to the input processors and pipeline processors. Once Webflow has traversed through the IPs and PPs, it will forward the request to the URL that you specify in the Webflow method call in your portlet JSP (this will be discussed later). An example Webflow property file is as follows:

   ```
   destinmyportlet.jsp.link(mylink) = myportlet.inputprocessor
   myportlet.inputprocessor.success = myportlet.pipeline
   myportlet.pipeline.success = myportlet2.pipeline
   ```

   In this example, once Webflow has traversed through the second pipeline processor, it will allow the default destination determiner to forward to the URL specified in the request.

   In WebLogic Commerce Server, you may have only one Web application that is Webflow enabled. You must identify the property set to be used by Webflow. Do this by adding the following section to your `web.xml` file located in your `web-inf` folder:

   ```
   <context-param>
      <param-name>WLCS_APPLICATION_URL</param-name>
   ```

```
   <param-value>/application/commercewf</param-value>
</context-param>
```

Replace `commercewf` with the name of your property set.

Finally, you need to connect to Webflow from your portlet. In your portlet JSP, you must make a call to createWebflowURL. In this call, you must specify two parameters. Specify a parameter called `portalized` as `true`, and a parameter called `dest` with the URL that Webflow should go to after it has finished. For example:

**myportlet.jsp**

```
<%@ page import="com.beasys.commerce.webflow.*" %>
<%@ page
extends="com.beasys.commerce.portal.admin.PortalJspBase"%>

<form method=”POST”
        action="<%= WebflowJSPHelper.createWebflowURL(
           pageContext,
           "portlet.jsp",
           "link(mylink)",
           “&portalized=true&dest=/portal.jsp”,
           true) %>">
...
</form>
```

In this example, when the user clicks the Submit button the request will be forwarded to the destination of the myportlet.jsp.link(mylink) transition. Once Webflow has finished with all input processors and pipeline processors it has found there, it will forward to `portal.jsp`. Your `dest` parameter should refer to your portal page and not your portlet.

5. Test your portal.

If you successfully completed all the steps in this exercise, you should now have Webflow working within your portal.

# Cache Control in the Portal

When a portal is accessed by large numbers of users, its performance is affected dramatically by the caching scheme employed. There are many areas in the WebLogic Commerce Server product where caching is utilized. This section describes one scheme used by the portal framework.

## The Flow Manager Cache Tags

The portal framework uses the Flow Manager cache tags `<fm:>` to cache various portal objects that are required frequently (during each page hit on the portal) but are not modified as often. The Flow Manager cache tags are described in detail in the chapter "Personalization Server JSP Tag Library Reference" in the *Guide to Building Personalized Applications*.

## The Flow Manager Caching Scheme Helps to Manage Performance

The Flow Manager caching scheme is also sensitive to the underlying performance characteristics of the WebLogic server. It is tempting for JSP developers to use the `HttpSession` as a temporary cache to store objects. However, if you plan to scale your application deployment on a WebLogic cluster, overuse of the `HttpSession` is costly in terms of performance since its contents are replicated across the clustered nodes. The Flow Manager maintains its own cache and does not store objects into the `HttpSession`.

## The Parameters Governing the Flow Manager Cache

The caching scheme allows you to set a `global` attribute which makes the object you cache accessible to other users via the same `name`. You may also limit the accessibility of the object by setting the `scoped` attribute. This prepends the Flow Manager's servlet name (`application` for the default portal) to the object's name, providing a Web application scope. Finally, you can limit the scope to the 'current user only' by setting the `global` attribute to `false`.

The parameters governing the Flow Manager cache are located in the `weblogicCommerce.properties` file, as follows:

```
#cache of session values
_sessionCache.ttl=900000
_sessionCache.capacity=10000
_sessionCache.enabled=true
#cache of values across multiple users
_globalCache.ttl=600000
_globalCache.capacity=1000
_globalCache.enabled=true
```

The first set of parameters is used when the `global` attribute is set to `false`, the second set is used when `true`.

- The `ttl` parameter is the time-to-live (in milliseconds). `ttl` determines the maximum duration each object added to the cache will be maintained.
- If `enabled` is set to `false`, the tags will not cache objects and return null for `getter` calls.

## Portal Framework Objects Stored in the Cache

The following portal framework objects are stored in the cache:

**Table 6-1 Portal Framework Objects Stored in the Cache**

| Object | Scope |
|---|---|
| `com.beasys.commerce.portal.Portal` | scope = global |
| `com.beasys.commerce.portal.Portlet[][]` | scope = global if user not logged in, else scope is user |
| `com.beasys.commerce.portal.PortalColumnInformation[]` | scope = global |

CHAPTER

# 7 Portal Management Database Schema

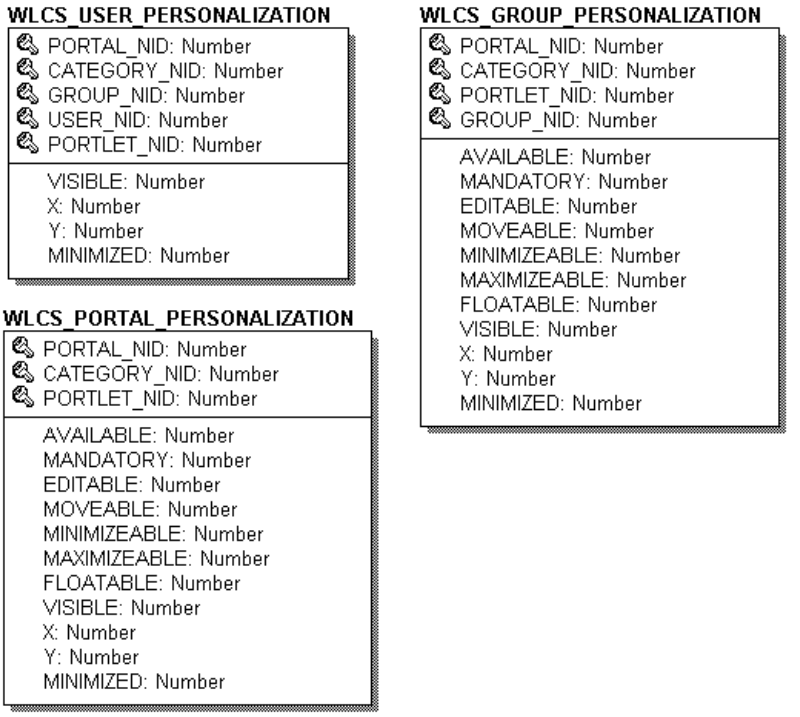
This topic documents the database schema for the WebLogic Personalization Server Portal Management package. This topic includes the following sections:

- The Entity-Relation Diagram
- List of Tables Comprising the Portal Management Package
- The Portal Management Data Dictionary
- The SQL Scripts Used to Create the Database

## The Entity-Relation Diagram

Figure 7-1 shows the logical Entity-Relation diagram for the WebLogic Commerce Server Portal and Portlet tables in the Commerce database. See the subsequent sections in this chapter for information about the data type syntax.

**Figure 7-1 Entity-Relation Diagram for the Portal and Portlet Tables**



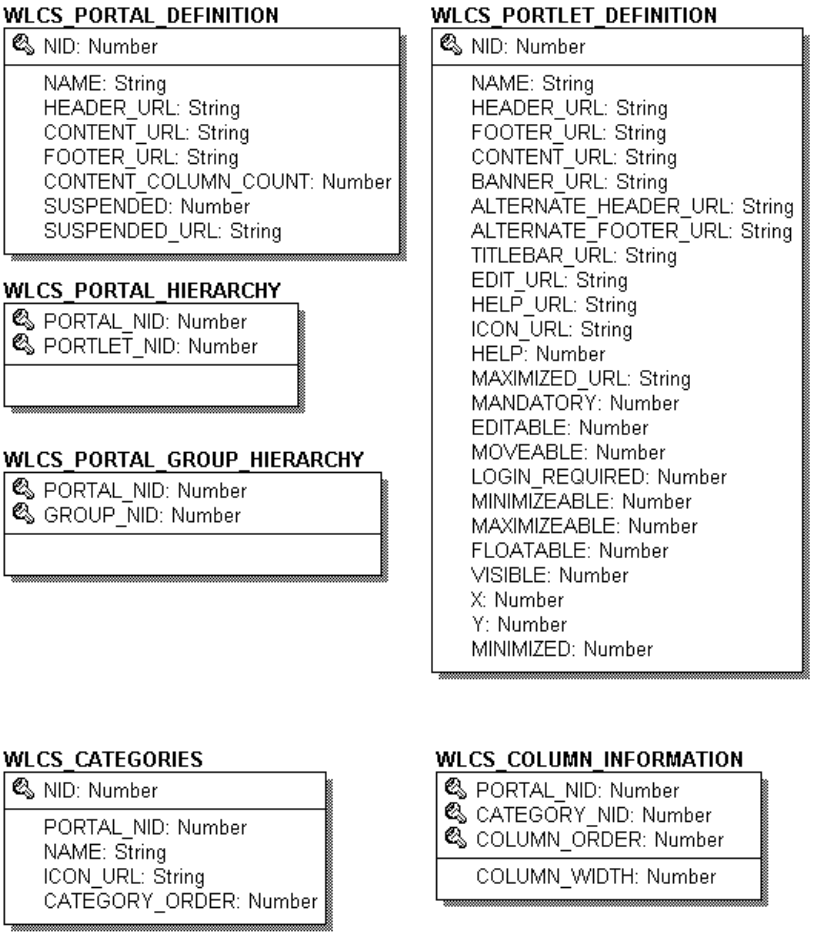
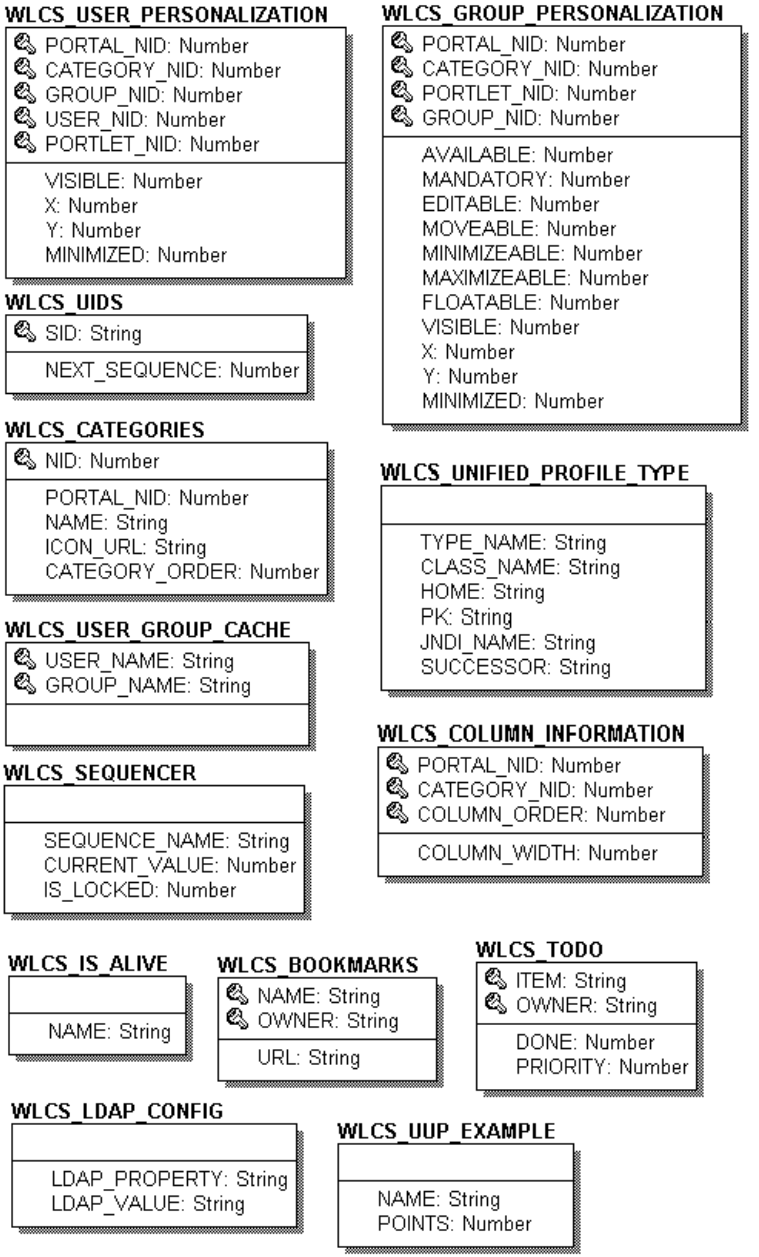



Figure 7-2 lists the tables used for the example portal.

**Note:** These tables are listed in the database schema chapter for the WebLogic Personalization Server.

**Figure 7-2 Tables Used for the Example Portal**



# List of Tables Comprising the Portal Management Package

**Portal Management tables:**

The WLCS_CATEGORIES Database Table
The WLCS_COLUMN_INFORMATION Database Table
The WLCS_PORTAL_DEFINITION Database Table
The WLCS_PORTLET_DEFINITION Database Table
The WLCS_PORTAL_HIERARCHY Database Table
The WLCS_PORTAL_GROUP_HIERARCHY Database Table
The WLCS_GROUP_PERSONALIZATION Database Table
The WLCS_PORTAL_PERSONALIZATION Database Table
The WLCS_USER_PERSONALIZATION Database Table

**The Tables Used for the Sample Portal:**

The WLCS_BOOKMARKS Database Table
The WLCS_CATEGORIES Database Table
The WLCS_COLUMN_INFORMATION Database Table
The WLCS_GROUP_PERSONALIZATION Database Table
The WLCS_IS_ALIVE Database Table
The WLCS_LDAP_CONFIG Database Table
The WLCS_SEQUENCER Database Table
The WLCS_TODO Database Table
The WLCS_UIDS Database Table
The WLCS_UNIFIED_PROFILE_TYPE Database Table
The WLCS_USER_GROUP_CACHE Database Table
The WLCS_USER_PERSONALIZATION Database Table
The WLCS_UUP_EXAMPLE Database Table

# The Portal Management Data Dictionary

In this section, the Portal and Portlet schema tables are arranged alphabetically as a data dictionary.

**Note:** Even though the following documentation references "foreign keys" to various tables, these constraints do not currently exist in this release of WebLogic Personalization Server. However, they will be (available in future releases) in place in future versions of WebLogic Personalization Server and we want you to be aware of these relationships now.

## The WLCS_BOOKMARKS Database Table

Table 7-1 describes the WLCS_BOOKMARKS table. This table is used by the Example portal and is not used except for demonstration purposes. It contains information used in the Bookmark portlet.

The Primary Key is comprised of `NAME` and `OWNER`.

**Table 7-1 WLCS_BOOKMARKS Table Metadata**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| `NAME` | `VARCHAR(150)` | The name of the bookmark. |
| `OWNER` | `VARCHAR(150)` | The owner of the bookmark. |
| `URL` | `VARCHAR(50)` | The URL of the bookmark. |

# The WLCS_CATEGORIES Database Table

Table 7-2 describes the WLCS_CATEGORIES table. This table is used to store category information for the portal portion of the WebLogic Personalization Server application.

**Note:** The CATEGORY feature has not been implemented at this time and, therefore, this table is not being used/populated.

The Primary Key is NID.

**Table 7-2 WLCS_CATEGORIES**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| NID | NUMBER(15) | Category identifier. |
| PORTAL_NID | NUMBER(15) | The Portal identifier. This column is a foreign key to the NID column of the WLCS_PORTAL_DEFINITION table. |
| NAME | VARCHAR(100) | The name for the category. |
| ICON_URL | VARCHAR(100) | The URL pointing to the icon associated with the category. This may be null. |
| CATEGORY_ORDER | NUMBER(5) | The sequence number identifying the order of display. |

# The WLCS_COLUMN_INFORMATION Database Table

Table 7-3 describes the WLCS_COLUMN_INFORMATION table. This table is used to store column definition information for each portal and category.

The Primary Key is comprised of PORTAL_NID, CATEGORY_NID and COLUMN_ORDER.

**Table 7-3 WLCS_COLUMN_INFORMATION**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| PORTAL_NID | NUMBER(15) | The Portal identifier. This column is a foreign key to the NID column of the WLCS_PORTAL_DEFINITION table. |
| CATEGORY_NID | NUMBER(15) | The Category identifier. |
| COLUMN_ORDER | NUMBER(5) | A sequence number identifying the display sequence for this column. Starting at the left-most part of the screen the COLUMN_ORDER would be 1. |
| COLUMN_WIDTH | NUMBER(5) | The value entered here is a percentage of the screen width. An example would be 30. This represents how wide this particular portal column is to be (30% of the screen). |

# The WLCS_IS_ALIVE Database Table

Table 7-4 describes the WLCS_IS_ALIVE table. This table is used by the JDBC connection pools to insure the connection to the database is still alive.

There is no Primary Key

**Table 7-4 WLCS_IS_ALIVE Table Metadata**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| NAME | VARCHAR(100) | Used by the JDBC connection pools to insure the connection to the database is still alive. |

# The WLCS_LDAP_CONFIG Database Table

Table 7-5 describes the WLCS_LDAP_CONFIG table. This table holds configuration information for LDAP functionality within the User Management module.

There is no Primary Key.

**Table 7-5 WLCS_LDAP_CONFIG Table Metadata**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| LDAP_PROPERTY | VARCHAR(100) | The property name. |
| LDAP_VALUE | VARCHAR(254) | The property value. |

# The WLCS_PORTAL_DEFINITION Database Table

Table 7-6 describes the WLCS_PORTAL_DEFINITION table.

The Primary Key is NID.

**Table 7-6 WLCS_PORTAL_DEFINITION**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| NID | NUMBER(15) | The identifier for the portal definition. |
| NAME | VARCHAR(500) | The name of the portal definition. Any combination of numbers and letters will be accepted in this field. |
| HEADER_URL | VARCHAR(500) | Enter a URL to display as the portal header. It can be a JSP or HTML fragment. |
| CONTENT_URL | VARCHAR(500) | Enter a URL relative to your portal working directory. |
| FOOTER_URL | VARCHAR(500) | Enter a URL to display as the portal footer. It can be a JSP or HTML fragment. |

**Table 7-6 WLCS_PORTAL_DEFINITION (Continued)**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| CONTENT_COLUMN_COUNT | NUMBER(5) | Specifies the number of content columns. Valid values at this time would be 1, 2, or 3. |
| SUSPENDED | NUMBER(5) | Set this flag to suspend the portal application and replace the portal home page with an "under maintenance" screen until service resumes. |
| SUSPENDED_URL | VARCHAR(500) | When the SUSPENDED flag is set, this URL will point to the JSP page to be displayed while the application is in suspend mode. |

# The WLCS_PORTAL_GROUP_HIERARCHY Database Table

Table 7-7 describes the WLCS_PORTAL_GROUP_HIERARCHY table. This table maintains records showing which groups are associated with each portal.

The Primary Key is comprised of PORTAL_NID and GROUP_NID.

**Table 7-7 WLCS_PORTAL_GROUP_HIERARCHY**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| PORTAL_NID | NUMBER(15) | The portal identifier. This column is a foreign key to the NID column of the WLCS_PORTAL_DEFINITION table. |
| GROUP_NID | NUMBER(15) | The group identifier. This column is a foreign key to the ENTITY_ID column of the WLCS_ENTITY_ID table. |

# The WLCS_GROUP_PERSONALIZATION Database Table

Table 7-8 describes the WLCS_GROUP_PERSONALIZATION table. Portals can be associated to groups and this table helps establish those relationships and maintain specific information for the group.

The Primary Key is comprised of `PORTAL_NID`, `CATEGORY_NID`, `PORTLET_NID` and `GROUP_NID`.

**Table 7-8 WLCS_GROUP_PERSONALIZATION**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| `PORTAL_NID` | `NUMBER(15)` | The portal identifier. This column is a foreign key to the `NID` column of the `WLCS_PORTAL_DEFINITION` table. |
| `CATEGORY_NID` | `NUMBER(15)` | The category identifier. This column is a foreign key to the `NID` column of the `WLCS_CATEGORIES` table. |
| `PORTLET_NID` | `NUMBER(15)` | The portlet identifier. This column is a foreign key to the `NID` column of the `WLCS_PORTLET_DEFINITION` table. |
| `GROUP_NID` | `NUMBER(15)` | The group identifier. This column is a foreign key to the `ENTITY_ID` column of the `WLCS_ENTITY_ID` table. |
| `AVAILABLE` | `NUMBER(5)` | A switch to identify whether or not this portlet is available. |
| `MANDATORY` | `NUMBER(5)` | This flag, when set, overrides the `VISIBLE` flag and requires the portlet be displayed. |
| `EDITABLE` | `NUMBER(5)` | This flag determines whether a user is allowed to edit any content. |
| `MOVEABLE` | `NUMBER(5)` | This column is not being used. |
| `MINIMIZEABLE` | `NUMBER(5)` | This flag determines whether or not the user will be allowed to minimize the portlet. |

**Table 7-8 WLCS_GROUP_PERSONALIZATION (Continued)**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| MAXIMIZEABLE | NUMBER(5) | This flag determines whether or not the user will be allowed to maximize the portlet. |
| FLOATABLE | NUMBER(5) | This flag determines whether the portlet can open up in its own browser window. |
| VISIBLE | NUMBER(5) | This flag determines whether or not the portlet is visible. |
| X | NUMBER(5) | The X coordinate determines the placement of the portlet on the screen. This is zero based and refers to the column placement (0=column 1, 1=column 2 and so on). |
| Y | NUMBER(5) | The Y coordinate determines placement of the portlet on the screen. Like the X coordinate, it is zero based. The Y coordinate refers to the row placement (0=row 1, 1=row 2 and so on). |
| MINIMIZED | NUMBER(5) | This flag determines whether or not the portlet should be displayed in a minimized format when initially displayed. |

# The WLCS_PORTAL_HIERARCHY Database Table

Table 7-9 describes the WLCS_PORTAL_HIERARCHY table. This table contains records showing which portlets are associated with each portal.

The Primary Key is comprised of PORTAL_NID and PORTLET_NID.

**Table 7-9 WLCS_PORTAL_HIERARCHY**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| PORTAL_NID | NUMBER(15) | The portal identifier. This column is a foreign key to the NID column of the WLCS_PORTAL_DEFINITION table. |

**Table 7-9 WLCS_PORTAL_HIERARCHY (Continued)**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| PORTLET_NID | NUMBER(15) | The portlet identifier. This column is a foreign key to the NID column of the WLCS_PORTLET_DEFINITION table. |

# The WLCS_PORTAL_PERSONALIZATION Database Table

Table 7-10 describes the WLCS_PORTAL_PERSONALIZATION table. This table maintains information pertinent to each personalized portal definition.

The Primary Key is comprised of PORTAL_NID, CATEGORY_NID and PORTLET_NID.

**Table 7-10 WLCS_PORTAL_PERSONALIZATION**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| PORTAL_NID | NUMBER(15) | The portal identifier. This column is a foreign key to the NID column of the WLCS_PORTAL_DEFINITION table. |
| CATEGORY_NID | NUMBER(15) | The category identifier. This column is a foreign key to the NID column of the WLCS_CATEGORIES table. |
| PORTLET_NID | NUMBER(15) | The portlet identifier. This column is a foreign key to the NID column of the WLCS_PORTLET_DEFINITION table. |
| AVAILABLE | NUMBER(5) | This flag, when set, overrides the VISIBLE flag and requires the portlet be displayed. 0 equates to false and 1 equates to true. |
| MANDATORY | NUMBER(5) | This flag, when set, overrides the VISIBLE flag and requires the portlet be displayed. |
| EDITABLE | NUMBER(5) | This flag determines whether a user is allowed to edit the content of the portal. 0 equates to false and 1 equates to true. |

**Table 7-10 WLCS_PORTAL_PERSONALIZATION (Continued)**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| MOVEABLE | NUMBER(5) | This column is not being used. 0 equates to `false` and 1 equates to `true`. |
| MINIMIZEABLE | NUMBER(5) | This flag determines whether or not the user will be allowed to minimize the portlet. 0 equates to `false` and 1 equates to `true`. |
| MAXIMIZEABLE | NUMBER(5) | This flag determines whether or not the user will be allowed to maximize the portlet. 0 equates to `false` and 1 equates to `true`. |
| FLOATABLE | NUMBER(5) | This flag determines whether the portlet can open up in its own browser window. 0 equates to `false` and 1 equates to `true`. |
| VISIBLE | NUMBER(5) | This flag determines whether or not the portlet is visible. 0 equates to `false` and 1 equates to `true`. |
| X | NUMBER(5) | The X coordinate determines the placement of the portlet on the screen. This is zero based and refers to the column placement (0=column 1, 1=column 2 and so on). |
| Y | NUMBER(5) | The Y coordinate determines placement of the portlet on the screen. Like the X coordinate, it is zero based. The Y coordinate refers to the row placement (0=row 1, 1=row 2 and so on). |
| MINIMIZED | NUMBER(5) | This flag determines whether or not the portlet should be displayed in a minimized format when displayed initially. 0 equates to `false` and 1 equates to `true`. |

# The WLCS_PORTLET_DEFINITION Database Table

Table 7-11 describes the WLCS_PORTLET_DEFINITION table. This table maintains information pertinent to each portlet definition.

The Primary Key is comprised of `NID`.

**Table 7-11 WLCS_PORTLET_DEFINITION**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| `NID` | `NUMBER(15)` | The portlet identifier. |
| `NAME` | `VARCHAR(500)` | The name of your portlet. Any combination of numbers and letters will be accepted in this field. |
| `HEADER_URL` | `VARCHAR(500)` | Enter a URL to display as the portlet header. It can be a JSP or HTML fragment. |
| `FOOTER_URL` | `VARCHAR(500)` | Enter a URL to display as the portlet footer. It can be a JSP or HTML fragment. |
| `CONTENT_URL` | `VARCHAR(500)` | Enter a URL relative to your portal working directory. |
| `BANNER_URL` | `VARCHAR(500)` | Enter a URL to display as the portlet banner under the portlet titlebar. It can be a JSP or HTML fragment. |
| `ALTERNATE_HEADER_URL` | `VARCHAR(500)` | Enter a URL to display as a Web page header when the portlet is floated or maximized. If this is null, the portal framework uses a default called `alternateheader.jsp`. |
| `ALTERNATE_FOOTER_URL` | `VARCHAR(500)` | Enter a URL to display as a Web page footer when the portlet is floated or maximized. If this is null, the portal framework uses a default called `alternatefooter.jsp`. |
| `TITLEBAR_URL` | `VARCHAR(500)` | Enter a URL to display as the portlet titlebar. It can be a JSP or HTML fragment. |

**Table 7-11 WLCS_PORTLET_DEFINITION (Continued)**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| EDIT_URL | VARCHAR(500) | If the EDITABLE flag has been set then a URL will be stored here that enables the user to edit the portlet content. |
| HELP_URL | VARCHAR(500) | If the HELP flag has been set then a URL must be specified that opens a help topic related to the portlet. |
| ICON_URL | VARCHAR(500) | A URL to display an icon (GIF) on the left side of the portlet titl bar. This image should be 27 pixels wide by 20 pixels high with 2 pixels of transparency on the right. |
| HELP | NUMBER(5) | This flag determines whether users can access a help screen in the portlet. If set, a Help icon displays in the portlet titlebar. |
| MAXIMIZED_URL | VARCHAR(500) | A URL for the content area of the maximized page. The default URL is your portlet content area URL. |
| MANDATORY | NUMBER(5) | This flag, when set, overrides the VISIBLE flag and requires the portlet be displayed. 0 equates to `false` and 1 equates to `true`. |
| EDITABLE | NUMBER(5) | This flag determines whether a user is allowed to edit the content of a portlet. 0 equates to `false` and 1 equates to `true`. |
| MOVEABLE | NUMBER(5) | This column is not being used. 0 equates to `false` and 1 equates to `true`. |
| LOGIN_REQUIRED | NUMBER(5) | This flag determines whether or not security is required for access to the portlet. 0 equates to `false` and 1 equates to `true`. |
| MINIMIZEABLE | NUMBER(5) | This flag determines whether or not the user will be allowed to minimize the portlet. 0 equates to `false` and 1 equates to `true`. |

**Table 7-11 WLCS_PORTLET_DEFINITION (Continued)**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| MAXIMIZEABLE | NUMBER(5) | This flag determines whether or not the user will be allowed to maximize the portlet. 0 equates to `false` and 1 equates to `true`. |
| FLOATABLE | NUMBER(5) | This flag determines whether the portlet can open up in its own browser window. 0 equates to `false` and 1 equates to `true`. |
| VISIBLE | NUMBER(5) | This flag determines whether or not the portlet is visible. 0 equates to `false` and 1 equates to `true`. |
| X | NUMBER(5) | The X coordinate determines the placement of the portlet on the screen. This is zero based and refers to the column placement (0=column 1, 1=column 2 and so on). |
| Y | NUMBER(5) | The Y coordinate determines placement of the portlet on the screen. Like the X coordinate, it is zero based. The Y coordinate refers to the row placement (0=row 1, 1=row 2 and so on). |
| MINIMIZED | NUMBER(5) | This flag determines whether or not the portlet should be displayed in a minimized format when displayed initially. 0 equates to `false` and 1 equates to `true`. |

# The WLCS_SEQUENCER Database Table

Table 7-12 describes the WLCS_SEQUENCER table. The WLCS_SEQUENCER table is used to maintain all of the sequence identifiers (for example, property_meta_data_id_sequence, and so on) used in the application.

There is no Primary Key.

**Table 7-12 WLCS_SEQUENCER Table Metadata**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| SEQUENCE_NAME | VARCHAR(50) | A unique name used to identify the sequence. |
| CURRENT_VALUE | NUMBER(15) | The current value of the sequence. |
| IS_LOCKED | NUMBER(1) | This flag identifies whether or not the particular SEQUENCE_ID has been locked for update. This column is being used as a generic locking mechanism that can be used for multiple database environments. |

# The WLCS_TODO Database Table

Table 7-13 describes the WLCS_TODO table. This table is used by the Example portal and is not used except for demonstration purposes. It contains information used in the To Do portlet.

The Primary Key is ITEM and OWNER.

**Table 7-13 WLCS_TODO Table Metadata**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| ITEM | VARCHAR(50) | The activity to be accomplished. |
| OWNER | VARCHAR(150) | The individual who owns, or is responsible, for this activity. |
| DONE | NUMBER(5) | The status identifying whether this item has been completed. |

**Table 7-13 WLCS_TODO Table Metadata (Continued)**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| PRIORITY | NUMBER(5) | The priority of the activity. |

# The WLCS_UIDS Database Table

Table 7-14 describes the WLCS_UIDS table. This table stores sequence information in a generic database independent format.

The Primary Key is SID.

**Table 7-14 WLCS_UIDS Table Metadata**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| SID | VARCHAR(100) | The name of the sequence. |
| NEXT_SEQUENCE | NUMBER(15) | The next value available for use with the sequence. |

# The WLCS_UNIFIED_PROFILE_TYPE Database Table

Table 7-15 describes the WLCS_UNIFIED_PROFILE_TYPE table. This table allows registration of classes which extend the ProvidedUser class.

There is no Primary Key.

**Table 7-15 WLCS_UNIFIED_PROFILE_TYPE Table Metadata**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| TYPE_NAME | VARCHAR(100) | Any unique name used for easy lookup. |
| CLASS_NAME | VARCHAR(100) | The name of the remote interface class. |
| HOME | VARCHAR(100) | The name of the home class. |
| PK | VARCHAR(100) | The name of the primary key class. |

**Table 7-15 WLCS_UNIFIED_PROFILE_TYPE Table Metadata (Continued)**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| JNDI_NAME | VARCHAR(100) | The name to look up in the JNDI tree. |
| SUCCESSOR | VARCHAR(100) | This column allows you to define another class should the TYPE_NAME not exist. This column is a foreign key to TYPE_NAME of the WLCS_UNIFIED_PROFILE_TYPE table. |

# The WLCS_USER_GROUP_CACHE Database Table

Table 7-16 describes the WLCS_USER_GROUP_CACHE table. In the event of a deep group hierarchy, this table will flatten the group hierarchy and enables quick group membership searches.

**Note:** The startup process GroupCache is disabled by default. This table will only be used if enabled.

The Primary Key is comprised of both USER_NAME and GROUP_NAME.

**Table 7-16 WLCS_USER_GROUP_CACHE Table Metadata**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| USER_NAME | VARCHAR(100) | FK—foreign key to WLCS_USER.IDENTIFIER. |
| GROUP_NAME | VARCHAR(100) | FK—foreign key to WLCS_GROUP.IDENTIFIER. |

# The WLCS_USER_PERSONALIZATION Database Table

Table 7-17 describes the WLCS_USER_PERSONALIZATION table.This table contains personalized portal information for the user.

The Primary Key is comprised of PORTAL_NID, CATEGORY_NID, GROUP_NID, USER_NID and PORTLET_NID.

**Table 7-17 WLCS_USER_PERSONALIZATION**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| PORTAL_NID | NUMBER(15) | The portal identifier. This column is a foreign key to the NID column of the WLCS_PORTAL_DEFINITION table. |
| CATEGORY_NID | NUMBER(15) | The category identifier. This column is a foreign key to the NID column of the WLCS_CATEGORIES table. |
| GROUP_NID | NUMBER(15) | The group identifier. This column is a foreign key to the ENTITY_ID column of the WLCS_ENTITY_ID table. |
| USER_NID | NUMBER(15) | The user identifier. This column is a foreign key to the ENTITY_ID column of the WLCS_ENTITY_ID table. |
| PORTLET_NID | NUMBER(15) | The portlet identifier. This column is a foreign key to the NID column of the WLCS_PORTLET table. |
| VISIBLE | NUMBER(5) | This flag determines whether or not the portlet is visible. 0 equates to false and 1 equates to true. |
| X | NUMBER(5) | The X coordinate determines the placement of the portlet on the screen. This is zero based and refers to the column placement (0=column 1, 1=column 2 and so on). |

**Table 7-17 WLCS_USER_PERSONALIZATION (Continued)**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| Y | NUMBER(5) | The Y coordinate determines placement of the portlet on the screen. Like the X coordinate, it is zero based. The Y coordinate refers to the row placement (0=row 1, 1=row 2 and so on). |
| MINIMIZED | NUMBER(5) | This flag determines whether or not the portlet should be displayed in a minimized format when displayed initially. 0 equates to `false` and 1 equates to `true`. |

# The WLCS_UUP_EXAMPLE Database Table

Table 7-18 describes the WLCS_UUP_EXAMPLE table. This is an example of how to use the Unified Profile Types.

There is no Primary Key.

**Table 7-18 WLCS_UUP_EXAMPLE Table Metadata**

| Column Name | Data Type | Description and Recommendations |
|---|---|---|
| NAME | VARCHAR(100) | A username. |
| POINTS | NUMBER(15) | A point accumulator based on various actions taken by the user. |

# The SQL Scripts Used to Create the Database

The database schemas for the WebLogic Personalization Server, WebLogic Commerce Server and BEA's Campaign Manager for WebLogic are all created by executing the `create_all` script for the target database environment.

## Cloudscape

For Cloudscape, execute one of the following:

- `WL_COMMERCE_HOME\db\cloudscape\3.5.1\create_all.bat` (Windows)
- `WL_COMMERCE_HOME/db/cloudscape/3.5.1/create_all.sh` (UNIX)

| Script Name | Description |
|---|---|
| `create_all.bat` | The execution of this script will create the WLPS, WLCS and Campaign Manager database schema. |
| `create_all.sh` | The execution of this script will create the WLPS, WLCS and Campaign Manager database schema. |
| `create_campaign.sql` | Creates the Campaign Manager specific database objects (e.g., tables, indexes and constraints). |
| `create_common.sql` | Creates the database objects which are common to WLPS and WLCS. |
| `create_mail_ad.sql` | Creates all the database objects used by the mail messaging component. |
| `create_wlcs.sql` | Creates all the database objects for WLCS (including Catalog and Order Management). |
| `create_wlps.sql` | Creates all the database object for WLPS. |
| `drop_campaign.sql` | Drops all database objects associated with Campaign Manager. |
| `drop_common.sql` | Drops the database objects which are common between WLPS and WLCS. |

| Script Name | Description |
|---|---|
| drop_mail_ad.sql | Drops the database objects used by the mail messaging component. |
| drop_wlcs.sql | Drops the database objects associated with WLCS. |
| drop_wlps.sql | Drops the database objects associated with WLPS. |
| insert_common.sql | Inserts core data into the common tables between WLPS and WLCS. |
| insert_wlcs.sql | Inserts core data into some of the WLCS tables. |
| insert_wlcs_sample_catalog.sql | Inserts sample data into the product catalog. |
| insert_wlcs_sample_customer.sql | Inserts sample customer information into WLCS tables. |
| insert_wlcs_sample_data.sql | Inserts sample data into various WLCS tables. |
| insert_wlps.sql | Inserts core data into WLPS tables. |
| insert_wlps_sample_data.sql | Inserts sample data into various WLPS tables. |

# Oracle

For Oracle, from the command line, move to the following directory:

```
WL_COMMERCE_HOME/db/oracle/8.1.6
```

After logging into SQL*PLus, simply execute the `create_all.sql` script (e.g., `@create_all`).

| Script Name | Description |
|---|---|
| create_campaign.sql | Creates the Campaign Manager specific database objects (e.g., tables, indexes and constraints). |
| create_common.sql | Creates the database objects which are common to WLPS and WLCS. |

| Script Name | Description |
|---|---|
| `create_mail_ad.sql` | Creates all the database objects used by the mail messaging component. |
| `create_wlcs.sql` | Creates all the database objects for WLCS (including Catalog and Order Management). |
| `create_wlps.sql` | Creates all the database object for WLPS. |
| `drop_campaign.sql` | Drops all database objects associated with Campaign Manager. |
| `drop_common.sql` | Drops the database objects which are common between WLPS and WLCS. |
| `drop_mail_ad.sql` | Drops the database objects used by the mail messaging component. |
| `drop_wlcs.sql` | Drops the database objects associated with WLCS. |
| `drop_wlps.sql` | Drops the database objects associated with WLPS. |
| `insert_common.sql` | Inserts core data into the common tables between WLPS and WLCS. |
| `insert_wlcs.sql` | Inserts core data into some of the WLCS tables. |
| `insert_wlcs_sample_catalog.sql` | Inserts sample data into the product catalog. |
| `insert_wlcs_sample_customer.sql` | Inserts sample customer information into WLCS tables. |
| `insert_wlcs_sample_data.sql` | Inserts sample data into various WLCS tables. |
| `insert_wlps.sql` | Inserts core data into WLPS tables. |
| `insert_wlps_sample_data.sql` | Inserts sample data into various WLPS tables. |
| `install_report.sql` | This script is used to summarize the database installation. Information such as the number of tables, indexes, etc., is displayed. |
| `statistics.sql` | This script is used in computing statistics on various database objects (e.g., tables and indexes) in an Oracle environment. |

# Defined Constraints

There are no constraints associated with this part of the schema.

# 8 Portal Management JSP Tag Library Reference

The Portal Management component includes JSP tags for access to the fundamental data comprising a portal, such as portal and portlet properties.

To import the Portal Management JSP tags, use the following code:

```
<%@ taglib uri="esp.tld" prefix="esp" %>
```

**Note:** In the following tables, the Required column specifies if the attribute is required (yes) or optional (no). In the R/C column, C means that the attribute is a Compile time expression, and R means that the attribute can be either a Request time expression or a Compile time expression.

# <esp:eval>

The <esp:eval> tag (Table 8-1) is used to evaluate a conditional attribute of a portlet, for example, isMinimizeable. The tag expects a com.beasys.portal.Portlet to be accessible in the session with the key PortalTagConstants.PORTLET. If the conditional attribute evaluates to true, the body of the <esp:eval> tag is processed. Otherwise, it is not.

**Table 8-1 <esp:eval>**

| Tag Attribute | Required | Type | Description | R/C |
|---|---|---|---|---|
| tag | Yes | String | The name of the portlet attribute to evaluate.<br>The following attributes can be retrieved:<br>■ isEditable<br>■ isVisible<br>■ hasHelp<br>■ isMandatory<br>■ isMoveable<br>■ isMinimizeable<br>■ isMaximizeable<br>■ isFloatable<br>■ isMinimized | R |
| target | No | Portlet | The com.beasys.portal.Portlet to be evaluated. | R |

## Example

```
<%@ taglib uri="esp.tld" prefix="esp" %>
.
.
.
<esp:eval tag="isMinimizable">
     <% titleBar.include(minimizeButton); %>
</esp:eval>
```

# <esp:get>

The <esp:get> tag (Table 8-2) retrieves a String attribute of a portlet. This tag expects a com.beasys.portal.Portlet to be accessible in the session with the key PortalTagConstants.PORTLET.

**Table 8-2 <esp:get>**

| Tag Attribute | Required | Type | Description | R/C |
|---|---|---|---|---|
| tag | Yes | String | The name of the portlet attribute to retrieve.<br>The following attributes can be retrieved:<br>■ editURL<br>■ maximizedURL<br>■ headerURL<br>■ footerURL<br>■ contentURL<br>■ title | R |
| target | No | Portlet | The com.beasys.portal. Portlet to be evaluated. | R |

## Example

■
```
<%@ taglib uri="esp.tld" prefix="esp" %>
.
.
.
<tr>
   <td>
      <esp:get tag="title"/>
   </td>
</tr>
```

# <esp:getGroupsForPortal>

The `<esp:getGroupsForPortal>` tag (Table 8-3) retrieves the names of the groups associated with a Portal. The results are put into the variable declared in the `id` attribute of the tag, which is a `String` array.

**Table 8-3 <esp:getGroupsForPortal>**

| Tag Attribute | Required | Type | Description | R/C |
|---|---|---|---|---|
| id | Yes | String | A resulting string array containing the names of the groups associated with the given Portal. | R |
| portalName | Yes | String | The name of the Portal to be checked for associated groups. | R |

## Example

```
<%@ taglib uri="esp.tld" prefix="esp" %>
.
.
.
<esp:getGroupsForPortal id ="groups" portalName="<%=portalName%>">
for (i=0;i<groups.length;i++)
{

String groupName = groups[i];
}
</esp:getGroupsForPortal>
```

# <esp:monitorSession>

The `<esp:monitorSession>` tag (Table 8-4) can be added to the beginning of any JSP page to disallow access to the page if the session is not valid or if the user is not logged in.

**Table 8-4 <esp:monitorSession>**

| Tag Attribute | Required | Type | Description | R/C |
|---|---|---|---|---|
| goToPage | No | String | The error page that you want displayed if the page is not accessible.<br>The default value is `portalerror.jsp`. | R |
| loginRequired | No | String | Indicates whether the user is required to be logged in to access the JSP page including the tag.<br>The default value is `false`. | R |

## Example

```
<%@ taglib uri="esp.tld" prefix="esp" %>
.
.
.
<esp:monitorSession loginRequired="true" />
```

# <esp:portalManager>

The `<esp:portalManager>` tag (Table 8-5) is used to perform `create`, `get`, `getColumnInfo`, `update`, and `remove` actions on `com.beasys.portal.Portal` objects. This tag is an empty tag.

**Table 8-5 <esp:portalManager>**

| Tag Attribute | Required | Type | Description | R/C |
|---|---|---|---|---|
| id | When action equals `get` or `getColumnInfo` | String | The name to which resultant information is assigned for subsequent use in the JSP page. | R |

**Table 8-5 <esp:portalManager> (Continued)**

| Tag Attribute | Required | Type | Description | R/C |
|---|---|---|---|---|
| action | No | String | The action to perform. Allowed values include:<br>`create`: Creates a new portal.<br>`get`: (default value) Retrieves an object of type com.beasys.portal.Portal.<br>`getColumnInfo`: Retrieves a com.beasys.portal.PortalColumnInformation[]<br>`update`: Updates the provided target com.beasys.portal.Portal.<br>`remove`: Removes the provided target com.beasys.portal.Portal. | R |
| portalName | No | String | The name of the portal to retrieve, or whose column information is to be retrieved. The default value is `session.getValue(com.beasys.commerce.portal.admin.PortalAdminHelper.qualifiedName(PortalTagConstants.PORTAL_NAME, request))` | R |
| target | When action equals `create`, `update`, or `remove` | Portal | The com.beasys.portal.Portal to be created, updated, or removed. | R |

## Example

```
<%@ taglib uri="esp.tld" prefix="esp" %>
.
.
.
<esp:portalManager id="portal" action="get"
portalName="BEAPortal"/>
```

# <esp:portletManager>

The <esp:portletManager> tag (Table 8-6) is used to perform create, get, getArranged, update, and remove actions on com.beasys.portal.Portlet objects. This tag is an empty tag.

**Table 8-6 <esp:portletManager>**

| Tag Attribute | Required | Type | Description | R/C |
|---|---|---|---|---|
| id | When action equals get or getArranged | String | The name to which resultant information is assigned for subsequent use in the JSP page. | R |
| action | No | String | The action to perform. Allowed values include:<br>create: Creates a new portlet.<br>get: Retrieves an object of type com.beasys.portal.Portlet.<br>getArranged: Retrieves a com.beasys.portal.Portlet[][] that prescribes the row-column layout of portlets for the provided portal-user-group combination.<br>update (default value): Updates the provided target com.beasys.portal.Portlet.<br>remove: Removes the provided target com.beasys.portal.Portlet. | R |
| portalName | No | String | The name of the portal corresponding to the target portlet or to the portlet(s) to be retrieved. The default value is session.getValue(com.beasys.commerce.portal.admin.PortalAdminHelper.qualifiedName(PortalTagConstants.PORTAL_NAME, request)). | R |
| portletName | No | String | The name of the portlet corresponding to the target portlet or to the portlet(s) to be retrieved.<br>There is no default value. | R |

**Table 8-6 <esp:portletManager> (Continued)**

| Tag Attribute | Required | Type | Description | R/C |
|---|---|---|---|---|
| groupId | No | Long | The name of the group corresponding to the target portlet or to the portlet(s) to be retrieved. The default value is `com.beasys.commerce.axiom.jsp.JspHelper.getSessionValue(com.beasys.commerce.user.tags.UserManagerTagConstants.PROFILE_SUCCESSOR_UID, request)` | R |
| userId | No | Long | The name of the user corresponding to the target portlet or to the portlet(s) to be retrieved. The default value is `com.beasys.commerce.axiom.jsp.JspHelper.getSessionValue(com.beasys.commerce.user.tags.UserManagerTagConstants.PROFILE_USER_UID, request)` | R |
| target | When action equals `create`, `update`, or `remove` | Portlet | The com.beasys.portal.Portlet to be created, updated, or removed. | R |
| scope | No | String | The scope to be applied to the provided action. Allowed values include:<br>`global` (default value): Specifies that portlet creation, removal, retrieval, or update should apply across all portals, groups, and users.<br>`portal`: Specifies that portlet creation, removal, retrieval, or update applies to the provided portal.<br>`group`: Specifies that portlet creation, removal, retrieval, or update applies to the provided portal-group combination.<br>`user`: Specifies that portlet creation, removal, retrieval, or update applies to the provided portal-group-user combination. | R |

## Example

```
<%@ taglib uri="esp.tld" prefix="esp" %>
.
.
.
<esp:portletManager id="arrangedPortlets" action="getArranged"
userId="myUser" portalName="myPortal"/>
```

# <esp:props>

The `<esp:props>` tag (Table 8-7) is used to get a property from the Portal Properties bean. The Portal Properties bean's deployment descriptor contains default values used by the Portal Administration Tool.

**Table 8-7 <esp:props>**

| Tag Attribute | Required | Type | Description | R/C |
|---|---|---|---|---|
| id | Yes | String | A java.lang.String variable name for the property value. | R |
| propertyName | Yes | String | The name of the property to get in the Portal Property Bean. | R |

## Example

```
<%@ taglib uri="esp.tld" prefix="esp" %>
.
.
.
<esp:props id="headerURL"
propertyName="commerce.default.portal.headerURL" />
```

# Index

## E



## F



## G



## H



## I



## J

## L



## M



## P

## R



## S



## T



## U



## W