webMethods Portal

Design Guide

VERSION 6.5.2

webMethods, Inc.
South Tower
3877 Fairfax Ridge Road
Fairfax, VA 22030
USA
703.460.2500
http://www.webmethods.com

Document ID: PRTL-DZG-652-20060330

Contents

About This Guide

This guide describes how to use the Portlet Developer to build portlets and develop portal pages for the webMethods Portal, as well as create and modify skins and shells that determine the look, feel, and structure of portal pages.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Italic	Identifies variable information that you must supply or change based on your specific situation or environment. Identifies terms the first time they are defined in text. Also identifies service input and output variables.
Narrow font	Identifies storage locations for services on the webMethods Integration Server using the convention *folder.subfolder:service*.
`Typewriter font`	Identifies characters and values that you must type exactly or messages that the system displays on the console.
UPPERCASE	Identifies keyboard keys. Keys that you must press simultaneously are joined with the "+" symbol.
\	Directory paths use the "\" directory delimiter unless the subject is UNIX-specific.
[ ]	Optional keywords or values are enclosed in [ ]. Do not type the [ ] symbols in your own code.

Additional Information

The webMethods Advantage Web site at http://advantage.webmethods.com provides you with important sources of information about webMethods components:

- **Troubleshooting Information.** webMethods provides troubleshooting information for many webMethods components in the webMethods Knowledge Base.
- **Documentation Feedback.** To provide documentation feedback to webMethods, go to the Documentation Feedback Form on the webMethods Bookshelf.
- **Additional Documentation.** All webMethods documentation is available on the webMethods Bookshelf.

webMethods.

CHAPTER 1

Setting Up the Portlet Developer

System Requirements for the Portlet Developer

The Portlet Developer is a plug-in for the Eclipse Platform, an integrated development environment (IDE) developed by the Eclipse Foundation. To develop portlets, the Portlet Developer needs access to libraries installed with Portal. To deploy portlets to a Portal server, the Portlet Developer needs HTTP access to the computer where the server resides. The system requirements for the Portlet Developer are as follows:

- Operating System: Microsoft Windows 2000 SP4 or Windows XP Professional
- Eclipse version 3.0.1
- Sun Java Development Kit 1.4.2
- Portal installed on a file system or mapped drive available on your computer

Installing the Eclipse Platform

The Portlet Developer is a plug-in for the Eclipse Platform, an integrated development environment (IDE) developed by the Eclipse Foundation. To use the Portlet Developer, you need to download and install Eclipse version 3.0.1, which is available at http://www.eclipse.org.

Installing Eclipse

To install the Eclipse Platform, use the following procedure.

To install the Eclipse Platform

1. Download the version of the Eclipse 3.0.1 SDK appropriate for the operating system on your development computer.
2. Extract the files from the compressed file you have downloaded.

   Files are extracted into a directory with the name eclipse.
3. In the eclipse directory, run the eclipse executable.

   The first time you run this executable, an installation process occurs, after which the Eclipse Platform opens.

Installing the Graphical Editor Framework

To use the Form Builder feature of the Portlet Developer, you need to install the Graphical Editor Framework (GEF), available from the Eclipse Foundation.

To install the Graphical Editor Framework

1. Start the Eclipse IDE.
2. On the **Help** menu of Eclipse, click **Software Updates** and **Find and Install**.

   The **Install/Update** perspective is displayed.
3. In the **Install/Update** wizard, choose the **Search for new features to install** option and click **Next**.
4. In the tree view, expand Eclipse.org update site, select the **GEF SDK** version that matches the version of Eclipse you have installed, and click **Next**.
5. In **Search Results**, select **Graphical Editing Framework** and click **Next**.

Note: You do not need the entire Graphical Editor Framework SDK.

6. Continue through the wizard to complete the installation.

Installing and Configuring the Portlet Developer

Before you can configure the Portlet Developer, these conditions must exist:

- You must download and install the Eclipse IDE, as described in "Installing the Eclipse Platform" on page 16.
- Portal must be running on a computer on the same network as the computer on which you have installed Eclipse.
- You need to have access to the Portal server as a member of the Portlet Developer group or have access to the Developer Commands Security Realm.

With these two conditions met, you can install and configure the Portlet Developer by following these steps:

1. Install the Portlet Developer, as described in "Installing the Portlet Developer on Eclipse" on page 18.
2. Configure the Portlet Developer, as described in "Configuring the Portlet Developer" on page 19.

Installing the Portlet Developer on Eclipse

To install the Portlet Developer on Eclipse, use the following procedure. If you are upgrading from an earlier version of the Portlet Developer, see "Upgrading the Portlet Developer from an Earlier Version" on page 20.

To install the Portlet Developer on Eclipse

1 Start the Eclipse IDE.

2 On the **Help** menu of Eclipse, click **Software Updates** and **Find and Install**.

3 In the **Install/Update** wizard, choose the **Search for new features to install** option and click **Next**.

4 Click **New Remote Site**.

5 In the **Name** field of the New Update Site dialog box, type `Portal`.

6 In the **URL** field, type this:

`http://`*`server_name`*`:`*`port_number`*`/portletDeveloper3/`

This parameter...	Is this...
server_name	The name or IP address of the computer where Portal is running. **Note:** If Portal is running on the local host, you must still use a valid host name or IP address. You cannot use the name `localhost` or 127.0.0.1.
port_number	The port number for the Web server used by Portal. Ask the portal administrator what port number you should use. If you leave out the port number, a default number of 80 is assumed.

Note: The final slash (/) is required. If you omit it, Eclipse cannot find the Portal plug-ins to install.

You can later connect the Portlet Developer to a different Portal server by modifying Portal preferences, as described in "Modifying Portlet Developer Preferences" on page 21.

7 In the **Sites to include in search** view, expand **Portal**, select **webMethods Portal Plugins**, and click **Next**.

8 In the **Select the features to install** view, select one or more of the following:

Select this...	To install this plug-in on Eclipse...
webMethods Portlet Form Builder	The Form Builder of the Portlet Developer. This plug-in requires the Graphical Editor Framework. See "Installing the Graphical Editor Framework" on page 17.
wmPortal Development for Eclipse	The Portlet Developer plug-in for Eclipse.

9 Click **Next**.

10 Accept the terms in the license agreements and click **Next**.

11 Click **Finish**.

12 In the Jar Verification dialog box, click **Install**.

The Jar Verification dialog box appears separately for each plug-in.

13 In the Install/Update dialog box, click **Yes** to restart Eclipse.

Configuring the Portlet Developer

After you have installed the Portlet Developer, as described in "Installing the Portlet Developer on Eclipse" on page 18, you are ready to configure it.

To configure the Portlet Developer

1 After Eclipse has restarted, set preferences for the Portlet Developer:

a In the **Window** menu of Eclipse, choose **Preferences**.

b In the navigation tree, select **Portal**.

c In **Portal Home Path**, click **Browse**, navigate to the top-level directory for Portal on a file system or mapped drive available on your computer and click **OK**.

The Portlet Developer uses this path to locate libraries it uses. If you have performed a default installation, the top-level directory for Portal is *webMethods_Install_Dir*/Portal.

d In the **Target Server** list, choose the Portal server instance against which you are building portlets.

The **Target Server** list is based on the value you entered in the **Portal Home Path** field in step c.

e In the **Portal Server URL** field, type the URL of the instance of Portal to which you will deploy completed portlets.

  For example, if Portal is running on the computer test_one and the Web server listens on port number 8080, the **Portal Server URL** field entry is:

  ```
  http://test_one:8080
  ```

f In the **Default Portlet Prefix** field, type the prefix to be appended to all portlets created by the Portlet Developer.

  By default, the default prefix is **wm**.

g Click **OK**.

2 On the Window menu choose **Open Perspective** and **Other**.

3 In the Select Perspective dialog box, choose **webMethods Portlets** and then click **OK**.

  The Eclipse IDE now displays the Portal perspective and is ready for portlet development.

Upgrading the Portlet Developer from an Earlier Version

If you have previously used an earlier version of the Portlet Developer, you can upgrade to the current version using the following procedure. This procedure assumes that both the Portlet Developer and the Portal server are installed on the same machines as for the earlier version. If either the Portlet Developer or the Portal server is on a different machine than before, use the procedure described in "Installing the Portlet Developer on Eclipse" on page 18.

To upgrade the Portlet Developer from an earlier version

1 Start the Eclipse IDE.

2 On the **Help** menu of Eclipse, click **Software Updates** and **Find and Install**.

3 In the **Install/Update** wizard, choose the **Search for new features to install** option and click **Next**.

4 In the **Sites to include in search** view, expand **Portal**, select **webMethods Portal Plugins**, and click **Next**.

5 In the **Select the features to install** view, select one or more of the following:

Select this...	To install this plug-in on Eclipse...
webMethods Portlet Form Builder	The Form Builder of the Portlet Developer. This plug-in requires the Graphical Editor Framework. If you have not previously used the Form Builder, see "Installing the Graphical Editor Framework" on page 17.
wmPortal Development for Eclipse	The Portlet Developer plug-in for Eclipse.

6 Click **Next**.

7 Accept the terms in the license agreements and click **Next**.

8 Click **Finish**.

9 In the Jar Verification dialog box, click **Install**.

The Jar Verification dialog box appears separately for each plug-in.

10 In the Install/Update dialog box, click **Yes** to restart Eclipse.

Modifying Portlet Developer Preferences

There are several Portlet Developer preferences you can modify in the Eclipse IDE.

To modify Portlet Developer Preferences

1 Start the Eclipse IDE.

2 On the **Window** menu of Eclipse, click **Preferences**.

3 In the **Preferences** window, click **Portal**.

4 In the Portal panel, modify preferences as needed:

For this preference...	Do this...
Portal Home Path	Click **Browse**, navigate to the top-level directory for Portal on a file system or mapped drive available on your computer and click **OK**. The instance of Portal at this location does not need to be running. The Portlet Developer uses this path to locate libraries it uses. If you have performed a default installation, the top-level directory for Portal is *webMethods_Install_Dir*/Portal. This entry does not have to be the same instance of Portal as specified in the **Portal Server URL** field.

For this preference...	Do this...
Target Server	From the list, choose the server instance within Portal from which to use libraries. An installation of Portal can have multiple instances of the Portal server, which can have additional libraries needed by the Portlet Developer. The **Target Server** list contains all server instances that have been added to the installation of Portal specified in the **Portal Home Path** field. The default target server is **default**.
Portal Server URL	Type the URL of the instance of Portal to which you will deploy completed portlets, including the port number for the Web server used by Portal. For example, if Portal is running on the computer test_one and the Web server listens on port number 8080, the **Portal Server URL** field entry is: `http://test_one:8080` This entry does not have to be the same instance of Portal as specified in the **Portal Home Path** field.
Default Portlet Prefix	Type the prefix to be appended to all portlets created by the Portlet Developer. By default, the prefix is **wm**.

5 Click **OK**.

Eclipse Documentation and Related Resources

It is beyond the scope of this guide to provide documentation for the Eclipse IDE. Information about Eclipse is available as online help and in PDF form from the Eclipse Foundation.

If you are working in Eclipse, online help is available from **Help ▸ Help Contents**.

You can download a user's guide the Eclipse IDE, from http://www.eclipse.org. On the Eclipse home page, locate the link to online documentation.

CHAPTER 2

Portlet Basics

What is a Portlet?

A portlet is a key component of the portal framework. A *portlet* is a server-side, mini-application that resides on the Portal server or a piece of functionality that runs on the back end of the Portal server. A portlet may or may not have a user interface (UI). Architecturally, a portlet is made up of JSP files, XML files, Java classes, and other UI components. All of the UI components within webMethods Portal exist as portlets.

Deploying portlets in your Portal server provides a way to bring together disparate data sources, such as XML, Web services, SQL data sources, back-end systems, enterprise applications, and information stored in the Portal server.

Portlets provide functionality and can have either a user or system interface. A portlet can also act as a client for a Web service. As a Web service client, the portlet may not necessarily have a UI. This type of portlet might perform a back-end service such as verifying a credit card.

A published portlet can be shared from one part of the portal to another (including from one portal page to another), can participate in exchanges through the Portal server's Java-based messaging system, and can be wired to other portlets by exchanging property values and other data.

Because portlets can contain logic as well as content, the range of new functionality you can add through them is extremely broad. webMethods Portal comes with many portlets already installed. Some examples of functionality added through portlets include:

- A hierarchical view of portal contents (Folder View).
- A view of content retrieved from an Internet site (Yahoo! Weather).
- A database connection through which SQL statements can be sent and data returned (SQL Data).
- A mechanism for embedding select portions of existing Web sites and Web applications inside the portal, including user authentication (Webclip).
- A mechanism for calling into other webMethods components and invoking a service.

What is the Portlet Developer?

webMethods Portal provides a framework called the *portlet controller* that helps to create portlets with multiple pages and actions. The framework consists of run-time, base portlet classes, declaration in the portlet descriptor, JSP tag library, and the Portlet Developer plug-in to the Eclipse IDE.

The portlet controller introduces two main concepts: portlet layout and portlet method. In a JSP-based portlet, the layout is a JSP and the metadata associated with it. A portlet method is an actual Java method defined in the portlet bean class plus the metadata about it. You can think of the multi-page portlet as a small Web application with a navigation model, state, and supported actions. This small Web application can be built using a collection of portlet layouts, methods, and a navigation logic.

There are some generic navigation models that are supported by the Portlet Developer:

- Wizard—the portlet has one or more layouts that the user can step through sequentially. At the end of the sequence, some final action is typically performed, based on the data collected on all pages. For an example of a wizard portlet, log on to webMethods Portal as the portal administrator and navigate as follows:

  a If webMethods Portal is not already displaying the Administrative Folders, in the global navigation toolbar, click **Administration**.

  b In the **Administration Dashboard** folder, click **Portal Configuration**.

  c On the **Portal Configuration** page, click **Install Administration**.

  The Install Administration portlet is a wizard that takes you through the steps needed to install or uninstall a component on the Portal server.

- Tabs—the portlet has one or more independent layouts. At the top of each layout there is a tab control that displays one tab for each portlet layout. Clicking the tab makes the corresponding layout active. For an example of a tabs portlet, log on to webMethods Portal as the portal administrator and navigate as follows:

  a If webMethods Portal is not already displaying the Administrative Folders, in the global navigation toolbar, click **Administration**.

  b In the **Administration Dashboard** folder, click **Portal Analysis**.

  c On the **Portal Analysis** page, click **View Log Messages**.

  The **View Log Messages** portlet has three tabs: **View Logging Messages**, **Edit Logging Configuration**, and **Cleanup Logging Sessions**.

A portlet is not limited to a single navigation model.

Getting Started with the Portlet Developer

The Portlet Developer is a plug-in to the Eclipse IDE. Before you can begin working with the Portlet Developer you must have completed these three tasks:

- Have an instance of webMethods Portal running on a computer with network access to your development computer.
- Install the Eclipse Platform on your development computer, as described in "Installing the Eclipse Platform" on page 16.
- Install and configure the Portlet Developer, which is a plug-in to the Eclipse IDE, as described in "Installing and Configuring the Portlet Developer" on page 17.

Note: To use the Portlet Developer, you need to have access to the Portal server as a member of the Portlet Developer group or have access to the Developer Commands Security Realm.

These aspects of the Portlet Developer are described as follows:

This aspect...	Is described here...
Making the Portlet Developer visible in Eclipse	"Displaying the Portlet Developer Perspective" on page 26
The components of a portlet	"Elements in the wmPortal Component View" on page 27
The source-file structure of a portlet	"Elements in the Navigator View" on page 27
The portlet types you can create	"Portlet Categories" on page 28

Displaying the Portlet Developer Perspective

With the Portlet Developer installed, you can display it in the Eclipse IDE.

To display the Portlet Developer in Eclipse

1 Start the Eclipse IDE.

2 To display the **webMethods Portlets** perspective, on the **Window** menu, click **Open Perspective** and **webMethods Portlets**.

Tip! If **webMethods Portlets** is not immediately visible as a choice for **Open Perspective**, click **Others** and look for it in the list displayed there. If **webMethods Portlets** is not displayed under **Others**, it has not been installed.

3 If needed, on the **Window** menu, click **Show View**, **Other**, **wmPortal Components** and then click **OK**.

4 At the bottom of the upper-left view of Eclipse, click the **wmPortal Components** tab to make it the active view.

Tip! If the **wmPortal Components** tab is missing, you can reset the perspective back to its original state by choosing **Window** ▸ **Reset Perspective**.

The **wmPortal Components** tab provides a tree view of your portlets and the elements of which they are composed. The **Navigator** tab provides a tree view of the file structure of your portlets.

Elements in the wmPortal Component View

The **WmPortal Component** view provides a tree view of the elements you can alter as you create and modify a portlet. For each portlet you create, a set of components is available for use in adding functionality to it. These components appear in the view:

This component...	Contains...
Portlet Bean	The major Java class for the portlet. If needed, you can open the text editor to add new Java code or alter existing code.
Property Groups	Properties you create to hold parameters for the portlet, such as input values for a Web service.
PCA Layouts	JSP pages used in the portlet.
PCA Methods	The methods that perform the actions to be taken by the portlet.
Web References	References to Web services Description Language (WSDL) files that the portlet uses as Web services.
Data Queries	Data queries the portlet is to make. Such queries are usually SQL database queries, but they can also be generic data queries to some arbitrary system.
Portal Forms	Forms you can create to be used in the portlet. The Portal Forms element is available only if you have installed the Graphical Editor Framework for Eclipse and the Form Builder installed.
Resource Strings	Resource strings that are placed into a resource bundle so they can be used to localize the portlet.

Elements in the Navigator View

The **Navigator** view provides a tree view of the file structure of a portlet. If you double-click a file in this view, the contents of the file appear in the editor. These structure elements appear in the view:

This element...	Contains...
classes directory	Classes associated with the portlet.
db directory	SQL code associated with the portlet.
src directory	Java source files associated with the portlet.
ui directory	JSP pages and client-side resources associated with the portlet.
WEB-INF directory	Configuration data about the portlet, including the portlet descriptor files portlet.xml (JSR 168 compliant) and wm_portlet.xml (contains webMethods extensions). The classes subdirectory contains compiled versions of the JSP pages.

This element...	Contains...
.classpath file	Classpath information used by Eclipse.
.project file	Eclipse project information.
.pdp file	The packaged portlet that can be deployed on the portal.

Portlet Categories

When you create portal pages on the Portal server, you can find portlets organized in terms of the types (or categories) of actions they perform. Setting a portlet's category is one of the tasks you perform as part of creating the portlet. The Portlet Developer provides a set of default categories you can use:

Use this category...	If the portlet is used to...
SDK	Have a generic purpose. This category is a placeholder until you determine the proper category for the portlet.
Communication	Facilitate communication among portal users.
Content Management	Manage content elements such as folders, lists of links, publishing, and searching.
Developer	Provide developer's tools.
Drawing	Provide mechanisms for placement of images and text.
System	Provide mechanisms for navigation and other elements of Web page infrastructure.
Internet - Business/Stocks	Make Web-based sources of business-related information available.
Internet - News/Weather	Make Web-based sources of news and weather information available.
Internet - Search	Make Web-based search engines available.
Internet - Shopping	Make Web-based shopping sites available.
Internet - Tools	Make Web-based information tools available.

You can also create your own set of portlet categories. If you type in a new category during the creation of a portlet project, that category is deployed to the Portal server along with the portlet.

Note: webMethods Portal is case sensitive with regard to category names.

The Portlet Development Cycle

The portlet development cycle has the following phases:

This phase...	Is described here...
Create the portlet	"Adding New Portlet Projects" on page 29
Develop the portlet	"Developing the Portlet" on page 33
Prepare to deploy	"Building the Portlet Deployment Package" on page 34
Deploy the portlet	"Deploying the Portlet to the Portal Server" on page 34
Test the portlet	"Testing the Portlet" on page 34

Adding New Portlet Projects

In the Portlet Developer, each portlet is an Eclipse project. There are three ways to add portlet projects:

This way to add projects...	Is described here...
Create a new portlet	"Creating a New Project" on page 29
Import source files from an existing portlet	"Importing Existing Components" on page 31
Clone a portlet that exists in the workspace	"Cloning Existing Portlets" on page 32

Creating a New Project

To create a new portlet, you create what is known in Eclipse as a *project*.

To create a new project (portlet)

1 In Eclipse, make sure the webMethods Portlets perspective is open, as described in "Getting Started with the Portlet Developer" on page 25.

2 On the **File** menu, choose **New** and **Project**.

3 In left panel of the New Project dialog box, select **webMethods Portal**.

4 In the right panel of the New Project dialog box, choose the type of portlet you want to create or import and click **Next**.

You can choose from among the follow portlet types:

Portlet Choice	Description
Import Existing Component	Imports the source code for an existing portlet that does not currently exist in the Eclipse workspace. See "Importing Existing Components" on page 31.
Clone Existing Portlet	Creates a new portlet from an existing portlet. See "Cloning Existing Portlets" on page 32.
New Generic Portlet	Creates an empty portlet into which you can place Web service calls, data base queries and whatever other code you may need.
New Wizard Portlet	Creates a new portlet that has one or more layouts the user can step through sequentially using Previous and Next buttons. At the end of the sequence, some final action is typically performed, based on the data collected on all pages.
New Tabs Portlet	Creates a new portlet that has one or more independent layouts. At the top of each layout there is a tab control that displays one tab for each portlet layout. Clicking the tab makes the corresponding layout active.
New Parameterized URL Portlet	Creates a new portlet that displays content from a URL using query parameters.

5 In the **Portlet Name** field of the **New wmPortal Portlet Project** dialog box, type the internal name by which you can refer to the portlet programmatically.

6 In the **Portlet Title** field, type the display name of the portlet.

7 In the **Portlet Description** field, type a description of the portlet.

8 From the **Required Portal Version** list, choose the version of webMethods Portal.

By default, the current version is selected. Select an earlier version if the portlet you are creating is intended to run with that earlier version of webMethods Portal.

9 From the **Category** list, choose a category or, if you want a category that does not appear in the list, type a new category in the field.

For a list of default categories provided with the Portlet Developer, see "Portlet Categories" on page 28.

10 If you are creating something other than a generic portlet, make other selections as appropriate.

11 (Optional) In the **Project Contents** panel, select a location for the portlet to reside while it is being developed.

The default is to use the Eclipse workspace.

12 Click **Finish**.

The Eclipse project is created and the portlet appears in the **wmPortal Components** view.

There are some portlet attributes that you can modify after you have created the portlet. For information on modifying portlets, see "Modifying Portlet Attributes" on page 41.

Importing Existing Components

You can create a new Eclipse project by importing the source code for a portlet created elsewhere. This feature is useful if you are collaborating with others working at different locations. You can use the Portlet Developer to modify the imported portlet, if needed, and deploy it to the Portal server.

To import source code for a portlet

1 In Eclipse, make sure the webMethods Portlets perspective is open, as described in "Getting Started with the Portlet Developer" on page 25.

2 On the **File** menu, choose **New** and **Project**.

3 In left panel of the New Project dialog box, select **webMethods Portal**.

4 In the right panel of the New Project dialog box, click **Import Existing Component** and **Next**.

5 In the **Existing Portlet Source Code** panel, click **Browse** and navigate to the location of the source directory.

The portlet source folder is equivalent to the top level of the portlet as seen in the Navigator view, as described in "Elements in the Navigator View" on page 27.

6 Select the source directory and click **OK**.

7 Click **Finish**.

The portlet is added to the Eclipse workspace and appears in the **Navigator** and **wmPortal Components** views.

You can now deploy the portlet or modify it as needed.

Cloning Existing Portlets

You can create a new Eclipse project by cloning a portlet that already exists. You can then use the Portlet Developer to modify the cloned portlet as needed, and deploy it to the Portal server.

To clone an existing portlet

1. In Eclipse, make sure the webMethods Portlets perspective is open, as described in "Getting Started with the Portlet Developer" on page 25.

2. On the **File** menu, choose **New** and **Project**.

3. In left panel of the New Project dialog box, select **webMethods Portal**.

4. In the right panel of the New Project dialog box, click **Clone Existing Portlet** and **Next**.

5. In **Portlet Name** field of the **New wmPortal Portlet Project** dialog box, type the internal name by which you can refer to the cloned portlet programmatically.

6. In the **Portlet Title** field, type the display name of the cloned portlet.

7. In the **Portlet Description** field, type a description of the cloned portlet.

8. From the **Required Portal Version** list, choose the version of webMethods Portal.

   By default, the current version is selected. Select an earlier version if the portlet you are creating is intended to run with that earlier version of webMethods Portal.

9. From the **Category** list, choose a category, or type a new category, which will become a part of the list.

   For a list of default categories provided with the Portlet Developer, see "Portlet Categories" on page 28.

10. In the **Existing Portlet Source Code** panel, choose the **Retrieve From Local Filesystem** option, click **Browse**, and navigate to the location of the source directory.

    The portlet source folder is equivalent to the top level of the portlet as seen in the Navigator view, as described in "Elements in the Navigator View" on page 27.

11. Click **Finish**.

    The portlet is added to the Eclipse workspace and appears in the **Navigator** and **wmPortal Components** views.

You can now deploy the portlet or modify it as needed.

Developing the Portlet

When you develop a portlet, you create an empty portlet and then provide content and a user interface.

Create the User Interface

Portlet properties describe the content, the user interface, and the business logic provided by a portlet. A property group combines a set of portlet properties into a visual and logical group. For more information on portlet properties, see "Working with Property Groups" on page 42.

Add New Pages

A Portlet Controller Architecture (PCA) layout is a JSP page associated with a portlet. If you create a wizard or a set of tabs, you need a PCA layout for each wizard page or tab in the portlet. When creating a new portlet, if you choose a new wizard portlet or a new tabs portlet, the Portlet Developer creates multiple PCA layouts for you.

For more information about PCA layouts, see "Working with PCA Layouts" on page 48.

Create Actions

PCA methods are the actions performed by the portlet. If you want the portlet to perform an action on the server side, you create a method to perform the action. If you create a wizard portlet, the Portlet Developer creates navigation PCA methods for you. The use of PCA methods requires knowledge of the Java programming language. For more information about PCA methods, see "Working with PCA Methods" on page 51.

Add Links to Web Services

Web service client portlets can communicate with remote Web service applications. For example, you can create a Web service client portlet that displays the content of any Web service, including those exposed by Integration Server. When you use the Portlet Developer to create a Web reference, it also creates the portlet properties, PCA layouts, and PCA methods needed for it to function properly. For information on adding Web references, see "Working with Web References" on page 54.

Add Database Queries

A database query is capable of performing SQL queries against a data source that is accessible from your Portal server. When you use the Portlet Developer to create a database query, it also creates the portlet properties, PCA layouts, and PCA methods needed for it to function properly. For information on adding queries to a portlet, see "Working with Data Queries" on page 63.

Building the Portlet Deployment Package

A portlet can only operate within the portal run-time environment. To deploy the portlet into the portal you first need to build the portlet deployment package from within the Portlet Developer.

To build a portlet deployment package

1 In the **wmPortal Components** view of the Eclipse IDE, select the portlet for which you want to build a portlet deployment package.

2 At the top of the **wmPortal Components** view, click the Build Portlet icon.

The **Console** view opens on the right side of the Eclipse IDE. If a BUILD SUCCESSFUL message appears at the end of the build process, the portlet is ready for deployment.

3 If the BUILD SUCCESSFUL message does not appear in the **Console** view, use the build error messages to correct problems and then build the portlet again.

Deploying the Portlet to the Portal Server

After the portlet is successfully built, you can deploy it to the portal.

To deploy a portlet to the Portal server

1 Make sure the Portal server associated with this instance of the Portlet Developer is running.

2 In the **wmPortal Components** view of the Eclipse IDE, select the portlet you want to deploy.

3 At the top of the **wmPortal Components** view, click the Deploy Portlet icon.

If the Deploy Portlet icon is unavailable, the portlet deployment package does not exist. See "Building the Portlet Deployment Package" on page 34.

4 If the Authentication Required dialog box appears, type the appropriate entries in the **UserName** and **Password** fields, and click **OK**.

5 When the wmPortal Component Deployment window displays a message that the portlet has deployed successfully, click **OK**.

Testing the Portlet

After the portlet is deployed to the Portal server, you can test it directly from the Eclipse IDE. Alternatively, you can place the portlet on a portal page and test it there. For information on creating a portal page, see "Creating a Portal Page" on page 109.

To test a portlet from the Eclipse IDE

1. Make sure the Portal server associated with this instance of the Portlet Developer is running.
2. In the **wmPortal Components** view of the Eclipse IDE, select the portlet you want to test.
3. At the top of the **wmPortal Components** view, click the View Portlet in Browser icon.

   If you receive an error message indicating that the file is not found, the portlet has not been deployed to the Portal server. See "Deploying the Portlet to the Portal Server" on page 34.
4. If the Authentication Required dialog box appears, type the appropriate entries in the **UserName** and **Password** fields, and click **OK**.
5. In the temporary portal page that appears in the browser window, test the portlet as needed.

Tip! If you run into problems, don't forget to check the webMethods Advantage Web site. See **Troubleshooting Information** on page 13.

Upgrading from an Earlier Release of the Portlet Developer

If you are upgrading from an earlier version of the Portlet Developer, you need to ensure that your portlets can use the latest features of the Portal server.

To upgrade existing portlets to the current release

1. In Eclipse, make sure the webMethods Portlets perspective is open, as described in "Getting Started with the Portlet Developer" on page 25.
2. On the **Project** menu, choose **Clean**.
3. Choose the **Clean all projects** option.

   As al alternative, you can select which projects to be cleaned.
4. Click **OK**.

Portlet Types

When you create a new portlet, you can create a generic portlet or you can create another type of portlet containing prebuilt features that make it easier for you to develop the finished portlet.

This new portlet...	Is described here...
Generic	"New Generic Portlet" on page 36
Wizard	"New Wizard Portlet" on page 36
Tabs	"New Tabs Portlet" on page 37
Parameterized URL	"New Parameterized URL Portlet" on page 37

New Generic Portlet

A new generic portlet is an empty portlet from which you can develop most any type of finished portlet.

To create a new generic portlet

1. Create the portlet as described in "Creating a New Project" on page 29 and, when you reach step 4, choose **New Generic Portlet**.
2. Complete the rest of the steps and click **Finish**.

The new generic portlet contains one default property group and one default PCA layout; everything else, you must create yourself.

New Wizard Portlet

A new wizard portlet has a set of wizard pages (PCA layouts) already created for you.

\

To create a new wizard portlet

1. Create the portlet as described in "Creating a New Project" on page 29 and, when you reach step 4, choose **New Wizard Portlet**.
2. From the **Number of Wizard Pages** list, choose the number of pages your wizard portlet is to contain.
3. Complete the rest of the steps and click **Finish**.

The new wizard portlet contains a new property group for each page, a new PCA layout for each page, and a set of PCA methods for use in navigating among the pages.

For an example of building a wizard portlet, see "Developing a Wizard Portlet" on page 84.

New Tabs Portlet

A new tabs portlet has one or more independent layouts. At the top of each layout there is a tab control that displays one tab for each portlet layout. Clicking the tab makes the corresponding layout active.

To create a new tabs portlet

1 Create the portlet as described in "Creating a New Project" on page 29 and, when you reach step 4, choose **New Tabs Portlet**.

2 From the **Number of Tabs** list, choose the number of pages your tabs portlet is to contain.

3 Complete the rest of the steps and click **Finish**.

The new tabs portlet contains a new property group for each page and a new PCA layout for each page.

For an example of building a tabs portlet, see "Developing a Tabs Portlet" on page 87.

New Parameterized URL Portlet

A new parameterized URL portlet allows you to create a portlet that issues a query on a Web page, such as to perform a search. When you create the portlet, you specify a base URL from which the query is performed.

To create a new parameterized URL portlet

1 Create the portlet as described in "Creating a New Project" on page 29 and, when you reach step 4, choose **New Parameterized URL Portlet**.

2 In the **Base URL** field, type the URL of the Web page from which to perform the query.

3 Complete the rest of the steps and click **Finish**.

The new portlet contains the urlParameters property group. In this property group, create the properties that represent the parameters to be used in the query. At run time, the user types a value into a portlet input field and the Portal server appends the resulting parameters to the URL.

CHAPTER 3

Portlet Development

Access to the Portlet Developer

To use the Portlet Developer, you need to have access to the Portal server as a member of the Portlet Developer group or have access to the Developer Commands Security Realm.

Maintaining a Portlet

After you have created a portlet, as described in "Creating a New Project" on page 29, you can modify portlet attributes and also edit the portlet descriptor file. There are several tasks you can perform in maintaining portlets:

This task...	Is described here...
To learn about portlet attributes	"Attributes of a Portlet" on page 40
Modify portlet attributes	"Modifying Portlet Attributes" on page 41
Edit portlet descriptor files	"Editing the Portlet Descriptor Files" on page 42

Attributes of a Portlet

When you create or modify a portlet, there are attributes you can set that control how the portlet functions.

For this attribute...	You do this...
Title	Type the display name of the portlet.
Description	Type a description of the portlet.
Category	Choose an existing portlet category or, if you want a category that does not appear in the list, type the category into the field. For a list of default portlet categories, see "Portlet Categories" on page 28.
Version Number	Assign a version number for the portlet. By default, the original portlet is assigned the version number 1.0.
Required Portal Version	Type the version of webMethods Portal. By default, the current version is selected. Select an earlier version if the portlet you are creating is intended to run with that earlier version of webMethods Portal.

For this attribute...	You do this...
Initial Status	From the Initial Status list, choose the portlet status from among these values: **enabled** The portlet is enabled for publishing in the page designer. **disabled** The portlet cannot be published. **hidden** New instances of the portlet cannot be published. Any previously published instances of the portlet continue to function.
Is Acl free	Select this check box if you do not want ACL checking to occur for this portlet. This option provides improved performance.
PCA Controller Self-Destruct Limit	From the list, select a click count that describes how many portal page requests that do not include the portlet should be allowed before the portlet controller object stops itself. (Think of this value as the number of clicks away from a page.) If you set that value to -1, the controller object lives until the user logs out. If you want a click count higher than can be selected from the list, type the value into the field.
PCA Type	From the PCA Type list, choose the Portlet Controller Architecture (PCA) from among these values: **session** Portlet state is maintained in the session scope. The session PCA is an architecture used in previous versions of Portal. **restful** Portlet state is maintained in the portlet URL, which is superior to the old session model. The default.

Modifying Portlet Attributes

To modify the attributes of a portlet, use the following procedure.

To modify the attributes of a portlet

1. In the **wmPortal Components** view of the Eclipse IDE, right-click a portlet; on the menu, click **Modify Portlet Attributes**.
2. In the Update Portlet Attributes dialog box, change the attributes as needed.

   For more information, see "Attributes of a Portlet" on page 40.
3. Click **Finish**.

   The Portlet Developer modifies the attributes of the portlet.

Editing the Portlet Descriptor Files

The portlet descriptor files contain configuration data for the portlet. The portlet.xml file is compliant with the JSR 168 Portlet Specification while the wm_portlet.xml file contains webMethods extensions to JSR 168.

Note: Although the portlet descriptors are editable XML files, it is not recommended that you edit them unless you cannot accomplish your goal using the normal features of the Portlet Developer.

To edit the portlet descriptor files

1 In the **wmPortal Components** view of the Eclipse IDE, right-click the portlet for which you want to edit a portlet descriptor; on the menu, click one of the following:

This menu item...	Opens this file in the editor...
Edit portlet.xml	The portlet.xml file, which is JSR 168 compliant.
Edit wm_portlet.xml	The wm_portlet.xml file, which contains webMethods extensions to JSR 168.

2 To save changes, on the **File** menu, click **Save**.

Working with Property Groups

Portlet properties describe the content, the user interface, and the business logic provided by a portlet. A property group combines a set of portlet properties into a visual and logical group. You can change the existing properties in the group, add new properties to the group, or remove properties from the group. Additionally, you can have multiple groups displayed on a single layout. There are several tasks you can perform in working with property groups:

These tasks...	Are described here...
Create, modify, and delete property groups, and change the order in which they appear	"Managing Property Groups" on page 43
Create, modify, and delete properties, and change the order in which they appear	"Managing Properties in a Property Group" on page 44
Set attributes that control how a property functions	"Attributes of a Property" on page 46

Managing Property Groups

Using the Portlet Developer, you can create, modify, and delete property groups, and change the order in which they appear.

Creating a Property Group

To create a property group associated with a portlet, use the following procedure.

To create a property group

1. In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **Property Groups** component.
2. Right-click **Property Groups**; on the menu, click **Create Property Group**.
3. In the **Internal Name** field of the New Portlet Property Group dialog box, type an internal name by which you can refer to the group programmatically.
4. In the **Display Name** field, type a display name for the property group.
5. Click **Finish**.

Modifying a Property Group

To modify the internal and display names for a property group, use the following procedure.

To modify a property group

1. In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **Property Groups** component.
2. Expand the **Property Groups** component, right-click the property group you want to modify, and on the menu, click **Modify Property Group**.
3. In the Modify Portlet Property Group dialog box, change these attributes as needed:

For this attribute...	Do this...
Internal Name	Type an internal name by which you can refer to the group programmatically.
Display Name	Type a display name for the portlet.

4. Click **Finish**.

Changing the Order of Property Groups in a Portlet

If a portlet PCA layout needs to display a set of property groups, it uses the property group order to control how the property groups are displayed. You can change the order in which property groups are displayed in a portlet. On the Portal server, the portlet properties page shows the values of all of the visible portlet properties; the property group order determines the order in which they are displayed on the screen.

To change the order of property groups in a portlet

1 In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **Property Groups** component.

2 Right-click **Property Groups**; on the menu, click **Change Property Group Order**.

3 In the Order Property Groups dialog box, select a property group to be moved and then click **Move Up** or **Move Down**, as appropriate.

4 After the order of property groups is set, click **Finish**.

Deleting a Property Group

To delete a property group from a portlet, use the following procedure.

To delete a property group

1 In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **Property Groups** component.

2 Expand the **Property Groups** component, right-click the property group you want to delete, and on the menu, click **Delete Property Group**.

The property group is deleted.

Managing Properties in a Property Group

Using the Portlet Developer, you can create, modify, and delete properties, and change the order in which they appear.

Creating a Property

To create a property within a property group, use the following procedure.

To create a property

1. In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **Property Groups** component.
2. Expand **Property Groups**, right-click a property group, and on the menu, click **Create Property**.
3. In the **Internal Name** field of the New Portlet Property dialog box, type a name by which you can refer to the property programmatically.
4. In the **Display Name** field, type a display name for this portlet property.
5. Complete other attributes of the property that control how it handles user input.

   For more information, see "Attributes of a Property" on page 46.
6. Click **Finish**.

   The Portlet Developer adds the property to the tree, and adds getProperty and setProperty methods to the portlet bean so you can use the property with Java code.

Modifying a Property

To modify the attributes of a property, use the following procedure.

To modify a property

1. In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **Property Groups** component.
2. Expand the **Property Groups** component.
3. Expand the property group and right-click the property you want to modify; on the menu, click **Modify Property**.
4. In the Modify Portlet Property dialog box, change the attributes as needed.

   For more information, see "Attributes of a Property" on page 46.
5. Click **Finish**.

Changing the Order of Properties in a Property Group

To change the order in which properties are displayed in a property group, use the following procedure.

To change the order of properties in a property group

1. In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **Property Groups** component.
2. Expand the **Property Groups** component.
3. Right-click the property group; on the menu, click **Change Property Order**.
4. In the Order Properties dialog box, select a property to be moved and then click **Move Up** or **Move Down**, as appropriate.
5. After the order of properties is set, click **Finish**.

Deleting a Property

To delete a property from a property group, use the following procedure.

To delete a property from a property group

1. In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **Property Groups** component.
2. Expand the **Property Groups** component.
3. Expand the property group, right-click the property you want to delete, and on the menu, click **Delete Property**.

   The property is deleted.

Attributes of a Property

When you create or modify a portlet property, there are attributes you can set that control how the property functions.

For this attribute...	Do this...
Internal Name	Type a name by which you can refer to the property programmatically.
Display Name	Type a display name for this portlet property. This value appears as the property title on the portal page.

For this attribute...	Do this...	
Scope	Choose the way a property value is to be stored. Choose from the following list:	
	Value Stored Per Portlet Instance	Different instances of the portlet on different portal pages have their own values. The value is stored in the Portal database. All users who access the portlet instance see the same value.
	Value Shared by All Portlet Instances	All instances of the portlet share the same value.
	Value Stored Per User	Each individual user has a separate property value.
	Value Stored Per Session	The value is stored for an individual user until that user logs out of the portal session or until the self-destruct limit is reached (see "Attributes of a Portlet" on page 40).
Property Editor Type	Select a property editor. Click **Change**, in the Property Editor type window, select a property editor, and click **OK**. For information on property editors, see "Property Editor Reference" on page 149.	
Property Editor Parameters	Modify parameters in the property editor, as described in "Property Editor Reference" on page 149.	
Default Value	Type a default value, if appropriate, in this field.	
UI Rendering Hints	Controls the way a property is rendered on the portal page. You can select any combination of these options:	
	Is Required	Appends a red asterisk (*) to the field. Selecting this option does not enforce the use of a required field. You must use the **Validation RegEx** field in the **Property Editor Parameters** attribute for that purpose.
	Is ReadOnly	Causes the input field to be read only.
	Is Hidden	Causes the input field to be hidden on the UI.

Working with PCA Layouts

A Portlet Controller Architecture (PCA) layout is a JSP page associated with a portlet. If you create a wizard or a set of tabs, you need a PCA layout for each wizard page or tab in the portlet. By associating a PCA layout with a portlet property group, you create the user interface for that page.

Using the Portlet Developer, you can create, modify, and delete PCA layouts, and change the order which they appear. Here are some tasks you can perform in working with PCA layouts:

This task...	Is described here...
Add additional PCA layouts to a portlet	"Creating PCA Layouts" on page 48
Modify PCA layout attributes	"Modifying PCA Layout Attributes" on page 50
Change a PCA layout directly in the JSP page	"Editing the JSP Page" on page 50
Delete a PCA layout	"Deleting a PCA Layout" on page 51

Creating PCA Layouts

When you create a new generic portlet, the Portlet Developer creates one default PCA layout. You can create additional PCA layouts as you add functionality to your portlet.

To create a PCA layout

1. In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **PCA Layouts** component.
2. Right-click **PCA Layouts**; on the menu, click **Create PCA Layout**.
3. In the **Layout Name** field of the New PCA layout dialog box, type an internal name by which you can refer to the layout programmatically.
4. In the **Display Name** field, type a display name for the PCA layout.
5. If the PCA layout is to be the default layout, select the **Is Default Layout** check box.
6. From the **Implementation Pattern** list in the **Initial Layout Contents** panel, choose from among these implementation patterns:
   - **Layout that displays a Portal Form**
   - **Tabs Layout that displays a Portal Form**
   - **Empty Page**

- **Empty Page with Tabs**
- **Tabs Page that displays a property group**

If you are creating this PCA layout within something other than a generic portlet, additional patterns may be available in the list.

7 Optionally, do one of the following:

If this list appears...	Do this...
Associated Property Group	From the list, choose a property group. The list is made up of property groups created within the portlet.
Portal Form	From the list, choose a portal form. the list is made up of forms created within the portlet using the Form Builder. See Chapter 4, "Using the Form Builder" for more information.

8 If you are associating the PCA layout with a portal form, in the UI Binding field, select how the form will be bound to the UI:

Binding	Description
Dynamic	The form UI is auto-generated when the form is built.
Static	The form UI is added directly to the PCA layout JSP page. The UI contains only the controls that existed in the form at the time the PCA layout was created.

Note: If you select dynamic binding, changes to the form are reflected in the PCA layout UI automatically.

9 Click **Finish**.

Modifying PCA Layout Attributes

To modify the layout and display names for a PCA layout, use the following procedure.

To modify a PCA layout

1. In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **PCA Layouts** component.
2. Expand the **PCA Layouts** component, right-click the PCA layout you want to modify, and on the menu, click **Modify PCA Layout Properties**.
3. In the Modify PCA Layout Properties dialog box, change these attributes as needed:

For this attribute...	Do this...
Layout Name	In this field, type an internal name by which you can refer to the group programmatically.
Display Name	In this field, type a display name for the portlet.

4. Click **Finish**.

Editing the JSP Page

If you are familiar with the Java programming language and JSP pages, you can open the PCA layout in the text editor and make changes directly.

To edit the JSP page of a PCA layout

1. In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **PCA Layouts** component.
2. Expand the **PCA Layouts** component, right-click the PCA layout you want to edit, and on the menu, click **Edit PCA Layout**.

   The JSP page opens in the text editor. Another way to open the JSP for editing is to double-click the PCA layout.
3. To save changes, on the **File** menu, click **Save**.

Changing the Order of PCA Layouts in a Portlet

You can change the order in which PCA layouts are displayed in a portlet. If you are creating a wizard you need to make sure the order of PCA layouts matches the order in which wizard pages appear.

To change the order of PCA layouts in a portlet

1. In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **PCA Layouts** component.
2. Right-click **PCA Layouts**; on the menu, click **Change Layout Order**.
3. In the Order PCA Layouts dialog box, select a PCA layout to be moved and then click **Move Up** or **Move Down**, as appropriate.
4. After the order of PCA layouts is set, click **Finish**.

Deleting a PCA Layout

To delete a PCA layout from a portlet, use the following procedure.

To delete a PCA layout

1. In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **PCA Layouts** component.
2. Expand the **PCA Layouts** component, right-click the PCA layout you want to delete, and on the menu, click **Delete PCA Layout**.

   The PCA layout is deleted.

Working with PCA Methods

Portlet Controller Architecture (PCA) methods are the actions performed by the portlet. If you want the portlet to perform an action on the server side, you create a method to perform the action. If you create a wizard portlet, the Portlet Developer creates navigation PCA methods for you. The use of PCA methods requires knowledge of the Java programming language. Here are some tasks you can perform in working with PCA methods:

This task...	Is described here...
Create a PCA method	"Creating a PCA Method" on page 52
Set attributes that control how a method functions	"Modifying PCA Method Attributes" on page 52
See information about valid PCA method attributes	"Attributes of a PCA Method" on page 53

This task...	Is described here...
Edit the Java code for a PCA method directly	"Editing the PCA Method" on page 54
Delete a PCA method	"Deleting a PCA Method" on page 54

For an example that shows the use of PCA methods, see "Developing a Wizard Portlet" on page 84.

Creating a PCA Method

To create a PCA method for use in a portlet, use the following procedure.

To create a PCA method

1 In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **PCA Methods** component.

2 Right-click **PCA Methods**; on the menu, click **Create PCA Method**.

The Create PCA Method dialog box is displayed.

3 In the **Method Name** field of Create PCA Method dialog box, type an internal name by which you can refer to the method programmatically.

The **Method Name** field is the only required field in this dialog box.

4 As needed, complete other attributes for the PCA method, as described in "Attributes of a PCA Method" on page 53.

5 Click **Finish**.

The Portlet Developer creates the PCA method and adds it to the portlet bean for the portlet.

Modifying PCA Method Attributes

To modify attributes of a PCA method, use the following procedure.

To modify a PCA method

1 In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **PCA Methods** component.

2 Expand the **PCA Methods** component, right-click the PCA method you want to modify, and on the menu, click **Modify PCA Method Properties**.

3 In the Modify PCA Method Properties dialog box, change the attributes as needed.

For more information, see "Attributes of a PCA Method" on page 53.

4 Click **Finish**.

The Portlet Developer modifies the attributes of the PCA method.

Attributes of a PCA Method

When you create or modify a PCA method, there are attributes you can set that control how the method functions.

For this attribute...	Do this...
Method Name	Type an internal name by which you can refer to the method programmatically.
Display Name	Type a display name for the method.
Result Layout Name	From the **Result Layout Name** list, choose the PCA layout to be used in displaying a result. The list is made up of PCA layouts created for the portlet you are building.
Error Layout Name	From the **Error Layout Name** list, choose the PCA layout to be used if an error occurs. The list is made up of PCA layouts created for the portlet you are building.
Portlet Controller State	Choose how portlet controller state is kept. The portlet controller state is a set of session-based parameters the Portal server keeps to record the current state of a user's interaction with a portlet. You can leave the state unchanged, or clear the state before or after the method executes. From the **Portlet Controller State** list, choose a state for the portlet controller from among these choices: ■ Leave controller state unchanged ■ Clear controller state before the method executes ■ Clear controller state after the method executes ■ Clear controller state before and after the method executes

Editing the PCA Method

If you have created a PCA method, you need to complete the Java code that provides functionality for the method.

To edit the PCA method

1 In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **PCA Methods** component.

2 Expand the **PCA Methods** component, right-click the PCA method you want to edit, and on the menu, click **Edit PCA Method**.

   The portlet bean opens in the text editor at the location of the PCA method you want to edit. Another way to open the PCA method for editing is to double-click the PCA method.

3 To save changes, on the **File** menu, click **Save**.

Deleting a PCA Method

To delete a PCA method from a portlet, use the following procedure.

To delete a PCA method

1 In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **PCA Methods** component.

2 Expand the **PCA Methods** component, right-click the PCA method you want to delete, and on the menu, click **Delete PCA Method**.

   The PCA method is deleted.

Working with Web References

Web service client portlets can communicate with remote Web service applications. For example, you can create a Web service client portlet that displays the content of any Web service exposed by Integration Server. Unlike a Web application, which serves content using HTML, the Web service provides information to its subscribing clients, such as your portlet, using SOAP (Simple Object Access Protocol) as its communication protocol. Information about the Web service is contained in the WSDL (Web Services Description Language) file, which specifies the location of the service and the operations the service exposes.

Using the Portlet Developer, you can link to a Web service in any of several ways:

- Type in the URL of a WSDL file for the Web service.

- Browse to a Web service on a local file system.
- Select the Web service from a list of publicly available Web services at XMethods.com.

  The Portlet Developer retrieves the Web service list and, once you have selected the Web service of interest, retrieves the WSDL file for that Web service.
- Browse to a service or flow on Integration Server.

  The Portlet Developer creates a WSDL file for it.
- Use a UDDI (Universal Description, Discovery, and Integration) server to locate a Web service.

  UDDI servers provide registries of Web services and other electronic and non-electronic services.
- Browse to a Web service over webMethods Servicenet.

  A Web service that has been published to a node on webMethods Servicenet is available anywhere on the fabric, and is included in the built-in UDDI registry.

You can perform the following tasks in working with Web references:

This task...	Is described here...
Add a Web reference to a portlet	"Adding a Web Reference" on page 55
Create a wiring property for output parameters of a Web reference	"Result Wiring in a Web Reference" on page 60
Find a Web service that has been published to a node on webMethods Servicenet	"Searching webMethods Servicenet for Web Services" on page 61
Browse UDDI servers to locate a Web service	"Browsing UDDI Servers" on page 61
Delete a Web reference from a portlet	"Deleting a Web Reference" on page 62

Adding a Web Reference

To add a Web reference to a portlet, use the following procedure.

To add a Web reference to a portlet

1. In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **Web References** component.
2. Right-click **Web References**; on the menu, click **Add Web Reference**.

   The New Web Reference dialog box is displayed.

3 In the **Logical Name** field of the New Web Reference dialog box, type a name that will be unique among all logical names on the Portal server.

The Portlet Developer creates a property group having the name you provide in the **Logical Name** field.

4 In the **WSDL Location** list, either type the URL of a WSDL file for the Web service or choose one the following items from the list:

Choose this list item...	And do this...
Choose a local File	Browse to the location of the WSDL file on the local computer or on a mapped drive, and click **Open**.
Choose a Web Service from XMethods	Select a service from the XMethods list and click **OK**.
Choose a Web Service from Servicenet	In the **Inquire URL** field, type the inquiry URL for the built-in UDDI registry. If authentication is required, in the **User Name** and **Password** fields, type the login credentials Click **OK**. For more information, see "Searching webMethods Servicenet for Web Services" on page 61.
Browse an Integration Server	In the **Login Server** field, type the hostname of the computer where Integration Server is running. In the **User Name** and **Password** fields, type the Integration Server login credentials and click **OK**.
Browse a UDDI Server **Choose a TModel from a UDDI Server**	In the **Inquire URL** field, type the inquiry URL for the built-in UDDI registry. If required, in the **Publish URL** field, type the publish URL for the build-in UDDI registry. If authentication is required, in the **User Name** and **Password** fields, type the login credentials Click **OK**. For more information, see "Browsing UDDI Servers" on page 61.

5 If the Web service you are referencing requires authentication, select the **HTTP Authentication Required** check box and do the following in the **Authentication Credentials** panel (authentication credential properties are placed in the property group having the name provided in the **Logical Name** field in step 3):

a Click the **Value Source** column for the **Username** row and then click the button that appears in the right edge of the column.

b In the Input Source wizard, specify where the input value will come from, using one of these three choices:

Choose this option...	And then do this...
Use a Fixed Value	In the **Value** field, type a value that should always be used as the authentication username in this portlet.
Use a New Portlet Property	In the **New Property Name** field, type the name of a portlet property that is to contain the username. The default is authUsername.
Reuse an Existing Portlet Property	In the **Existing Property Name** tree view, expand the property group containing the existing property and then click the property to select it.

c Click **OK**.

d Click the **Value Source** column for the **Password** row and then click the button that appears in the right edge of the column.

e In the Input Source wizard, specify where the input value will come from, using one of these three choices:

Choose this option...	And then do this...
Use a Fixed Value	In the **Value** field, type a value that should always be used as an authentication password in this portlet.
Use a New Portlet Property	In the **New Property Name** field, type the name of a portlet property that is to contain the password. The default is authPassword.
Reuse an Existing Portlet Property	In the **Existing Property Name** tree view, expand the property group containing the existing property and then click the property to select it.

f Click **OK**.

6 Click **Next**.

The Portlet Developer saves a copy of the WSDL file locally, generates the client classes for the Web reference and then displays the next page in the wizard.

7 In the Select Web Reference Operations page, select the operations (methods) that should be included in the portlet and then click **Next**.

By default, all operations are selected. You must clear the operations you do not want to use. The selected operations appear in separate tabs on the following page.

8 For each parameter in each tab, do the following:

a Click the **Value Source** column for the parameter and then click the button that appears in the right edge of the column.

b In the Input Source wizard, specify where the input value is to come from, using one of these choices:

Choose this option...	And then do this...
Use a Fixed Value	In the **Value** field, type a value that should always be used as the input value for this parameter.
Use a New Portlet Property	In the **New Property Name** field, type the name of a portlet property that is to contain the input value.
Reuse an Existing Portlet Property	In the **Existing Property Name** tree view, expand the property group containing the existing property and then click the property to select it.

If you do nothing, the Portlet Developer uses a new portlet property and assigns it a unique name.

c Click **OK**.

9 Click **Next**.

The next page allows you to define how the data returned from each operation is to be displayed in the portlet UI. As with the previous page, each operation has its own tab. Each tab has the following panels:

On this panel...	You will see this...
Data Returned	Each tab represents an operation defined in the target Web service. The tree shows the structure of the data that the Web service operation is expected to return.
User Interface Parts	The currently selected pieces of the result data to be displayed and a Display Style that defines how to render each piece.
Result Wiring	A table into which you can place selected pieces of the result data that will be available as a wiring source for another portlet. See "Result Wiring in a Web Reference" on page 60.

10 As needed, expand elements in the **Data Returned** panel, select those to be available in the UI, and click **>>** for the **User Interface Parts** panel.

11 In the **User Interface Parts** panel, if you need to change the display style for a particular result, click the **Display Style** column and choose a style from the list.

If the parameter is...	You can choose from among these Display Styles...	
A string or number	**Property Line**	Displays a label and a value
	Raw Value	Displays a Java toString() value. Useful if you want to perform customizations on the logic.
A simple array of strings or numbers	**Tabular format**	Displays results horizontally with each field in a column.
	Raw Value	Displays a Java toString() value. Useful if you want to perform customizations on the logic.
A complex object	**Property Group**	Displays the object vertically, with each field as a property line.
A simple array of strings or numbers	**Tabular format**	Displays results horizontally with each field in a column.
	Raw Value	Displays a Java toString() value. Useful if you want to perform customizations on the logic.

12 As needed, take any of the following actions in the **User Interface Parts** panel:

To accomplish this...	Do this...
Move a result up one level	Select the result you want to move and click **Move Up**.
Move a result down one level	Select the result you want to move and click **Move Down**.
Remove a result	Select the result you want to delete and click **Remove Selected**.

13 If you want to make one or more data items in an operation available as a wiring source for another portlet (see "Result Wiring in a Web Reference" on page 60), do the following:

a Expand parameters in the **Data Returned** panel, select those to be used as wiring sources, and click **>>** for the **Result Wiring** panel.

a Click the **Value Destination** column for the parameter and then click the button that appears in the right edge of the column.

b In the Wiring Destination wizard, specify where the wiring value is to be stored, using one of these choices:

Choose this option...	And then do this...
Use a New Portlet Property	In the **New Property Name** field, type the name of a portlet property that is to contain the wiring value.
Reuse an Existing Portlet Property	In the **Existing Property Name** tree view, expand the property group containing the existing property and then click the property to select it.

If you do nothing, the Portlet Developer uses a new portlet property and assigns it a unique name.

c Click **OK**.

14 Click **Finish**.

The Portlet Developer adds several components to the portlet:

- A portlet property group with the same name as the Web Reference logical name. This property group contains a portlet property for changing the Web service endpoint address and, if needed, portlet properties for the authentication credentials that are needed to access the target Web service.
- A PCA method for each of the operations exposed by the target Web service.
- A pair of PCA layouts for each of the operations exposed by the target Web service. The first layout uses an HTML form to collect any inputs the operation requires. When the input form is posted, it calls the related PCA method, which executes the Web service operation. After the Web service returns, the Portal server displays the second layout, which provides the result.

Result Wiring in a Web Reference

When you add a Web reference to a portlet in the Portlet Developer, you can create a wiring property for output parameters of that Web reference. When you finish creating the Web reference you can find the result wiring properties under Property Groups for the portlet. The new property groups take the name *logical name_operation name*_wiring, where:

This element...	Is this...
logical name	The logical name of the Web reference.
operation name	The name assigned of the Web service operation.

After the portlet is deployed to a Portal server and is added to a portal page, the wiring properties are available on the Wiring properties page for the portlet. When the Portal server calls the Web service associated with the Web reference you have created, the server copies the results into the appropriate wiring property.

Searching webMethods Servicenet for Web Services

A Web service that has been published to a node on webMethods Servicenet is available anywhere on the fabric, and is included in the built-in UDDI registry.

To get a Web service from the fabric, you need to know the inquiry URL for the built-in UDDI registry. The registry may also require that you provide a publish URL, even through the Portlet Developer does not publish Web services, or a valid user name and password.

The publish and inquiry URLS are posted on the console of a Servicenet server. To start a console, enter the URL of any Servicenet server into a browser, followed by /console. For example, http://Servicenet_*server*:8004/examples/console, where Servicenet_*server* is the host name of a computer on which a Servicenet server is running.

When you search for a Web service in the fabric UDDI registry, the Select a Web Service window displays a table in which each row represents a Web service. The table has these columns: Table Name, WSDL's URL, and Description.

To choose a WSDL URL, click the row in which it appears and then click **OK**. The URL appears in the **WSDL Location** field of the New Web Reference window.

Browsing UDDI Servers

The Portlet Developer offers three ways to search a UDDI server for Web services: browsing the UDDI server, a service-only search of the server, and a tModel-only search of the server. Browsing the entire UDDI server takes longer to return data than either the services or tModel search, and may take longer for you to locate WSDL files.

To use a UDDI server, you need to know the inquiry URL for the server. Some UDDI servers also require that you provide a publish URL, even through the Portlet Developer does not publish Web services, or a valid user name and password. Go to the Web site for the UDDI server to get this information.

Browsing the Entire UDDI Server

Portlet Developer conducts a search of the entire registry of the UDDI server and displays a tree view in the Select a Web Service window with these primary nodes: Businesses, Bindings, Services, and TModels. To find the URL of a WSDL:

- Under the Businesses node, expand a business in the following sequence to find a valid URL: Services, an individual service, Bindings, (an individual) Binding, TModelInstances, Overview
- Under the top-level Services node, expand a service in the following sequence to find a valid URL: Bindings, (an individual) Binding, TModelInstances, Overview.
- Under the TModel node, expand a tModel and then expand the Overview node.

With the URL to the WSDL file displayed in the **WSDL's URL to use as web reference field**, click **OK**. The URL appears in the **WSDL Location** field of the New Web Reference window.

Performing Service and TModel Searches of Services

The Service or TModel (Technical Model) search is designed to be limited to services that contain WSDL content. The Select a Web Service window displays a table in which each row represents a service. The table has these columns: Table Name, WSDL's URL, and Description.

To choose a WSDL URL, click the row in which it appears and then click **OK**. The URL appears in the **WSDL Location** field of the New Web Reference window.

Deleting a Web Reference

To delete a Web reference from a portlet, use the following procedure.

To delete a Web reference

1. In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **Web references** component.
2. Expand the **Web references** component, right-click the Web reference you want to delete, and on the menu, click **Delete Web Reference**.

   The Web reference is deleted.

Working with Data Queries

A database query is capable of performing SQL queries against a data source. The data source must be accessible from your Portal server. Use the Data Queries component to add this capability to a portlet.

To add a database query to a portlet

1. In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **Data Queries** component.
2. Right-click **Data Queries**; on the menu, click **Add Database Query**.
3. If the Authentication Required dialog box appears, type the appropriate entries in the **UserName** and **Password** fields, and click **OK**.
4. In **Logical Name** field of the New Database Query dialog box, type a logical name for the database query.
5. From the DataSource list, choose the data source for the query.

   To appear in the list, the data source must be accessible from your Portal server. For more information, see the *webMethods Portal Administrator's Guide*.
6. In the **SQL Query** field, construct the database query to be sent by the portlet.
7. To add query parameters, select text from the **SQL Query** field, click **Add Query Parameters**, and add the parameters in the **Query Parameters** table.
8. To validate the query, click **Validate Query and Load Result Column Data**.

   You can check the results in the **Query Result Information** panel.
9. After the database query is correct, click **Finish**.

   The Portlet Developer creates the SQL query and creates a PCA layout.

webMethods.

CHAPTER 4

Using the Form Builder

What is the Form Builder?

The Form Builder in Portlet Developer is a tool you can use to create a form. The Form Builder uses the Graphical Editor Framework (GEF) plug-in to provide a canvas with rows and columns into which you can place the various elements of a form. To add the GEF to Eclipse, see "Installing the Graphical Editor Framework" on page 17. When you select an element on the canvas, the Properties view in the lower-right corner of Eclipse contains properties of the element you can modify.

The user interface of the Form Builder has the follow features:

Feature	Purpose	
Design tab	The interface you use to design the form. This tab has two elements:	
	Canvas	The area upon which you create the form. When you create a new form, the default configuration is a single-celled table containing a **Submit** button. The canvas does not present the form exactly as it will appear on a portal page at run time.
	Palette	On the left side of the Design tab, a grouping of design elements (property editors) from which to choose. Similar elements are grouped under Drawers.
Preview tab	A portal-page preview of the form you are creating.	

The Form Builder gives you access to property editors that make it easy to create the various elements that make up a form. To find information about the property editors, see "Property Editor Reference" on page 149.

The Form Builder declares the structure of a form. To show a form in the UI, you must create a new PCA layout and associate it with a form. You cannot use the Form Builder to update an existing PCA Layout.

Note: To use the Form Builder, you need to have access to the Portal server as a member of the Portlet Developer group or have access to the Developer Commands Security Realm.

Getting Started with the Form Builder

In using the Form Builder, you follow this basic workflow:

This task...	Is described here...
Create a new form	"Creating a New Form in the Form Builder" on page 67
Edit the form	"Editing a Form in the Form Builder" on page 67
Match a form with a PCA layout	"Adding a New PCA Layout to the Portlet" on page 68

There are some additional tasks you can perform in the Form Builder:

This task...	Is described here...
Delete a form	"Deleting a Form from a Portlet" on page 70
Use undo and redo	"Using Undo and Redo in the Form Builder" on page 70

Creating a New Form in the Form Builder

Before you can use Form Builder to create a new form, you must have the GEF plug-in installed in Eclipse. See "Installing the Graphical Editor Framework" on page 17.

To create a form to be modified using the Form Builder, use the following procedure.

To create a new form for use in the Form Builder

1 In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **Portal Forms** component.

2 Right-click **Portal Forms**; on the menu, click **Create New Portal Form**.

3 In the **Portal Form Name** field of the New Portal Form dialog box, type an internal name by which you can refer to the form programmatically, and click **Finish**.

   The Form Builder creates the new form and lists it under the **Portal Forms** component. The form takes the name you typed and has an extension of `.wpform`.

After you have created a new form, you can open it in the Form Builder, as described in "Editing a Form in the Form Builder" next in this chapter.

Editing a Form in the Form Builder

The Eclipse IDE does not display the Form Builder until you want to edit a form.

To edit a form in the Form Builder

1 In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **Portal Forms** component.

2 Expand the **Portal Forms** component, right-click the form you want to edit, and on the menu, click **Edit Portal Form**.

The Form Builder is displayed, having the following tabs in the lower left corner:

This tab...	Does this...
Design	Displays the form canvas on which to build or modify the form. The default canvas contains a Submit button to be used in submitting the form.
Preview	Displays a preview of how the form will be displayed on the server. You may be required to provide your username and password for connection to the Portal server.

3 Perform editing tasks as needed within the Form Builder:

These tasks...	Are described here...
Manage tables, rows and columns	"Using Tables in the Form Builder" on page 70
Add, move, or delete property editors	"Manipulating Property Editors in the Form Builder" on page 74

4 On the **File** menu, click **Save**.

Adding a New PCA Layout to the Portlet

The Form Builder creates a template to be used when adding a new PCA layout to your portlet. After you have created the form in the Form Builder, you need to add a new PCA layout to the portlet as described here.

Note: If you select static UI binding when you create your PCA layout, there is no synchronization between the form and the generated PCA layout. If you change the form in the Form Builder, you need to regenerate the PCA layout manually for the changes to be reflected in the UI.

To add a new PCA layout for the form you have created

1. In the **wmPortal Components** view of the Eclipse IDE, expand the portlet to display its components, including the **PCA Layouts** component.
2. Right-click **PCA Layouts**; on the menu, click **Create PCA Layout**.
3. In the **Layout Name** field of the New PCA layout dialog box, type an internal name by which you can refer to the layout programmatically.
4. In the **Display Name** field, type a display name for the PCA layout.
5. If the PCA layout is to be the default layout, select the **Is Default Layout** check box.
6. From the **Implementation Pattern** list in the **Initial Layout Contents** panel, choose an implementation pattern that allows you to associate it with a portal form:
   - **Layout that displays a Portal Form**
   - **Tabs Layout that displays a Portal Form**
7. From the **Portal Form** list, choose a portal form.

   The list contains portal forms that have already been created within the portlet.
8. In the UI Binding field, select how the form will be bound to the UI:

Binding	Description
Dynamic	The form UI is auto-generated when the form is built.
Static	The form UI is added directly to the PCA layout JSP page. The UI contains only the controls that existed in the form at the time the PCA layout was created.

Note: If you select dynamic binding, changes to the form are reflected in the PCA layout UI automatically.

9. Click **Finish**.

Deleting a Form from a Portlet

To delete a form from a portlet, use the following procedure.

To delete a form from a portlet

1. In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **Portal Forms** component.
2. Expand the **Portal Forms** component, right-click the form you want to delete, and on the menu, click **Delete Portal Form**.

   The form is deleted.

Using Undo and Redo in the Form Builder

In the Form Builder, you can perform multiple-level undo operations and multiple-level redo operations. These operations are available in two ways within the Form Builder user interface:

- On the **Edit** menu, click either **Edit ▸ Undo** or **Edit ▸ Redo**.
- Right-click anywhere within the **Design** tab. On the menu, click **Undo** or **Redo**.

Undo and redo operations undertaken from either location in the user interface have the same effect.

Using Tables in the Form Builder

Tables provide structure and organization to a form. By placing property editors in the cells of one or more tables, you populate the form and determine the positions where various elements are displayed. There are some basic things you can do with tables in a form:

This task...	Is described here...
Add a table	"Adding a Table to a Form" on page 71
Delete a table	"Deleting a Table from a Form" on page 71
Adjust rows and columns	"Inserting or Removing Rows and Columns in a Table" on page 72
Select multiple table cells	"Selecting Cells in a Table" on page 73

Adding a Table to a Form

When creating a form in the Form Builder, you use one or more tables to create structure and organization. You need to add at least one table to contain other elements of the form.

To add a table in the Form Builder.

1 In the Form Builder, click the **Design** tab in the lower left corner to bring it to the front.

2 In the Palette, click the **Form Features** Drawer to display the **Table** tool.

3 Click **Table** and then click an empty area of the canvas.

4 In the Table Dimensions dialog box, do the following:

In this field...	Do this...
Initial Rows	Type the number of rows for the table. The default is two rows.
Initial Columns	Type the number of columns for the table. The default is two columns.

5 Click **OK**.

6 If you need to move the table, such as to place it above the default table containing the Submit button, drag the table to its new location.

Deleting a Table from a Form

To delete a table from a form, use the following procedure.

To delete a table from a form

1 In the Form Builder, click the **Design** tab in the lower left corner to bring it to the front.

2 Right-click the title area of the table to be deleted.

3 On the menu, click **Delete**.

You can also delete a table by selecting it and pressing DELETE.

Inserting or Removing Rows and Columns in a Table

To insert or remove rows and columns in a table, use the following procedure.

To insert rows and columns in a table

1. In the Form Builder, click the **Design** tab in the lower left corner to bring it to the front.
2. In the table, right-click a cell next to where you want to insert the row or column.
3. From the menu, click one of these menu items:

Click this menu item...	To do this...
Insert Column to Left	Insert a new column to the left of the selected cell.
Insert Column to Right	Insert a new column to the right of the selected cell.
Insert Row Above	Insert a new row above the selected cell.
Insert Row Below	Insert a new row below the selected cell.

To remove rows or columns from a table

1. In the Form Builder, click the **Design** tab in the lower left corner to bring it to the front.
2. In the table, right-click a cell in the row or column you want to remove.
3. From the menu, click one of these menu items:

Click this menu item...	To do this...
Remove Column	Remove the column containing the selected cell.
Remove Row	Remove the row containing the selected cell.

You cannot remove the last row or column in a table.

Selecting Cells in a Table

If you need to select more than one cell at the same time, there are multiple ways to do so:

To use this...	Do this...
The Marquee tool to select all cells in a range	In the Form Builder Palette, click **Marquee** and then drag across the group of table cells to be selected. To clear a selected cell, use CTRL+click.
CTRL+click to select or clear a group of individual cells	In the Form Builder Palette, click **Select**. CTRL+click each cell to select it. To clear a selected cell, use CTRL+click again.

Specifying the Action for a Form

When a form is deployed on a portal page, clicking the Submit button causes the contents of the form to be posted to any destination that accepts a form. This destination can be an external URL, such as a search page on a Web site, or another portal URL. You can also post the contents of the form to a PCA Layout or a PCA Method within the portlet.

To specify the action for a form in the Form Builder.

1. In the Form Builder, click the **Design** tab in the lower left corner to bring it to the front.
2. Click an empty area of the canvas to select the form.
3. In the **Properties** view, click the **Action** property and then click the button that appears in the right edge of the row.
4. In the Form Action dialog box, do one of the following:

Select this option...	And do this...
URL (any)	Type the URL where the form should be posted.
Portlet Controller	From the **PCA Method** list, choose a PCA Method that exists within the portlet. From the **PCA Layout** list, select a PCA Layout that exists within the portlet. You can choose from either list or from both.

5. Click **OK**.

6 (Optional if you want to change the method used to pass the form data.) In the **Properties** view, click the **Method** property, click the button that appears in the right edge of the row, and in the list, select either of these two methods:

This method...	Does this
POST	Passes form data in the body of the HTTP request. This method is preferable because it can safely pass long data strings. POST is the default.
GET	Passes form data as a string appended to the URL after a question mark. This method is not suitable for forms with large amounts of data.

Manipulating Property Editors in the Form Builder

Property editors provide the features that make up the form you are creating. You can add text fields, radio buttons, lists, and other elements to the form by placing them in table cells on the Form Builder canvas and editing properties as needed. The following sections describe some basic tasks you can perform on property editors:

This task...	Is described here...
Adding a property editor	"Adding a Property Editor to the Canvas" on page 74
Moving a property editor	"Moving a Property Editor within a Form" on page 75
Deleting a property editor	"Deleting a Property Editor from a Form" on page 75

Adding a Property Editor to the Canvas

Before you can add a property editor to the Form Builder Canvas, you need to have a table cell to place it in. For information on using tables and cells, see "Using Tables in the Form Builder" on page 70.

To add a property editor to a form

1 In the Form Builder, click the **Design** tab in the lower left corner to bring it to the front.

2 In the Palette, click the appropriate Drawer to expose the property editor you want to add.

3 Click the property editor and, on the Canvas, click an empty table cell.

An instance of the property editor appears in the table cell.

4 With the property editor instance selected on the Canvas, modify properties as needed in the **Properties** view.

  Properties vary with individual property editors.

5 Click the **Preview** tab to view how the form will appear on a portal page.

6 On the **File** menu, click **Save**.

Moving a Property Editor within a Form

To move a property editor within a form, use the following procedure.

Moving a property editor within a form

1 In the Form Builder, click the **Design** tab in the lower left corner to bring it to the front.

2 Drag the property editor to a table cell elsewhere on the form.

  You can drag to a cell within the table or in another table on the form. Multiple property editors can exist in the same cell.

3 On the **File** menu, click **Save**.

Deleting a Property Editor from a Form

To delete one or more property editors from a form, use the following procedure.

Deleting a property editor from a form

1 In the Form Builder, click the **Design** tab in the lower left corner to bring it to the front.

2 Select one or more property editor elements to be deleted, as described in "Selecting Cells in a Table" on page 73.

3 Right-click one of the selected cells and, from the menu, click **Delete**.

4 On the **File** menu, click **Save**.

You can also delete a property editor by selecting it and then pressing DELETE.

Using Existing Properties and Property Groups

If you have existing properties or property groups in the current project (portlet), you can drag them into the Form Builder.

When you drag a portlet property to a table cell in the Form Builder, a property label and a property value appear in the table cell.

When you drag a property group to a table cell in the Form Builder, a new table is created, having a row for each property in the property group. In each row, the first column contains a property label and the second column contains a property value.

webMethods.

CHAPTER 5

Creating Portlet-Level Help

What is Portlet-Level Help?

One important characteristic of portlets is that they are portable and self-contained. Portlets can be moved from one portal page to another and easily exported from one Portal server to another. For this reason, any help information associated with a portlet must also be portable and self-contained. The portlet must carry with it the main help file and any secondary files, graphics files, or cascading style sheets (CSS) that it needs. For the purposes of this guide, we refer to this as the *portlet help package*.

Hypertext links among HTML pages in the portlet help package remain valid as long as all of the files maintain the same relationship. Links from within the portlet help package to external pages cannot be considered reliable, nor can links from external pages to pages inside the portlet help package.

Information in the portlet help package cannot be accessed by the Portal server search portlet. Elements of the portlet help package are not stored in the Portal database.

To use the portlet-level help for a portlet, use the following actions:

- To view the help for a portlet, click the question mark-shaped Help icon on the portlet title bar. The Help page opens inside the portlet.

  If the Help icon is not available, there is no portlet-level help for the portlet.
- If the portlet does not cover the entire page, you can use the Maximize icon in the portlet title bar. Conversely, use the Minimize icon to return the portlet to its normal size.
- To return from the help page, click the Content icon in the portlet title bar.

If the portlet title bars are hidden, portlet-level help is not available unless you have created a link to the help files, as described in "Adding a Help Link to a Portlet" on page 82.

Preparing the Help Files

You can have multiple help pages with hypertext links among them. Use these guidelines to ensure a robust portlet help package:

- If you have multiple help pages associated with a portlet, create a main help page with links to the other pages.
- Place all files in the portlet help package in a single directory, or in subdirectories within it.
- Include all graphic and CSS files within the structure you create for the help package.

- JavaScript is acceptable as long as there are no dependencies outside the structure of the portlet help package.
- Do not make hypertext links help pages for other portlets. These links may not be reliable. You can, however make links to external Web sites.

To associate the portlet help package with the portlet

1. If you have not already done so, use the Portlet Developer to create the portlet and build a structure for it in the Eclipse workspace.
2. In the Eclipse workspace, create a directory with the name `help`, as shown in the following path:

   eclipse\workspace\portlet_name\ui\help
3. Copy all of the files in your portlet help package into the help directory.
4. Modify the helpPage property for the portlet, as described in "Linking to Help Pages" next in this chapter.
5. Build the portlet, as described in "Building the Portlet Deployment Package" on page 34.
6. Deploy the portlet, as described in "Deploying the Portlet to the Portal Server" on page 34.

Linking to Help Pages

When you create a new portlet the Portlet Developer, the portlet contains a helpPage property you can use to identify the URL for the main portlet help page. There are three types of URL you can call:

This type of URL	Is used to do this...
Portlet relative	Open the main page in the portlet help package deployed with the portlet.
Server relative	Open a page deployed elsewhere on the Portal server. This type of URL is only suitable for files you are certain will be available on the Portal server, and which will not have access denied to a particular user or group. You should never use a server-relative URL to call pages for another portlet, because that portlet may not be deployed on the server or may have access denied for a particular user or group.
External URL	Open a page on an external Web server.

Use the following procedure to set a link to the main help page for all instances of a portlet. If you want to specify a different link for a specific instance of a portlet, see "Help for Individual Instances of a Portlet" on page 81.

To set a help link in the helpPage property

1. In the **wmPortal Components** view of the Eclipse IDE, expand the portlet to display its components, including the **Property Groups** component.
2. Expand the **Property Groups** component.
3. Expand the **properties** property group and right-click the **helpPage** property; on the menu, click **Modify Property**.
4. In the Modify Portlet Property dialog box, leave the following fields untouched:

This field...	Should be left like this...
Internal Name	This field contains the internal name `helpPage`, which is the required value. Using any other name will cause the URL to fail.
Scope	By default, this field is set to **Value Shared by All Portlet Instances**. Do not change this value unless you want to use separate help pages for instances of the portlet, as described in "Help for Individual Instances of a Portlet" on page 81.

5. In **Default Value** field, do one of the following:

   - For a portlet-relative URL, type *directory/help_file*, where:

This element...	Is this...
directory	The name of the directory you have created within the portlet's ui directory. In "Preparing the Help Files" on page 78 we suggested that you name this directory `help`. **Note:** Do *not* precede the directory name with a slash (/).
help_file	The name of the main file that will be called by the portlet's help icon.

- For a server-relative URL, type /*server_path*/*help_file*, where:

This element...	Is this...
server_path	The path within the Portal server to the target file, starting within the portal.war directory. For example, if the target file is *webMethods_Install_Dir*/Portal/server/default/deploy/portal.war/help/glossary/index.html, the *server_path* value is: /help/glossary **Note:** You must precede this entry with a slash (/).
help_file	The name of the file that will be called by the portlet's help icon.

- For an external URL type the complete entry, including http://.

If the Default Value field is empty, the Portal server ignores the helpPage property and does not display the Help icon.

6 Click **Finish**.

Help for Individual Instances of a Portlet

When you follow the procedure described in "Linking to Help Pages" on page 79 the same portlet help package is used for all instances of the portlet on a Portal server. It is possible to alter the portlet-level help so that each instance of a portlet can call a different page. You designate the URL for the help page after the portlet is placed on a Portal page.

To specify a different help page for each instance of a portlet

1 Place the various alternate help pages into the directory you have created in "Preparing the Help Files" on page 78.

2 Create a helpPage property for the portlet, as described in "Linking to Help Pages" on page 79, with the following exceptions.

   a In the **Scope** list, select **Value Stored Per Portlet Instance**.

   b In the **Default Value** field, type a default URL to be called.

3 Build the portlet in Portlet Developer and deploy the portlet to the Portal server.

4 Add the portlet to a portal page, as described in "Adding a Portlet to a Portal Page" on page 113.

5 At the right edge of the title bar for the portlet you want to modify, click (Popup Menu) and then click **Properties**.

6 In the **Help Page** field, type the URL for the alternate page using the guidelines described in "Linking to Help Pages" on page 79.

7 At the bottom of the page, click **Apply**.

Adding a Help Link to a Portlet

With the default implementation of portlet-level help, you view the help page by clicking the Help icon in the portlet's title bar. If portlet title bars are hidden, there is no way to view portlet-level help unless you create a link within the body of the portlet. This procedure requires that you add code to the PCA Layout files (JSP pages) for the portlet.

Even if the portlet has multiple pages, all portlet-level help links in the portlet go to the same help page. You can however, use a main help page to provide links to other topics.

To create a link to portlet-level help from within the body of the portlet

1 In the **wmPortal Components** view of the Eclipse IDE, expand a portlet to display its components, including the **PCA Layouts** component.

2 Expand the **PCA Layouts** component, right-click the PCA layout you want to edit, and on the menu, click **Edit PCA Layout**.

   The JSP page opens in the text editor. Another way to open the JSP for editing is to double-click the PCA layout.

3 Add the code needed to create the help link.

   Place the code at your discretion anywhere between the <portlet:bean> and </portlet:bean> tags.

   Use one of these code fragments:

   To create a link:

```
<a href="<portlet:controller layout="helpLayout" context="href"/>">Help</a>
```

   To create a Help button:

```
<input type="button" value="Help" onclick="<portlet:controller
layout="helpLayout" context="href"/>"/>
```

4 To save changes, on the **File** menu, click **Save**.

webMethods.

CHAPTER 6

Examples of Portlet Development

Developing a Wizard Portlet

A wizard portlet has one or more layouts that the user can step through sequentially. At the end of the sequence, some final action is typically performed, based on the data collected on all pages. When you create a new wizard portlet, it contains a new property group for each page, a new PCA layout for each page, and a set of PCA methods for use in navigating among the pages.

To create a wizard portlet, see "New Wizard Portlet" on page 36. At the time you create the portlet, you specify the number of pages it should have.

Customizing the Finish Logic of a Wizard Portlet

The default wizard portlet navigation logic is to step through all layouts sequentially, starting from the first layout. The default business logic is to do nothing when the user clicks **Finish**.

By default the finish() method calls the **Finish** command defined in the portlet. You can either implement the **Finish** command or implement the finish() portlet method. The main difference between using the command and implementing the logic directly in the portlet is that the command is automatically available for other clients by means of different protocols, like Web services, HTTP GET and POST, Java RMI, and the Portlet Java API. If the method is implemented directly in the portlet bean, it is not easily accessible from outside the portlet.

Implementing Finish Logic in the Portlet Bean

To implement the logic for the Finish method in a portlet bean, use the following procedure.

To implement the logic for the Finish method in the portlet bean

1. In the **wmPortal Components** view of the Eclipse IDE, expand the wizard portlet to display its components, including the **PCA Methods** component.
2. Expand the **PCA Methods** component, right-click the finish layout, and on the menu, click **Edit PCA Method**.

   The portlet bean opens in the text editor at the location of the finish method. Another way to open the method for editing is to double-click the finish PCA method.
3. Add any Java code needed to customize the finish method.
4. To save changes, on the **File** menu, click **Save**.

All properties you put on the wizard layouts are available by means of get methods on the portlet bean. Use these methods to get the property values and call the Java API of your choice.

Implementing Finish Logic in the Finish Command

To implement the business logic for the Finish method in the Finish command, use the following procedure.

To implement the business logic for the Finish method in the Finish command

1 In the **wmPortal Components** view of the Eclipse IDE, expand the wizard portlet to display its components, including the **Mechanics** component.

2 Right-click the **Mechanics** component; on the menu, click **Edit Mechanics**.

   Mechanics are helper method classes that perform operations on publicly accessible services and the objects returned by services.

   The portlet mechanics implementation class opens in the text editor. Another way to open the class for editing is to double-click the **Mechanics** component.

3 Add any Java code needed to implement business logic.

4 To save changes, on the **File** menu, click **Save**.

Modifying Command Parameters

The default implementation of the command and mechanics does not take any custom parameters. If you need one or more parameters, you have to modify the command definition.

To modify the Finish command definition

1 In the **wmPortal Components** view of the Eclipse IDE, expand the wizard portlet to display its components, including the **BizPolicy** component.

   A bizpolicy (business policy) is protocol-independent business logic that leverages mechanics (method classes) and metadata services (control the life cycle and storage of data related to the Portal server) to perform its function.

2 Expand the **BizPolicy** component, right-click the **finish** command, and on the menu, click **Edit Command**.

   The file finish.java opens in the text editor.

3 Add any Java code needed to implement business logic.

4 To save changes, on the **File** menu, click **Save**.

If you name your parameters exactly the same as the names of the portlet properties to be used as the source data for the command, you can use the executeCommand("finish") method to invoke the command with automatic mapping of the command parameters to the portlet properties, as shown here:

```
public void finish() throws PortalException {
    // implicit mapping and conversion of command parameters from
    // portlet properties
    executeCommand("finish");
}
```

If the command parameters have different names or types than the portlet properties, use the following code example for the portlet bean finish() method to convert the portlet properties to the command parameters and invoke the biz policy. The class name takes the form *portlet_name*BizPolicy:

```
public void finish() throws PortalException {
    IBizPolicyManager provider =
(IBizPolicyManager)PortalSystem.getBizPolicyProvider();
    portlet_nameBizPolicy policy =
(portlet_nameBizPolicy)provider.getBizPolicy(getPortletThingID());
    // a command parameter that differs from the portlet property by
    // name or/ and type
    Integer i = new Integer(getProperty1());
    // strongly typed biz policy method call that in its turn calls the
    // finish command
    return policy.finish(getPortletContext(), getPortletThingID(), i);
}
```

Implementing Branching Logic for Layout Navigation

If the wizard portlet needs to implement a branching logic for the layout navigation, you have to implement the onNext() method in the portlet bean class.

To implement a branching logic in the wizard portlet

1. In the **wmPortal Components** view of the Eclipse IDE, expand the wizard portlet to display its components, including the **Portlet Bean** component.
2. Right-click the **Portlet Bean** component; on the menu, click **Edit Portlet Bean**.

   The portlet bean class opens in the text editor.
3. Scroll to the onNext method implementation and make changes to it.
4. To save changes, on the **File** menu, click **Save**.

In the following example, the onNext() method implements this logic:

1. The first wizard page shows an HTML form with an input field for the property1 property.
2. The user types some data into the input field for property1 and clicks **Next**.
3. The next() method of the portlet bean is called by the portlet controller, which calls the onNext() method to get the layout name of the next page to show.
4. The onNext() method checks the current layout name; if it is the first page and the value of the property1 is `some value`, the method returns `page3`. Otherwise, the method calls the default implementation, returning the next wizard layout which is, by default, `page2`.

```
protected String onNext() throws PortalException {
    if ("page1".equals(getLayout()) && "some
value".equals(getProperty1())) {
        return "page3";
    }

    return super.onNext();
}
```

The next example shows how you can use the onNext() method to set the portlet properties programmatically between the page transitions.

```
protected String onNext() throws PortalException {
    if ("page1".equals(getLayout()) &&
        "some value".equals(getProperty1())) {
        setProperty2(getProperty1() + " something");
        return "page3";
    }

    return super.onNext();
}
```

Developing a Tabs Portlet

A new tabs portlet has one or more independent layouts. At the top of each layout there is a tab control that displays one tab for each portlet layout. Clicking the tab makes the corresponding layout active. The new tabs portlet contains a new property group for each page and a new PCA layout for each page.

To create a tabs portlet, see "New Tabs Portlet" on page 37. At the time you create the portlet, you specify the number of pages your tabs portlet is to contain.

Customizing the UI of a Tabs Portlet

The default UI for a tabs portlet page is based on the portlet property groups in the same way as for Wizard portlet layouts. The main difference between the two portlet types is in the layout JSPs.

```
<%-- this draws tabs, one tab for each layout --%>
<portlet:portlet uri='portlet.tabs'>
    <util:param name='portlet' value='<%=portlet.getPortletThingID()%>'/>
    <util:param name='titles' value='{caption}'/>
    <util:param name='values' value='{name}'/>
    <util:param name='links'><portlet:url layout='{name}'
        clearState='portlet'/></util:param>
    <util:param name='list' value='<%=
        com.webmethods.rtl.util.WildcardMatcher.filterCollection(
                                portlet.getControllerInfo().getLayouts(),
                                "name",     /* match layout name */
                                "*",        /* include pattern */
                                "result_*") /* exclude pattern */
    %>'/>
    <util:param name='view' value='tabs'/>
    <util:param name='selection' value='<%=portlet.getLayout()%>'/>
    <util:param name='selectionLinked' value='false'/>
</portlet:portlet>
```

The default HTML form on each tabs portlet layout posts back to the same layout, updating all the portlet properties displayed on this form. Use the same technique to customize the form UI as for the wizard portlets, as described in "Developing a Wizard Portlet" on page 84.

Controlling Tab Range

By default, all layouts are available as tabs in the tabs control. You can customize the tabs range using the subList(int,int) API to specify a slice of the filtered layout list. For example, suppose you have a seven-tab layout and want only the third through seventh tabs to be displayed when the third tab is active. To do so, you change the layout for the third tab as follows:

```
    <util:param name='list' value='<%=
        com.webmethods.rtl.util.WildcardMatcher.filterCollection(
                                portlet.getControllerInfo().getLayouts(),
                                "name",/* match layout name */
                                "*",/* include pattern */
                                "result_*").subList(3,7) /* exclude pattern */
    %>'/>
```

Filtering PCA Layouts

The `include pattern` and `exclude pattern` parameters are used to filter which PCA layouts are displayed as tabs in the UI. Each of these parameters can contain a comma-separated list of filter conditions. The value of each of the filter conditions can be either the full name of a PCA layout or a partial PCA layout name with a single wildcard character (a `*` character) somewhere in the string. Consider the following code fragment.

```
    <util:param name='list' value='<%=
        com.webmethods.rtl.util.WildcardMatcher.filterCollection(
                                portlet.getControllerInfo().getLayouts(),
                                "name",     /* match layout name */
                                "form*,test*one,*two,job", /* include pattern */
                                "result_*") /* exclude pattern */
    %>'/>
```

The `include pattern` and `exclude pattern` parameters match any PCA layout whose name matches any of these filters:

- Layout name starts with `form`
- Layout name starts with `test` and ends with `one`
- Layout name ends with `two`
- Layout name equals `job`

When the Portal server determines which pages to display as tabs, it uses this logic for each PCA Layout in the portlet:

1. Each of the filters in the `exclude pattern` list is checked to see if it matches the name of the PCA layout. If there is a match, the PCA layout is not rendered as a tab.
2. Each of the filters in the `include pattern` list is checked to see if it matches the name of the PCA layout. If there is a match, the PCA layout is rendered as a tab. If there is no match, the PCA layout is ignored and is not rendered as a tab.

Customizing the Logic of a Tabs Portlet

To perform an operation when the user clicks **Submit** on a Tabs portlet layout, you need to add a new portlet method, as described in "Creating a PCA Method" on page 52.

Then you have to change the HTML form generated by the portlet:controller tag to invoke the portlet method you have created instead of posting back to the same portlet layout. The following code demonstrates how to post the data from the HTML form based on the `page1` property group to a portlet method called `myMethod` and then go to the layout `page2`.

```
<portlet:controller context="form" method="myMethod" layout="page2" >
    <portlet:propertyGroups excludeGroups="*" includeGroups="page1" />

    <ui:propertySubmit>
        <util:param name="property_submit">  Apply  </util:param>
        <util:param name="property_submit_cancel_fn"><portlet:controller method="return"
            context="js" /></util:param>
    </ui:propertySubmit>
</portlet:controller>
```

Converting an Existing JSP Application to a Portlet

You may currently have a variety of small JSP applications deployed to your corporate intranet. To embed these JSP applications directly into the portal and take full advantage of the webMethods Portal toolkit, you need to convert them to portlets.

JSP applications are typically composed of some UI, Java beans, and some business logic. These elements map to portlet components as follows:

Concept	JSP version	Portal version
User Interface	JSP pages	Portal layouts
Session State	Java beans	Portlet beans
Business Logic	Java code or EJBs	Portlet methods

When converting a JSP application to a portlet, use the Portlet Developer to assist you in creating each part of the portlet.

For purposes of this example, we will convert the class number guessing game that ships with the Apache Tomcat servlet container. In this game, a random number is chosen between 1 and 100, and the user attempts to guess the number.

Creating the Portlet Properties

The first step when converting a JSP application to a portlet is to identify the data that makes up the state of the application. In the case of the number guessing game, there are three pieces of information that need to be tracked:

- The secret number the user is trying to guess
- The total number of guesses the user has made
- The current guess the user has made

Two of the properties are controlled by the application and require no user input, the last property (the current guess) requires user input. We will create two portlet property groups, one group for the two internal properties and one group for the property that requires the user input. The last property is separated from the others because the Portlet Controller Tag Libraries automatically generate HTML forms that display the property group in an HTML form, posting the form data back to the portlet and invoking the optionally specified portlet method.

To create the portlet properties

1. In the **wmPortal Components** view of the Eclipse IDE, create a generic portlet, as described in "Creating a New Project" on page 29.
2. Create the first property group, as described in "Creating a Property Group" on page 43.

   Give this property group, which will hold the two internal properties, an internal name by which you can refer to it programmatically, such as internalNumbers.
3. In the first property group, create two properties, as described in "Creating a Property" on page 44; set the **Scope** attribute for each property to **Value Stored per Session**.

   Give these properties the internal names targetNumber and numGuesses. Setting the scope to **Value Stored per Session** causes the value of each property to be stored between page requests during the current user's session.
4. Create the second property group, as described in "Creating a Property Group" on page 43.

   The names you give this property group have the following uses:

This name...	Is used for this...
Internal Name	Provides a name by which you can refer to it programmatically.
Display Name	Provides the display name of the form.

5. In the second property group, create one property having the internal name currentGuess, as described in "Creating a Property" on page 44; set the **Scope** attribute for this property to **Value Stored per Session**.

   The names you give this property have the following uses:

This name...	Is used for this...
Internal Name	Provides a name by which you can refer to it programmatically.
Display Name	Provides the display name of the input field.

Creating the Portlet Methods

Converting the JSP application also requires moving the associated business logic of the JSP application into the portlet. Most of the business logic is invoked from calls made by the JSP page, but some logic is invoked as part of HTML form processing. Those methods are registered as portal methods, and will later be automatically invoked by the Portlet Controller Tags.

We will create two portlet methods for this portlet. The first portlet method resets the state of the portlet for the current user. This method is invoked at the end of the game. The second portlet method automatically increments the number of guesses every time the user submits a guess.

To create the portlet methods

1 In the **wmPortal Components** view of the Eclipse IDE, expand the portlet to display its components, including the **PCA Methods** component.

2 Create the first method, as described in "Creating a PCA Method" on page 52, setting attributes as follows:

For this attribute...	Do this...
Method Name	Type a valid function name, such as reset.
Display Name	Type a name that can be the same as the method name.
Result Layout Name	Leave as it is. There is only one JSP page in this example.
Error Layout Name	Leave as it is. There is only one JSP page in this example.
Port Controller State	In the list, choose **Clear controller state after the method executes**. We want to clear session state.

3 Expand the **PCA Methods** component, right-click the reset method, and on the menu, click **Edit PCA Method**.

The portlet bean opens in the text editor at the location of the reset method. Another way to open the PCA method for editing is to double-click the PCA method.

4 In the text editor, replace the `//TODO` comment with code for the reset method, as shown by the bold text in the following code fragment:

```
/**
 * this method is called by portlet controller dispatch handler
 */
public void reset() throws PortalException {
    release();
}
```

5 On the **File** menu, click **Save**.

6 Create the second method, as described in "Creating a PCA Method" on page 52, setting attributes as follows:

For this attribute...	Do this...
Method Name	Type a valid function name, such as incrementGuesses.
Display Name	Type a name that can be the same as the method name.
Result Layout Name	Leave as it is. There is only one JSP page in this example.
Error Layout Name	Leave as it is. There is only one JSP page in this example.
Port Controller State	In the list, choose **Leave controller state unchanged**. We want to preserve the session state.

7 Expand the **PCA Methods** component, right-click the incrementGuesses method, and on the menu, click **Edit PCA Method**.

The portlet bean opens in the text editor at the location of the incrementGuesses method. Another way to open the PCA method for editing is to double-click the PCA method.

8 In the text editor, replace the `//TODO` comment with code for the incrementGuesses method, as shown by the bold text in the following code fragment:

```
/**
 * this method is called by portlet controller dispatch handler
 */
public void incrementGuesses() throws PortalException {
    int numGuesses = getNumberOfGuesses() + 1;
    setNumGuesses(String.valueOf(numGuesses));
}
```

9 To save changes, on the **File** menu, click **Save**.

Additional Java code is needed to support the UI features provided in the layout, along with some internal helper code. The following code sample shows the complete portlet bean with bold text to indicate code you need to add.

```
/**
 ** $Workfile: Guess.java $
 ** Copyright (c) 2001-2003 webMethods, Inc. All Rights Reserved.
 */
package com.webmethods.portal.portlet.wm_guess;

import com.webmethods.portal.PortalException;
```

```
/**
 * Guess portlet implementation.
 *
 * @version    $Revision: 1.0 $
 * @author     <a href="mailto:!authorEmail!">!authorName!</a>
 */
public class Guess extends com.webmethods.portal.framework.portlet.beans.JspPortletBean
{


    public final static String PROPERTY_TARGETNUMBER = "targetNumber";
    public final static String PROPERTY_NUMGUESSES = "numGuesses";
    public final static String PROPERTY_CURRENTGUESS = "currentGuess";

    /**
     * Release any local variables so the object can be added back into the
     *  object pool.
     */
    public void release() {
        super.release();
    }

    /**
     * Gets the value of the targetNumber property.
     *
     * @return the value of the targetNumber property
     */
    public String getTargetNumber() {
        return getPropertyAsString(PROPERTY_TARGETNUMBER);
    }


    /**
     * Sets the value of the targetNumber property.
     *
     * @param value the value to use for the targetNumber property
     */
    public void setTargetNumber(String value) {
        setProperty(PROPERTY_TARGETNUMBER, value);
    }


    /**
     * Gets the value of the numGuesses property.
     *
     * @return the value of the numGuesses property
     */
    public String getNumGuesses() {
        return getPropertyAsString(PROPERTY_NUMGUESSES);
    }
    /**
     * Sets the value of the numGuesses property.
     *
     * @param value the value to use for the numGuesses property
     */
    public void setNumGuesses(String value) {
        setProperty(PROPERTY_NUMGUESSES, value);
    }


    /**
     * Gets the value of the currentGuess property.
     *
     * @return the value of the currentGuess property
     */
    public String getCurrentGuess() {
        return getPropertyAsString(PROPERTY_CURRENTGUESS);
    }


    /**
     * Sets the value of the currentGuess property.
     *
     * @param value the value to use for the currentGuess property
     */
    public void setCurrentGuess(String value) {
        setProperty(PROPERTY_CURRENTGUESS, value);
    }


    /**
     * this method is called by portlet controller dispatch handler
     */
    public void reset() throws PortalException {
        release();
    }

    /**
    * this method is called by portlet controller dispatch handler
     */
    public void incrementGuesses() throws PortalException {
        int numGuesses = getNumberOfGuesses() + 1;
        setNumGuesses(String.valueOf(numGuesses));
    }
}
```

```
    /**
     * Method invoked by the Portal Layout to get the text of the next hint
     */
    public String getHint() throws PortalException {
        return calcCurrentGuess() > calcCurrentNumber() ? "lower" : "higher";
    }


    /**
     * Method invoked by the Portal Layout to get the current number of guesses
     */
    public int getNumberOfGuesses() {
        return getPropertyAsInt(PROPERTY_NUMGUESSES, 0);
    }


    /**
    * Method invoked to determine if the user has correctly guessed the secret number
    */
    public boolean isSuccess() throws PortalException {
    return calcCurrentNumber() == calcCurrentGuess();
    }


    /**
     * Method invoked by the Portal Layout to determine if this is a new game
     */
    public boolean isNewGame() {
        return getNumberOfGuesses() == 0;
    }


    /**
     * The following are simple internal helper functions
     */
    protected int calcCurrentNumber() throws PortalException {
        int targetNumber = getPropertyAsInt(PROPERTY_TARGETNUMBER, 0);
        if (targetNumber == 0) {
            Random random = new Random(System.currentTimeMillis());
            targetNumber = Math.abs(random.nextInt()) % 99;
            targetNumber += 1;
            getControllerBean().setProperty(PROPERTY_TARGETNUMBER,
                String.valueOf(targetNumber));
        }
        return targetNumber;
    }


    protected int calcCurrentGuess() {
        return getPropertyAsInt(PROPERTY_CURRENTGUESS, 0);
    }
}
```

Creating the Portlet UI

The final step to converting a JSP application to a portlet is migrating the UI. In the number guessing game, there are three different screens to show the user, based upon the state of the game. Each of these screens could be broken up into different layouts, but they are simple enough that we will place them on one portal layout and use some simple if-then Tag Libraries to switch among them. The three states of this game are:

- The user has guessed the correct number
- The user is playing a new game
- The user has made an incorrect guess

For this example, rather than creating a new layout, we will modify the existing layout. We will, however, use a variety of Web-based UI techniques including:

- Displaying PortletBean property values
- Using if-then statements
- Creating Portlet Controller Buttons that invoke portal methods
- Creating Forms that display a portlet property group and automatically invoke a portlet method.

To modify the existing layout, you need to edit the JSP page, which in this case is default.jsp.

To add UI features to the layout

1. In the **wmPortal Components** view of the Eclipse IDE, expand the portlet to display its components, including the **PCA Layouts** component.
2. Expand the **PCA Layouts** component, right-click the **default** layout, and on the menu, click **Edit PCA Layout**.

   The JSP page opens in the text editor. Another way to open the JSP for editing is to double-click the PCA layout.
3. Remove all of the lines in default.jsp between the `<portlet:bean>` and `</portlet:bean>` lines.

   Between these lines, you will add your own layout.
4. Create a scriptlet that invokes a method on the portlet:

```
<b>Number of guesses: <%=portlet.getNumberOfGuesses()%></b>
```

5 Create an if-then-else clause to present the three different displays:

```
<util:if expr="<%=portlet.isNewGame()%>">
   <util:then>
      <p>Welcome to the Number Guess game.</p>
   </util:then>
  <util:else>
      <p>Good guess, but nope.  Try <b><%=portlet.getHint()%></b>.
         You have made <%=portlet.getNumberOfGuesses()%> guesses.</p>
   </util:else>
</util:if>
```

6 Create a normal HTML input button, add the Portlet Controller tag to the OnClick attribute, and configure it to invoke a portlet method:

```
<input type="button" value="Play Again"
   onClick="<portlet:controller context="js" method="reset"/>" />
```

7 Create a form that displays a a portlet property group and automatically invokes a portlet method:

   a Create a Portlet Controller tag and set its method attribute.

   b Inside the Portlet Controller tag, create a portlet property groups tag that displays just the property group you want to have in your HTML form.

   c Create a Property Submit tag that draws the custom Apply and Cancel buttons.

```
<portlet:controller name="gameForm" context="form" layout="default" method="incrementGuesses">
   <portlet:propertyGroups form="gameForm" excludeGroups="*" includeGroups="gameProps" />

   <%-- these are Apply (submit) and Cancel buttons.  --%>
   <ui:propertySubmit>
      <util:param name="property_submit"> Guess </util:param>
      <util:param name="property_submit_fn">validateForm</util:param>
      <util:param name="property_submit_cancel_fn"><portlet:controller method="return"
         context="js" /></util:param>
      <util:param name="property_submit_form" value="gameForm" />
   </ui:propertySubmit>
</portlet:controller>
```

8 Make other additions as needed to complete the page layout.

9 To save changes, on the **File** menu, click **Save**.

The completed Portal UI page is shown below:

```
<%@include file="/ui/system/taglibs.inc" %>
<%@include file="/ui/system/beans.inc" %>
<%@page import="com.webmethods.portal.portlet.wm_numberguess.*"%>
```

```
<portlet:bean id="portlet" className="com.webmethods.portal.portlet.wm_numberguess.NumberGuess" >

   <b>Number of guesses: <%=portlet.getNumberOfGuesses()%></b>


   <util:if expr="<%=portlet.isSuccess()%>">
      <%-- Success --%>
      <util:then>
         <p>Congratulations!  You got it. And after just <%=portlet.getNumberOfGuesses()%>
            tries.</p>
         <input type="button" value="Play Again" onClick="<portlet:controller context="js"
            method="reset"/>" />
      </util:then>

      <util:else>
         <%-- Play the game --%>
         <%
            int numGuesses = portlet.getNumberOfGuesses();
            boolean newGame = portlet.isNewGame();
         %>

         <util:if expr="<%=newGame%>">
            <util:then>
               <p>Welcome to the Number Guess game.</p>
            </util:then>

            <util:else>
               <p>Good guess, but nope.  Try <b><%=portlet.getHint()%></b>.
                  You have made <%=portlet.getNumberOfGuesses()%> guesses.</p>
            </util:else>
         </util:if>


         <p>I'm thinking of a number between 1 and 100.</p>
         <portlet:controller name="gameForm" context="form" layout="default"
method="incrementGuesses">


            <%-- this draws property group "gameProps". --%>
      <portlet:propertyGroups form="gameForm" excludeGroups="*" includeGroups="gameProps" />


         <%-- these are Apply (submit) and Cancel buttons. TODO: customize/remove the buttons --%>
            <ui:propertySubmit>
               <util:param name="property_submit"> Guess </util:param>
               <util:param name="property_submit_fn">validateForm</util:param>
               <util:param name="property_submit_cancel_fn"><portlet:controller method="return"
                  context="js" /></util:param>
               <util:param name="property_submit_form" value="gameForm" />
            </ui:propertySubmit>


         </portlet:controller>

      </util:else>
   </util:if>

</portlet:bean>
```

Developing Portal Java Applications

webMethods Portal contains an Application Programming Interface (API) that allows a Java developer to control nearly every aspect of the portal. This includes such operations as publishing, deleting, updating, subscribing, setting permissions, and many other types of activities. Collectively, these high-level APIs are known as *Portal Verbs*.

Portal Verbs are obtained from the BizPolicyManager, which manages registration and access to all of the Portal Verbs. To obtain a reference to a Portal Verb, you need an IContext object, which is a Java bean that represents a portal session. An IContext object is easy to obtain and, as a Java developer, you can decide whether you want to use the current user's IContext, or use a System IContext. The main difference between the two is that if you use a System IContext, no access checks are performed, but if you use the current user's IContext, all operations occur in the context of the user, and only the operations the user has access to execute are allowed to operate.

Once you obtain a reference to the Portal Verb that you want to invoke, you can call its Strongly-Typed API. The following sections show examples of invoking some Portal Verbs.

Publishing to the Portal Programmatically

The following examples include publishing a new folder, publishing a new link, and publishing a new document.

Publishing a New Folder

To programmatically publish a new folder, you need to consider a few requirements. First, you need to decide where you are going to publish the new folder. This implies that you can obtain a valid URI to the parent of the folder. Often, this is most easily accomplished by using aliases to provide common names for a folder. Next, you need to consider what name you are going to give the new folder. If you are programmatically generating folders, you need a way to ensure that your folder's name will be unique if all the new folders are published to the same folder.

To publish a new folder

1. Acquire an IContext from the ContextFactory.

   In the following example, the context is acquired on behalf of the current user.

2. Acquire the URI of the parent to which you want to publish.

3. Obtain a reference to the IContainerPolicy, which allows you to invoke the createChild Portal Verb.

4 Fill in the map to specify the appropriate properties of the new folder.

```
//Get the context of the current user  (if you're running in an asychronous agent
//mode, then this will return a System IContext)
IContext context = ContextFactory.acquireContext(true);

//Get the parent folder to publish to.  'folder.public' is the alias for the Public Folders
IThingID parentID = (IThingID) PortalSystem.getPortalSystem().acquireURI("folder.public");

//Get the ContainerPolicy that exposes the createChild Portal Verb
IBizPolicyManager bizPolicyManager = (IBizPolicyManager) PortalSystem.getBizPolicyProvider();
IContainerPolicy containerPolicy = (IContainerPolicy) bizPolicyManager.getBizPolicy(parentID);

//Set the properties of the new folder
Map props = new HashMap();
props.put("xtype", "folder"); //This is the type of object to create
props.put("name", "New Folder"); //This is the name of the new folder
props.put("description", "Folder Description"); //This is the folder's description

//Create the new folder
IThingID newThingID = containerPolicy.createChild(context, parentID, props);
```

Publishing a New Link

Publishing a new link is similar to publishing a new folder. The main difference is that a link is a different portal object type. A link takes the URL parameter, which is the URL to which the link points.

To publish a new link

1 Acquire an IContext from the ContextFactory.

In the following example, the context is acquired on behalf of the current user.

2 Acquire the URI of the parent to which you want to publish.

3 Obtain a reference to the IContainerPolicy, which allows you to invoke the createChild Portal Verb.

4 Fill in the map to specify the appropriate properties of the new link.

```
//Get the context of the current user  (if you're running in an asychronous agent mode,
//this will return a System IContext)
IContext context = ContextFactory.acquireContext(true);

//Get the parent folder to publish to.  'folder.public' is the alias for the Public Folders
IThingID parentID = (IThingID) PortalSystem.getPortalSystem().acquireURI("folder.public");

//Get the ContainerPolicy that exposes the createChild Portal Verb
IBizPolicyManager bizPolicyManager = (IBizPolicyManager) PortalSystem.getBizPolicyProvider();
IContainerPolicy containerPolicy = (IContainerPolicy) bizPolicyManager.getBizPolicy(parentID);

//Set the properties of the new link
Map props = new HashMap();
props.put("xtype", "link"); //This is the type of object to create
props.put("name", "New Link"); //This is the name of the new link
props.put("description", "Link Description"); //This is the links's description
props.put("URL", "http://www.webmethods.com"); //This is the Web address that the link points to

//Create the new link
IThingID newThingID = containerPolicy.createChild(context, parentID, props);
```

Publishing a New Document

Publishing a new document is only slightly more complex than the previous examples. The complexity comes from generating the actual content of the document. In the following example, we generate the text of the document at run time as a simple XML document.

To publish a new document

1 Acquire an IContext from the ContextFactory.

In the following example, the context is acquired on behalf of the current user.

2 Acquire the URI of the parent to which you want to publish.

3 Obtain a reference to the IContainerPolicy, which allows you to invoke the createChild Portal Verb.

4 Fill in the map to specify the appropriate properties of the new document.

Subscribing to Portal Events

```
//Get the context of the current user  (if you're running in an asychronous agent mode,
//this will return a System IContext)
IContext context = ContextFactory.acquireContext(true);

//Get the parent folder to publish to.  'folder.public' is the alias for the Public Folders
IThingID parentID = (IThingID) PortalSystem.getPortalSystem().acquireURI("folder.public");

//Get the ContainerPolicy that exposes the createChild Portal Verb
IBizPolicyManager bizPolicyManager = (IBizPolicyManager) PortalSystem.getBizPolicyProvider();
IContainerPolicy containerPolicy = (IContainerPolicy) bizPolicyManager.getBizPolicy(parentID);

//Create the document text
   String myXMLDoc = "<xml>Some Random Content</xml>";
   InputStream inputStream = new ByteArrayInputStream(myXMLDoc.getBytes());
   PortalFileBean portalFileBean = new PortalFileBean("newDoc", inputStream, "Doc.xml", "text/xml", 
false);

   //set the properties of the new document
   Map props = new HashMap();
   props.put("xtype", "content"); //This is the type of object to create
   props.put("name", "New Content"); //This is the name of the new content
   props.put("description", "Content Description"); //This is the content's description
   props.put("newDoc", portalFileBean); //The portalFileBean represents the content to be published

   //create the new document
   IThingID newThingID = containerPolicy.createChild(context, parentID, props);
```

To programmatically subscribe to a publish event, you typically would publish a portlet that subscribes to specific types of portal events and implements any custom event processing logic. After creating the portlet, you modify the portlet to implement two interfaces:

- IInitializable—Allows you to set up your subscriptions at run time
- ICreateEvent.ISynchronousListener—Allows you to synchronously handle any ICreateChild (publish) events that are generated

Note: There is also an Asychronous variant that is not described here.

In this example, we modify an existing portlet to subscribe to publish events, but only execute custom code when a newly published portal item has been published to a specific folder.

To subscribe to publish events

1. Implement the IInitializable method in your portlet.

   This method allows you to subscribe to the ICreateEvent at startup and unsubscribe at shutdown. These methods are also invoked at portlet deployment and undeployment time.

2. Implement the ICreateEvent.ISynchronousListener, which allows your portlet to handle portal events for newly created items.

3. In the onSynchronousEvent method, examine the event to find out if the new item is in the specific folder in which you are interested.

4. Execute the custom logic. In this case, write an entry to the log file.

```
//This Portlet implements the IInitializable and the ICreateEvent.ISynchronousListener interfaces
public class SamplePub extends JspPortletBean implements IInitializable,
ICreateEvent.ISynchronousListener {

   //Subscribe to the ICreateEvents
   public void init(IComponentData data, String phase) throws InitializationException {
      ICreateEvent.SynchronousListeners.add(this);
   }

   //Unsubscribe to the ICreateEvents
   public void shutdown() {
      ICreateEvent.SynchronousListeners.remove(this);
   }

   //An ICreateEvent has been generated
   public void onSynchronousEvent(ICreateEvent ev) {
      //Print out a debug statement
      Debug.debug("onEvent: " + ev.getCreatedID(), m_logCategory);

      //Is this the specified folder?
      IThingID publicFolderID = null;
      try {
         IContext context = ContextFactory.acquireContext(true);
         publicFolderID = (IThingID) PortalSystem.getPortalSystem().acquireURI("folder.public");
      catch (PortalException e) {
         Debug.warn(e, m_logCategory);
         return;
      }
      //Compare the new Item's parent with our specified folder, and if it matches,
      //Print out another Debug statement
      IThingID containerID = ev.getContainerID();
      if (containerID.equals(publicFolderID)) {
         Debug.debug("onEvent: " + ev.getCreatedID() + " was created in the public Folder",
            m_logCategory);
      }
   }

   ...
```

webMethods

CHAPTER 7

Developing Portal-Based Web Sites and Web Applications

Overview of webMethods Portal

webMethods Portal aggregates information from a variety of sources and serves that information as Web pages. A static Web site served by a Web server like Apache or Internet Information Services (IIS) usually has three kinds of items: folders, documents, and executables. Executables, such as .cgi scripts, enable a Web server to serve dynamic content or process form data from a Web application. The Portal server can also contain these items, but executable-like presentation and logic are encapsulated as portlets. Instead of serving a simple directory listing when a client requests a folder, the Portal server can serve a sophisticated portal page that combines the contents of the folder into a dynamic Web page. By wiring two or more portlets together on a portal page, you can even create a whole new Web application (see "Wiring Portlets" on page 116.

You can create new or simpler names for items in the portal, configure the Portal server to recognize those names when they appear in a URL, and automatically redirect a user to the named item. In the Portal server, a simple name that points to a portal item is called an alias. To manage aliases for portlets, see "Managing Portlet Aliases" on page 115. To manage aliases for a portal page, see the *webMethods Portal Administrator's Guide*.

All Web pages the portal serves are generated dynamically. You can configure the Portal server to serve the exact same content to different users using a different look and feel for each user. You can also configure the Portal server to serve certain areas of the portal or specific portal pages with differing looks and feels. The look and feel of a portal page is called a skin. See Chapter 8, "Working with Skins in webMethods Portal" for more information about skins.

You can configure the Portal server to serve the same portal page content to different users using different headers, footers, or side-navigations for each user. You can also configure the Portal server to serve certain areas of the portal or specific portal pages with differing headers, footers, or side-navigations. The specific combination of a header, footer, and side-navigation is called a shell. See Chapter 9, "Working with Shells in webMethods Portal" for more information about shells.

Creating and Modifying Portal Pages

In webMethods Portal, the terms *portal page* and *folder* are synonyms. By default, when a client requests a folder, the Portal server combines all the contents of the folder into a dynamic portal page.

A portal page is made up of rows, column, and portlets. On a portal page, the Portal server displays any items that are not actually portlets, such as documents, links, or sub-folders, in a way that makes them look and behave like portlets. You can move any portlet on a portal page into any row or column on that page, and you can create any number of rows and columns. You can also order portlets from top to bottom within a column. By default, a new portal page has one row containing two columns.

The following sections describe basic ways of creating and modifying a portal page:

This activity...	Is described here...
Create a new portal page.	"Creating a Portal Page" on page 109
Add, reposition, or remove rows in a portal page.	"Modifying Rows in a Portal Page" on page 109
Add, reposition, or remove columns in a portal page.	"Modifying Columns in Portal Pages" on page 111

Creating a Portal Page

To create a new portal page, use the following procedure.

To create a new portal page

1. Log in to webMethods Portal.
2. In the **Home Folder**, click **My Folders**.
3. In the upper right-hand corner of **My Folders**, click the menu icon.
4. On the menu, click **New** and **Portal Page**.
5. In the **Name** field, type the name of the new portal page.
6. (Optional) In the **Description** field, type a description of the new portal page.
7. Click **Create**.
8. In **My Folders**, click the new portal page you have created.

   A blank portal page appears in your browser window.

Modifying Rows in a Portal Page

On a portal page, you can add a row, reposition a row, or remove a row.

Adding a Row to Portal a Page

To add a row to a portal page, use the following procedure.

To add a row to a portal page

1. In the upper right-hand corner of the portal page, click the menu icon.
2. On the menu, click **Edit Portal Page**.
3. In the **Root** list of the **Available Portlets** panel, click **Page Features**.
4. In the **Page Features** list of the **Available Portlets** panel, drag the **New Row** portlet and drop it onto the portal page at the location where you want to add the row.

   A red box appears beneath the cursor location whenever the cursor is over a portal page row location, indicating where the row would be positioned if you released the mouse button.
5. Drag a new row portlet for each row you want to add to the page.
6. On the left side of the page control area, click **Save**.

   A blank row appears on your portal page.

Repositioning a Row on a Portal Page

To reposition a row on a portal page, use the following procedure.

To reposition a row on a portal page

1. In the upper right-hand corner of the portal page, click the menu icon.
2. On the menu, click **Edit Portal Page**.
3. Hold the mouse down over the title bar of the row you want to reposition.
4. Drag the row and drop it onto the new location on the portal page.

   A red box appears beneath the cursor location whenever the cursor is over a portal page row location, indicating where the row would be positioned if you released the mouse button.
5. On the left side of the page control area, click **Save**.

Removing a Row from a Portal Page

To remove a row from a portal page, use the following procedure.

To remove a row from a portal page

1. In the upper right-hand corner of the portal page, click the menu icon.
2. On the menu, click **Edit Portal Page**.
3. On the title bar of the row you want to delete, click the delete icon(**X**).

Note: Any portlets in the deleted row are permanently deleted from the portal page.

4. On the left side of the page control area, click **Save**.

Modifying Columns in Portal Pages

On a portal page, columns appear within rows. You can add a column, reposition a column, or remove a column.

Adding Columns to a Portal Page

To add a column to a portal page, use the following procedure.

To add a column to a portal page

1. In the upper right-hand corner of the portal page, click the menu icon.
2. On the menu, click **Edit Portal Page**.
3. In the **Root** list of the **Available Portlets** panel, click **Page Features**.
4. In the **Page Features** list of the **Available Portlets** panel, drag the **New Column** portlet and drop it onto the portal page in the row where you want to add the column.

   A red box appears beneath the cursor location whenever the cursor is over a portal page column location, indicating where the column would be positioned if you released the mouse button.
5. Drag a new column portlet for each column you want to add to the page.
6. On the left side of the page control area, click **Save**.

   A blank column appears on your portal page.

Repositioning a Column on a Portal Page

To reposition a column on a portal page, use the following procedure.

To reposition a column on a portal page

1 In the upper right-hand corner of the portal page, click the menu icon.

2 On the menu, click **Edit Portal Page**.

3 Hold the mouse down over the title bar of the column you want to reposition.

4 Drag the column and drop it onto the new location on the portal page

A red box appears beneath the cursor location whenever the cursor is over a portal page column location, indicating where the column would be positioned if you released the mouse button.

5 On the left side of the page control area, click **Save**.

Removing a Column from a Portal Page

To remove a column from a portal page, use the following procedure.

To remove a column from a portal page

1 In the upper right-hand corner of the portal page, click the menu icon.

2 On the menu, click **Edit Portal Page**.

3 On the title bar of the column you want to delete, click the delete icon(**X**).

Note: Any portlets in the deleted column are permanently deleted from the portal page

4 On the left side of the page control area, click **Save**.

Adding and Managing Portlets

Portlets provide the functionality for a portal page. There are several tasks you can perform on portlets from within webMethods Portal:

These tasks...	Are described here...
Add, reposition, or remove a portlet.	"Manipulating Portlets on Portal Pages" on page 113
Modify the settings of a portlet.	"Modifying the Settings of an Existing Portlet" on page 115
Add and remove portlet aliases.	"Managing Portlet Aliases" on page 115
Make connections between portlets.	"Wiring Portlets" on page 116

Manipulating Portlets on Portal Pages

On a portal page, portlets appear within columns. You can add a portlet, reposition a portlet, or remove a portlet.

Adding a Portlet to a Portal Page

To add a portlet to a portal page, use the following procedure.

To add a portlet to a portal page

1. In the upper right-hand corner of the portal page, click the menu icon.
2. On the menu, click **Edit Portal Page**.
3. In the **Root** list of the **Available Portlets** panel, click **Portlets**.
4. In the **Portlets** list of the **Available Portlets** panel, click the category that contains the portlet you want to add to the portal page.
5. In the appropriate category list of the **Page Editor**, drag the portlet and drop it onto the portal page in the column where you want it to reside.

   A red box appears beneath the cursor location whenever the cursor is over a portal page column, indicating where the portlet would be positioned if you released the mouse button.
6. On the left side of the page control area, click **Save**.

   The portlet appears on your portal page.

Repositioning a Portlet in a Portal Page

To reposition a portlet on a portal page, use the following procedure.

To reposition a portlet on a portal page

1. In the upper right-hand corner of the portal page, click the menu icon.
2. On the menu, click **Edit Portal Page**.
3. Hold the mouse down over the title bar of the portlet you want to reposition.
4. Drag the portlet and drop it onto the new location on the portal page

   A red box appears beneath the cursor location whenever the cursor is over a portal page column, indicating where the portlet would be positioned if you released the mouse button.
5. On the left side of the page control area, click **Save**.

Tip! If you are the portal administrator or have administrator privileges, you can move a portlet on the page without editing the portal page. Place the cursor over the Move icon on the portlet's title bar and drag the portlet where you want to move it.

Removing a Portlet from a Portal Page

To remove a portlet from a portal page, use the following procedure.

To remove a portlet from a portal page

1. In the upper right-hand corner of the portal page, click the menu icon.
2. On the menu, click **Edit Portal Page**.
3. On the title bar of the portlet you want to delete, click the delete icon(**X**).

   Note: The portlet is permanently deleted from the portal page.
4. On the left side of the page control area, click **Save**.

Modifying the Settings of an Existing Portlet

You can configure the settings of a portlet independently from settings of the portal page in which it resides. Some of the settings are properties defined during portlet development.

To modify the properties of a portlet

1 At the right edge of the title bar for the portlet you want to modify, click 🗏 (Popup Menu) and then click **Properties**.

2 In the Properties page make modifications as appropriate:

Make changes here...	If you want to...
General	Change the name or description of the portlet.
Properties	Change or set properties designed into the portlet at the time it was developed. For example, a portlet dealing with locations might require a postal code.
Display	Change how the portlet is displayed with respect to the page.
Maintenance	Add or remove aliases by which the portlet can be called by other sites, as described in "Managing Portlet Aliases" on page 115.

3 At the bottom of the page, click **Apply**.

Tip! If you are editing a portal page, you can also set portlet properties from the page control area at the bottom of the page. In edit mode, click a portlet to select it. In the page control area, click the **Properties** tab.

Managing Portlet Aliases

Just as you can assign aliases to be used in accessing a portal page from another Web site or Web application, you can assign one or more aliases to individual portlets.

Adding an Alias to a Portlet

To add an alias to a portlet, use the following procedure.

To add an alias to a portlet

1. At the right edge of the title bar for the portlet you want to modify, click ▤ (Popup Menu) and then click **Properties**.
2. In the **Aliases** section of the Properties page, click **Add**.
3. In the **Script Prompt** field, type the portlet alias to be added.
4. At the bottom of the page, click **Apply**.

Removing an Alias from a Portlet

To remove an alias from a portlet, use the following procedure.

To remove an alias from a portlet

1. At the right edge of the title bar for the portlet you want to modify, click ▤ (Popup Menu) and then click **Properties**.

   The Properties page of the portlet is displayed.
2. In the **Aliases** section of the Properties page, select the alias to be removed and click **Remove**.
3. At the bottom of the page, click **Apply**.

Wiring Portlets

You can connect, or *wire*, any property of any portlet on a portal page to any property of any other portlet on the same page. When you wire one property to another, whenever the page is rendered, the Portal server automatically sets the value of the destination property to the value of the source property. This feature allows you to quickly create a composite application out of several different portlets.

For example, if you create a page with two portlets, one a search form and one a search results display, you can wire the search form value (the source value) to the results display input value (the destination value). When a user enters some information into the search form and submits it, the Portal server updates the results display to the results of that search.

To wire one portlet to another

1. Decide which portlet is the wiring source and which is the wiring destination.

   The destination portlet property receives its property value from the source portlet property.

2. At the right edge of the title bar for the destination portlet, click (Popup Menu) and then click **Wiring**.

   The wiring dialog page displays a list of properties on the destination portlet that are available for wiring to other portlets.

3. Decide which destination portlet property to wire from among properties listed on the left-hand side of the wiring dialog.
4. In the **Portlet** list for the property to be wired, select the source portlet.
5. In the **Property** list for the property to be wired, select a property from among those available on the source portlet and click **Submit**.

   The run-time view of this page should display the destination portlet with whatever values are configured in the source portlets.

Tip! If you are editing a portal page, you can also perform wiring from the page control area at the bottom of the page. In edit mode, click a portlet to select it. In the page control area, click the **Wiring** tab.

Global Wiring

Users and groups have a set of Principal Attributes that you can wire to a portlet. For example, suppose a portlet uses a postal code to display certain information when a user views a portal page. If the postal code is provided by wiring from a Principal Attribute Provider, when the postal code attribute is modified within a directory service, the portlet uses the modified attribute value.

To wire a Principal Attribute to a portlet

1. At the right edge of the title bar for the portlet, click (Popup Menu) and then click **Wiring**.
2. In the **Portlet** list, choose **Other**.

   A new window opens.

3. In the **Location** list, choose **Root**.

4 Click **Global Wiring Data**.

Depending on what Principal Attributes exist on the Portal server, you can see one or more of the following:

Principal Attribute Provider	Description
Core Attributes Wiring	A set of attributes valid for all users of the portal.
User Profile Wiring	A set of user attributes maintained by the portal administrator.
Ldap Attributes Wiring	A set of attributes exposed from external directory services.

5 Click the ➪ **Select** icon for the Principal Attribute set to move it to the **Selected** panel, and click **Select**.

6 Click **Browse**.

A new window opens, containing attributes belonging to the selected Principal Attribute Providers.

7 From the list, select the attribute you want to wire to the portlet and click **Select**.

The portlet is now wired to use the attribute value belonging to the user who views the portal page on which it appears.

For information about Principal Attribute Providers, see the *webMethods Portal Administrator's Guide*.

Controlling the Layout of the Portal Page

You can control the layout of the portal page by modifying properties of rows, columns, and portlets as you place them on the page. You can use these properties on regular portal pages and on the Header, Leftnav, Rightnav, and Footer shell sections. Some layout properties are ignored when the page is displayed as a portal page and some are ignored when the page is used as a shell section.

To control portal page layout by modifying row, column, and portlet properties

1 In the upper right-hand corner of the portal page, click the menu icon.

2 On the menu, click **Edit Portal Page**.

3 In edit mode, click the title bar for the row, column, or portlet for which you want to change properties.

4 In the page control area at the bottom of the window, make changes as needed.

5 On the left side of the page control area, click **Save**.

In edit mode, if you have a row, column, or portlet selected, the following properties appear in the **Layout** panel of the page control area at the bottom of the page:

This property...	Applies to...	Affects...	And does this...
Titlebar	Portal pages	Portlets	Specifies whether or not to use the Titlebar shell section at the top of the portlet. Choices are **Yes** or **No**.
Border	Portal pages	Portlets	Specifies whether or not to place a border around the portlet. Choices are **Yes** or **No**.
Attributes	Portal pages and shell sections	Rows, columns	Reserved for future use.
Width	Portal pages and shell sections	Columns	Specifies a width for the column. Type a percentage, such as `40%`. Actual widths may vary, based on the number and size of columns in a row, and may also be affected by the **Word wrap** property.
Height	Portal pages and shell sections	Rows, columns	Specifies a height for the row or column. Accepts units that are valid for HTML and CSS, such as px or in.
Word wrap	Portal pages and shell sections	Columns	Specifies whether or not to use word wrap to enforce column width. Choices are **Yes** or **No**. If you choose **No**, the column will never be narrower than the width of the text contained on any line within it.
Horz align	Portal pages and shell sections	Columns	Specifies how items in the column are aligned horizontally. Choices are **Left**, **Center**, and **Right**.
Vert align	Portal pages and shell sections	Columns	Specifies how items in the column are aligned vertically. Choices are **Top**, **Middle**, and **Bottom**.

This property...	Applies to...	Affects...	And does this...
CSS class	Shell sections	Rows, columns, portlets	Applies a CSS class to the row, column or portlet. Type the name of the class, omitting the leading period (use `nav` for the `.nav` class). The Portal server uses the CSS file for the skin associated with the portal page. The location of the CSS file is: *webMethods_Install_Dir*/Portal/server/ default/deploy/portal.war/ui/skins/ *skin_name*/css/base.css
CSS style	Shell sections	Rows, columns, portlets	Applies a style to the row, column, or portlet. Type any style that is valid for use in a CSS file. For example, if you type `border: 1pt dashed red`, a dashed red line appears as a border.
Background image	Shell sections	Rows, columns, portlets	Applies a background from the skin associated with the portal page. Type the name of the image file. The file is assumed to be in this location: *webMethods_Install_Dir*/Portal/ser ver/default/ deploy/portal.war/ui/skins/*skin_n ame*/images/ The background is like a regular HTML background image in that it does not affect the size of the content area. You can use standard CSS background properties in the **CSS style** field to control how the background is displayed: whether it repeats, whether it is centered or is offset in some direction, and whether it scrolls with the page or is fixed.

Creating Links for Single Sign-On

Single sign-on is the ability for a user to log into one application and then use other applications without having to log into each one separately. webMethods Portal supports single sign-on through the Security Assertion Markup Language (SAML), an XML-based framework for the exchange of security information.

To take advantage of single sign-on, a user must be known on both the source Portal server and the target entity. In most cases, common knowledge of a user is provided by use of the same directory service. For more information on configuring a Portal server to be used as a target for single sign-on, see the *webMethods Portal Administrator's Guide*.

On any portal page, you can add a link to a SAML target entity, such as a Portal server. If the target accepts SAML assertions from the source Portal server, when a known user clicks the link, no login credentials are required. If the target entity does not accept SAML assertions from the source Portal server, or if the user is not known on the target entity, login credentials may be required.

Under the SAML specification, an intermediary called an artifact receiver can perform authentication on behalf of the target Web application. In such a case, the SAML source requires two URLs: one for the Artifact Receiver and one for the target Web application.

You can place one or more SAML links on any portal page you have permission to edit.

To create a SAML link on a source portal page

1. In the upper right-hand corner of the portal page, click the menu icon.
2. On the menu, click **Edit Portal Page**.
3. In the **Root** list of the **Available Portlets** panel, click **Links**.
4. In the **Links** list of the **Available Portlets** panel, drag the **Single Sign-on Link** portlet and drop it onto the portal page at the location where you want to add the link.

   A red box appears beneath the cursor location whenever the cursor is over a valid portal page location, indicating where the portlet would be positioned if you released the mouse button.
5. On the left side of the page control area, click **Save**.
6. At the right edge of the title bar for the single sign-on portlet, click 🗏 (Popup Menu) and then click **Properties**.
7. In the Properties page make modifications as appropriate:

Make changes here...	If you want to...
Name	Replace `Single Sign-on Link` with the text that is to go with the link.
SAML Authentication URL	Type the URL for a resource on the target computer. The target can be any portal page on a Portal server. If you are connecting to a Web application through a SAML Artifact Receiver, use this field for the Artifact Receiver URL.

Make changes here...	If you want to...
Use POST or GET	Determines the method used to pass data to the target computer. POST — Passes data to a gateway program's STDIN. POST, the default, is the preferred method for single sign-on data. GET — Passes data as a string appended to the URL after a question mark.
artifactParameterName	If this is a SAML connection with another Portal server or other webMethods component, do not change the default value SAMLart. If this is a SAML connection to a third-party source, type the artifact parameter name used by the third-party application.
Application Target URL	If you have typed the URL for a SAML Artifact Receiver in the **SAML Authentication URL** field, type the URL for a Web application. Otherwise, leave this field empty.

8 At the bottom of the page, click **Apply**.

webMethods

CHAPTER 8

Working with Skins in webMethods Portal

What are Skins?

The look and feel of a portal page is encapsulated in a skin. A skin is composed of a particular set of colors, fonts, and images. You can associate a skin with a particular user or group, in which they will view the contents of the portal using the skin. You can associate a skin with a particular folder hierarchy; all users who view portal pages within that hierarchy view them using the skin.

You can perform the following types of tasks in managing skins:

These tasks...	Are described here...
Create or delete a skin.	"Managing Skins" on page 124
Replace images in a skin with images from your local drive, from an existing skin, or from a Web site.	"Replacing Images in a Skin" on page 125
Replace colors in a skin using a color picker, colors from an existing skin, or colors from a Web site.	"Managing Colors in a Skin" on page 127
Replace font families in a skin, including the use of font families taken from a Web site.	"Managing Font Families in a Skin" on page 129
Preview changes made to a skin using portal pages not in the My Folders, Home Page, or Public Folders folders.	"Previewing a Portal Page Elsewhere on the Portal Server" on page 131

Managing Skins

You can create and delete skins with the Skin Administration portlet. To use the Skin Administration portlet, you must either have the portal administrator grant you permission to access the portlet, or you must log in as the portal administrator.

Creating a New Skin

As a portal administrator, you can create a skin with the Skin Administration portlet.

To create a new skin

1 As the portal administrator, browse to the **Administration Dashboard** and in the **Portal User Interface** folder, click **Skin Administration**.

2 On the sub-tab bar of the **Skin Administration** page, click **Create New Skin**.

3 In the **System Name** field of the **Create New Skin** page, type a short name that contains only letters, numbers, and the underscore character.

   This name is used internally by the Portal server.

4 In the **Display Name** field, type the skin title that you want users to see.

   This name has no character restrictions.

5 From the **Parent Skin** list, choose the skin from which the new skin will inherit any unspecified properties.

   The system default skin is selected by default.

6 Click **Create**.

A new skin initially inherits all of its properties (colors, fonts, and images) from its parent. You can modify the new skin, changing just a single property, such as adding a new header logo, or you can modify the skin to create a radical new look and feel with completely different colors, fonts, and images.

Deleting a Skin

As a portal administrator, you can delete a skin with the Skin Administration portlet.

To delete a skin

1 As the portal administrator, browse to the **Administration Dashboard** and in the **Portal User Interface** folder, click **Skin Administration**.

2 At the right edge of the row for the skin you want to delete, Click (Popup Menu) and on the menu, click **Delete**.

Replacing Images in a Skin

The skin of a portal page contains images, such as logos, that help shape the appearance and structure of the page. Using the Skin Administration portlet, you can replace images in a skin with images from your local drive, images from an existing skin, or images from a Web site.

To replace an image in a skin

1 As the portal administrator, browse to the **Administration Dashboard** and in the **Portal User Interface** folder, click **Skin Administration**.

2 At the right edge of the row for the skin you want to modify, click (Popup Menu) and then click **Edit**.

3 On the sub-tab bar of the General Properties page, click **Images**.

4 In the **Skin Properties** list on the left side of the page, select the image you want to replace.

The image is highlighted with a red box.

5 Depending on the source of the image, do one of the following:

If the image source is...	Do this...
On the local drive	Take the following steps, in order: ■ In the **Picker** panel on the upper-right side of the page, click **Browse**. ■ In the dialog box, navigate to the new image, select it, and click **Open**. ■ With the location of the new image displayed in the **Picker** panel, click **Upload**. ■ Click the arrow directly to the left of the **Picker** panel.
Part of an existing skin	Take the following steps, in order: ■ On the lower-right side of the page, from the **-- Select palette --** list, select the skin from which you want to copy the image. ■ Scroll through the Palette and select the new image. ■ Click the arrow directly to the left of the Palette.
On a Web site	Take the following steps, in order: ■ On the lower-right side of the page, from the **-- Select palette --** list, select **URL**. ■ In the Script Prompt, type the Web site URL and click **OK**. ■ Scroll through the Palette and select the new image. ■ Click the arrow directly to the left of the Palette.

6 At the bottom of the page, under the **Preview** heading, click **Preview**.

The preview demonstrates how your changes have affected the skin.

Tip! To preview a portal page other than My Folders, My Home Page, or Public Folders, see "Previewing a Portal Page Elsewhere on the Portal Server" on page 131.

7 Close the Preview window.

8 Click **Save**.

Managing Colors in a Skin

The skin of a portal page uses colors to define the look and feel of the page. Using the Skin Administration portlet, you can replace color settings in a skin using colors created in a color picker, colors from an existing skin, or colors taken from a Web site.

Replacing Colors Using a Color Picker

There are a number of different color settings in a skin that affect different parts of a portal page. To change settings, you select colors in the skin editor, apply the colors to skin properties, and preview your changes.

To replace the color of a skin property using a color picker

1 As the portal administrator, browse to the **Administration Dashboard** and in the **Portal User Interface** folder, click **Skin Administration**.

2 At the right edge of the row for the skin you want to modify, click (Popup Menu) and click **Edit**.

3 On the sub-tab bar of the General Properties page, click **Colors**.

4 In the **Skin Properties** list on the left side of the page, select the skin property for which you want to replace the color.

The skin property is highlighted with a red box.

5 In the **Picker** panel on the top-right side of the page, click the color to be used as a replacement.

The selected color appears in the horizontal bar at the bottom of the **Picker** panel, and the hexadecimal value appears in the **#** field at the top.

Tip! If you know the hexadecimal value of the replacement color, you can type it directly in the **#** field.

6 To save a color to the **Scratchpad** panel, click the down arrow directly beneath the **Picker** panel.

Tip! If you want lighter and darker shades of a particular color for use in foregrounds and backgrounds, save the color in the **Scratchpad** panel and, in the vertical bar on the right edge of the **Picker** panel, click above or below the original color. Save multiple colors to the **Scratchpad**.

7 To set the color for a skin property, do either of the following:

- Click the arrow directly to the left of the **Picker** panel.
- Select a color in the **Scratchpad** panel and click the left arrow for that panel.

8 At the bottom of the page, under the **Preview** heading, click **Preview**.

The preview demonstrates how your changes have affected the skin.

Tip! If the portal page is not in one of the folders labeled My Folders, My Home Page, or Public Folders, see "Previewing a Portal Page Elsewhere on the Portal Server" on page 131.

9 Close the Preview window.

10 Click **Save**.

Replacing Colors from a Skin or Web Site

You can replace the color of a skin property using color values from an existing skin or from a Web site.

To replace the color of a skin property using a color from a skin or a Web site

1 As the portal administrator, browse to the **Administration Dashboard** and in the **Portal User Interface** folder, click **Skin Administration**.

2 At the right edge of the row for the skin you want to modify, click (Popup Menu) and then click **Edit**.

3 On the sub-tab bar of the General Properties page, click **Colors**.

4 In the **Skin Properties** list on the left side of the page, select the skin property for which you want to replace the color.

The skin property is highlighted with a red box.

5 Depending on the source of the color, do one of the following:

If the color source is...	Do this...
Part of an existing skin	Take the following steps, in order: ■ On the lower-right side of the page, from the **-- Select palette --** list, select the skin from which you want to copy the color. ■ Scroll through the Palette and select the new color. ■ Click the arrow directly to the left of the Palette.
On a Web site	Take the following steps, in order: ■ On the lower-right side of the page, from the **-- Select palette --** list, select **URL**. ■ In the Script Prompt, type the Web site URL and click **OK**. ■ Scroll through the Palette and select the new color. ■ Click the arrow directly to the left of the Palette.

6 At the bottom of the page, under the **Preview** heading, click **Preview**.

The preview demonstrates how your changes have affected the skin.

Tip! If the portal page is not in one of the folders labeled My Folders, My Home Page, or Public Folders, see "Previewing a Portal Page Elsewhere on the Portal Server" on page 131.

7 Close the Preview window.

8 Click **Save**.

Managing Font Families in a Skin

In controlling the font families used by a skin, you have the choice of designing the style yourself or selecting a set of font families used by another Web site. Rather than attempting an exhaustive description of the capabilities available to you in the selection of font families, the following procedure presents a scenario describing how one might go about replacing font families based on the families used in another Web site.

Note: Font properties used by individual skins can vary. The properties described in the following sample procedure may not be available in all skins.

To replace the font families used by a skin with the font families used by another Web site

1. As the portal administrator, browse to the **Administration Dashboard** and in the **Portal User Interface** folder, click **Skin Administration**.
2. At the right edge of the row for the skin you want to modify, click 🗏 (Popup Menu) and then click **Edit**.
3. On the sub-tab bar of the General Properties page, click **Fonts**.
4. On the lower-right side of the page, from the **-- Select palette --** list, select **URL**.
5. In the Script Prompt, type the Web site URL and click **OK**.

   The Palette displays a list of font families derived from the Web site.
6. In the Palette on the lower-right side of the page, find a line that seems to represent the normal body text of the Web site, and select it.

   This action should result in several font families being listed in the **Picker** panel on the top-right side of the page. If not, try some other lines in the palette that look like normal body text.
7. Click the arrow directly above the Palette.

   This action saves the font information to the **Scratchpad** panel for later use.
8. In the line you just created in **Scratchpad** panel, select the text in the edit field and rename it to something meaningful, such as **my regular text**.
9. In the **Skin Properties** list on the left side of the page, select the **regular** skin property.

   The **regular** skin property is highlighted with a red box, meaning it is selected for editing.
10. Click the left arrow directly to the left of the **Scratchpad** panel.

    This action sets the **regular** skin property with the value selected in the **Scratchpad** panel.
11. In the **Skin Properties** list on the left side of the page, select the **bold** skin property.

    The **bold** skin property is highlighted with a red box.
12. In the **Scratchpad** panel, select **my regular text**.

    While the **my regular text** scratchpad item was already the active item in the **Scratchpad** panel (with a dark red border around it), selecting it again makes it the active item for the page (with a bright red border around it), and loads the **my regular text** font information back into the **Picker** panel.
13. In the **Font-style** area of the **Picker** panel, select the **Bold** check box.

14 Click the arrow directly to the left of the **Picker** panel.

This action sets the **bold** skin property with the value in the **Picker** panel.

15 In the **Skin Properties** list, select the **small** skin property.

16 In the **Scratchpad** panel, select **my regular text**.

17 In the **Font-size** area of the **Picker** panel, do one of the following:

If this option is selected...	Do this...
Relative size	Change the value in the list to be a size smaller than the existing size
Numeric size	Edit the number in the field to be a value two or three smaller than the existing size

18 Click the arrow directly to the left of the **Picker** panel.

19 In the **Skin Properties** list, select the **medium** skin property.

20 In the **Scratchpad** panel, select **my regular text**.

21 In the **Preview** list at the bottom of the page, choose **Public folders** and then click **Preview**.

The preview demonstrates how your changes have affected the skin.

Tip! If the portal page is not in one of the folders labeled My Folders, My Home Page, or Public Folders, see "Previewing a Portal Page Elsewhere on the Portal Server" on page 131.

22 Close the preview window.

23 Click **Save**.

Previewing a Portal Page Elsewhere on the Portal Server

You can preview a portal page other than My Folders, My Home Page, or Public Folders in the skin editor.

To preview a portal page elsewhere on the Portal server

1 Make changes to skin properties within Skin Administration.

2 In the **Preview** list at the bottom of the page, choose **URL**.

3 Open a new browser window, navigate to the portal, and within the portal, navigate to the portal page you want to preview.

4 In the **Address** bar of the browser, select the URL and type CTRL+C to copy it.

5 Return to the Script Prompt dialog from the first browser and paste in the URL by typing CTRL+V.

6 Click **OK**.

7 Click **Preview**.

CHAPTER 9

Working with Shells in webMethods Portal

What are Shells?

The Portal server derives the content and layout of the header and footer of a portal page from the content and layout of the current shell's Header and Footer folders. When the Portal server renders one portal page, it is actually displaying the contents of up to five folders at once: the shell's Header, the shell's Leftnav, the requested portal page, the shell's Rightnav, and the shell's Footer. The Portal server displays the content of the requested portal page as individual portlets, but it renders the content of shell sections without title bars, borders, or additional spacing. You can apply Cascading Style Sheet (CSS) classes and styles to the rows and columns that compose the shell sections, as well as specify the exact dimensions of those rows and columns, to further customize the layout of the shell.

The Titlebar shell section applies a title bar to each portlet on the portal page, which includes the display name and buttons for controlling the portlet. You can hide the Titlebar shell section individually for each portlet.

You can create and modify a shell with the Shell Administration portlet. To use the Shell Administration portlet, you must either have the portal administrator grant you permission to access the portlet, or you must log in as the portal administrator.

Creating a New Shell

The first step in constructing a new shell is to create it using an existing shell as a parent. A new shell initially inherits all of its properties from its parent. These properties (or shell sections) are: Header, Footer, Leftnav, Rightnav, and Titlebar. You can replace any of these sections with a new, custom shell section.

To create a new shell

1. As the portal administrator, browse to the **Administration Dashboard** and in the **Portal User Interface** folder, click **Shell Administration**.
2. On the sub-tab bar of the Shell Administration page, click **Create New Shell**.
3. In the **Name** field of the Create New Shell page, type a name for the shell.

   This name has no character restrictions.
4. Optionally, in the **Description** field, type a description of the shell.

   The description appears in the list of shells on the Shell Administration page.
5. From the **Parent Shell** list, choose the shell from which the new shell will inherit any unspecified properties.

   The system default shell is selected by default.
6. Click **Create**.

Modifying a Shell

After you have created a new shell from a parent shell, you can modify individual sections of that shell independently to construct a new shell.

To modify a shell

1 As the portal administrator, browse to the **Administration Dashboard** and in the **Portal User Interface** folder, click **Shell Administration**.

2 At the right edge of the row for the shell you want to modify, click (Popup Menu) and then click **Edit**.

   The General Properties page of the shell is displayed.

3 If you want to change the display name for the shell, in the **Display Name** field, type a new name for the shell.

4 If you want to change the parent shell from which to take the various shell sections, in the **Parent Shell** list, choose the parent shell. The list contains shells that currently exist on the Portal server.

5 For each shell section, choose the parent or source for the section from among these choices:

Choose this option...	To do this...
Inherited	To use the shell section from the shell chosen in the **Parent Shell** field.
Portal Page	(Not available for the Titlebar shell section.) To use the content of an existing folder for the shell section, click **Browse**. In the left panel of the Picker window, navigate to the folder and click the **Select** icon; the selected folder appears in the **Selected Items** panel. Click **Select**.
Portlet	(Titlebar shell section only.) To use an existing portlet, click **Browse**. In the left panel of the Picker window, navigate to the portlet and click the **Select** icon; the selected portlet appears in the **Selected Items** panel. Click **Select**.

6 For shell sections that you want to edit, take one of the following actions:

To accomplish this...	Do this...
Edit a shell section inherited from another shell	With the **Inherited** option selected, click **Clone from Parent**. The Portal server creates a folder based on the inherited shell section.
Edit an existing folder used as a shell section	With the **Portal Page** option selected, make sure the name of the target folder is displayed.

Note: Within the Portal server, you cannot edit JSP pages or portlets. You need to use the Portlet Developer.

7 Click **Edit**.

The folder that represents the shell section opens in edit mode.

8 Modify the shell section just as you would any other folder, as described here:

- "Creating and Modifying Portal Pages" on page 108
- "Adding and Managing Portlets" on page 113

9 After editing the shell section folder, on the left side of the page control area, click **Save**.

10 At the bottom of the General Properties page, click **Save**.

Using an Alias with a Shell Section

If a folder has an alias, you can use the alias to select it for use as a shell section. For information about using aliases with folders, see the *webMethods Portal Administrator's Guide*.

To select a shell section using an alias

1 As the portal administrator, browse to the **Administration Dashboard** and in the **Portal User Interface** folder, click **Shell Administration**.

2 At the right edge of the row for the shell you want to modify, click 🗏 (Popup Menu) and then click **Edit**.

The General Properties page of the shell is displayed.

3 In the shell section you want to associate with an alias, click **Use Alias**.

4 In the **Alias Name** field of the portal resource selector, type the alias the alias of the folder you want to use for this shell section.

5 To determine if the Portal server can find the alias, click **Test**.

6 If the Portal server correctly resolves the alias, click **Select**.

7 If needed, you can clone this folder or edit it directly, as described in "Modifying a Shell" on page 135.

8 At the bottom of the General Properties page, click **Save**.

Deleting a Shell

After it is no longer needed, you can delete a shell.

To delete a shell

1 As the portal administrator, browse to the **Administration Dashboard** and in the **Portal User Interface** folder, click **Shell Administration**.

2 At the right edge of the row for the shell you want to delete, click ▤ (Popup Menu) and then click **Delete**.

Making an Empty Shell Section

A shell always has the four folders that make up the shell sections. You may, however, want a shell design in which one or more of the shell sections is empty and takes up no space in the display. A shell section is empty if it contains no portlets and has no formatting information associated with it. In the default shells provided with webMethods Portal, the Leftnav and Rightnav shell sections display as being empty.

You cannot edit the Titlebar shell section as you do the others. To hide the title bar, you need to set the **Titlebar** attribute for individual portlets to **No**. For more information, see "Controlling the Layout of the Portal Page" on page 118.

To make a shell section (other than a Titlebar) empty

1 As the portal administrator, browse to the **Administration Dashboard** and in the **Portal User Interface** folder, click **Shell Administration**.

2 At the right edge of the row for the shell you want to modify, click ▤ (Popup Menu) and then click **Edit**.

The General Properties page of the shell is displayed.

3 In the **Alias Name** field of the portal resource selector, type the alias the alias `shell.section.blank`.

4 To determine if the Portal server can find the alias, click **Test**.

5 If the Portal server correctly resolves the alias, click **Select**.

6 At the bottom of the General Properties page, click **Save**.

webMethods

CHAPTER 10

Developing Composite Applications

Some Examples of Composite Applications

webMethods Portal provides a number of tools and components you can use to develop portal-based composite applications. The following examples describe ways you can build and deploy composite applications without writing a line of code. After following these examples, you should be able to start developing your own composite applications using webMethods Portal.

The examples focus on the following portal component and technologies:

- "Using the HTML Text Portlet" on page 140
- "Using the Frame Portlet" on page 142
- "Using the Webclip Portlet" on page 142
- "Portlet Wiring" on page 146

Before you can begin following the examples, you need to create a portal page, as described in "Creating a Portal Page" on page 109.

Using the HTML Text Portlet

The HTML Text portlet is useful on portal pages where announcements, notices, and so forth need to be updated on a regular basis. Note that each time you change the content of the HTML Text portlet, the previous HTML you included is overwritten.

You can use the HTML Text portlet to quickly embed external HTML content in your portal. Keep in mind that when you are copying and pasting external HTML content from an external site, you are really taking a snapshot of the HTML from the external site. Any changes that are made to the external site after you copy and paste into the HTML text portlet will *not* be rendered inside of your portlet. If you need to have content updated whenever the external Web site is updated, see "Using the Webclip Portlet" on page 142.

There are two parts to using an HTML Text portlet. First you add the portlet to the portal page; then you pass content from the external Web site to the portlet.

Adding an HTML Text Portlet

To add an HTML Text portlet to your composite application, use the following procedure,

To add an HTML Text portlet to your composite application

1. In the upper right-hand corner of the portal page, click the menu icon, and on the menu, click **Edit Portal Page**.
2. In the **Root** list of the **Available Portlets** panel, click **Portlets**.

3 In the **Portlets** list of the **Available Portlets** panel, click **Drawing**.

4 In the **Drawing** list, drag the **HTML Text** portlet and drop it into one of the columns on your portal page.

5 On the left side of the page control area, click **Save**.

   A blank HTML Text portlet appears on your portal page.

6 Click the menu icon at the right edge of the HTML Text portlet, and on the menu, click **Properties**.

7 In the **Name** field of the Properties page, type the name of the new HTML Text portlet.

8 (Optional) In the **Description** field, type a description of the new HTML Text portlet.

9 Under the Text property, use the simple browser-based HTML Text Editor to add content.

   You have several options for adding HTML content to your portlet. You can use the WYSIWYG editing capabilities for simple HTML editing, or you can click **< >** to toggle between the WYSIWYG view and the HTML source view.

10 When you are finished adding HTML content, at the bottom of the page, click **Apply**.

   The run-time view of your new portal page is displayed in the browser window.

Passing Content to the HTML Text Portlet

To pass HTML content from an external Web site into the HTML Text portlet, use the following procedure.

To pass HTML content from an external Web site into the HTML Text portlet

1 Open another browser window and browse to a Web site.

   For example, you might want to include the Yahoo! Directory from the Yahoo! home page in your HTML Text portlet.

2 Highlight the text for the Yahoo! Directory in your browser window and press CTRL+C to copy the directory.

3 Go back to the browser with the **HTML Text Properties** page open.

4 Click on the **HTML Text** area and then type CTRL+V to paste the content into the HTML Text editing area.

5 Click **Apply**.

   The Yahoo! Directory content appears within the **HTML Text** portlet.

Using the Frame Portlet

The Frame portlet is useful when you need to embed an entire existing Web site or Web application inside a portlet. This is a quick and easy way of exposing existing Web site content inside the portal. You cannot use the Frame portlet if the target Web site requires authentication or an external user account. In such a case, use the Webclip portlet instead. For information about the Webclip portlet, see "Using the Webclip Portlet" on page 142.

Tip! To embed an entire Web site or Web application to your portal page, it is a good idea to place the Frame portlet in a separate row that spans the entire page.

To add a Frame portlet to your composite application (portal page)

1 Create a new row that spans the entire portal page, as described in "Adding a Row to Portal a Page" on page 110.

2 Return to the **Root** list of the **Available Portlets** panel, click **Portlets**, and then click **Drawing**.

3 In the **Drawing** list of the **Available Portlets** panel, drag the **Frame** portlet and drop it into the row you just added to your portal page.

4 On the left side of the page control area, click **Save**.

A blank Frame portlet appears on your portal page.

5 To configure the Frame portlet, click the menu icon for the Frame portlet, and on the menu, click **Properties**.

6 On the Properties page, fill in the following fields:

Field	Information
Name	A name for the Frame portlet.
Description	(Optional) A description of the portlet.
URL	The URL to the target Web site or Web application.
Frame Height	The height of the resulting frame. The default is 100 pixels.
Frame Name	A name to be used for portlet wiring.

7 Click **Apply**.

Using the Webclip Portlet

You can use a Webclip portlet to clip a section of an existing Web site or Web application. The Webclip portlet is significantly more sophisticated than the HTML Text and Frame portlets, and provides a more robust way of embedding selected portions of existing Web sites and Web applications inside the portal. The Webclip portlet also allows you to clip content from Web sites and applications that require user authentication.

Configuring the Webclip Portlet Without Authentication

If a Web site or Web application does not require a username and password, you can configure the Webclip portlet without authentication.

To configure a Webclip portlet without authentication

1. In the upper right-hand corner of the portal page, click the menu icon.
2. On the menu, click **Edit Portal Page**.
3. In the **Root** list of the **Available Portlets** panel, click **Portlets**.
4. In the **Portlets** list of the **Available Portlets** panel, click **Internet - Tools**.
5. In the **Internet - Tools** list, drag the **Webclip** portlet and drop it into one of the columns on your portal page.
6. On the left side of the page control area, click **Save**.
7. From the run-time view of your portal page, click 🗏 (Popup Menu) at the right edge of the Webclip row and then click **Properties**.
8. For the **URL** property on the Properties of Webclip page, click **Select**.
9. In the new browser window, scroll down to the **Location** property and, in the **URL** field, type the URL of the target Web site or Web application from which you want to clip Web content and click **Load**.

   The target page is loaded into the upper portion of the browser window.
10. Maximize the browser window and note that the Web content is rendered in the following way:

This content...	Is rendered this way...
Table-based content	Blue **table** heading bar
Div-based content	Red **div** heading bar
Form-based content	Free **form** heading bar

Note: The Webclip portlet implements a fully validating SAX parser to break up the target Web site into *clippable* portions that you can then select. In addition, the Webclip portlet identifies any malformed HTML and attempts to correct it before including the clipped content in the portlet container.

11 To select a TABLE, DIV, or FORM element from the browser window, click the corresponding **table**, **div**, or **form** heading bar.

   The content to be clipped by the portlet is highlighted with a yellow border.

12 Click **Select**.

13 Return to the portal browser window and configure additional properties as appropriate.

   a In the **Link Targets** list, define the behavior for any hyperlinks in the clipped content by selecting one of the following options:

      - Open links in the current window
      - Open links in new windows
      - Open links in a specific frame

   b In the **Javascript** list, define the behavior for any JavaScript in the clipped content by selecting one of the following options:

      - Do not block scripts
      - Block scripts
      - Proxy scripts

   c In the **CSS** list, define the behavior for any Cascading Style Sheet (CSS) definitions in the clipped content by selecting one of the following options:

      - Do not block CSS styles
      - Block CSS styles
      - Proxy CSS styles

   d In the **Images** list, define the behavior for any image content in the clipped content by selecting one of the following options:

      - Do not block images
      - Block images
      - Proxy images

   e In the **Cache Age** list, choose the length of time clipped content is cached on the Portal server.

      Content that is clipped by Webclip can be cached on the Portal server to improve performance.

14 In the **Name** field at the top of the Properties of Webclip page, type the new name for this clipped content.

15 In the **Description** field, type a description of the clipped content.

16 Click **Apply**.

The run-time view of your portal page now displays the clipped content.

Configuring Webclip with HTTP Authentication

The Webclip portlet provides HTTP authentication for passing user login and password credentials from the Portal server to the target Web site or Web application that is being clipped.

Note: The following procedure requires access to an existing Web site or Web application that requires HTTP authentication.

To configure a Webclip portlet with HTTP authentication

1. To begin configuring the Webclip portlet, follow step 1 through step 8 in "Configuring the Webclip Portlet Without Authentication" on page 143.
2. In the new browser window, scroll down to the **Access** property and click **HTTP auth**.
3. In the **Username** field, type the name required for authentication.
4. In the **Password** field, type the password required for authentication.
5. Scroll down to the **Location** property and, in the **URL** field, type the URL of the target Web site or Web application from which you want to clip Web content and click **Load**.

   The target page is loaded into the upper portion of the browser window.
6. Complete configuration of the Webclip portlet by following step 10 through step 16 in "Configuring the Webclip Portlet Without Authentication" on page 143.

Configuring Webclip with Forms-Based Authentication

The Webclip portlet provides forms-based authentication for passing user login and password credentials from the Portal server to the target Web site or Web application that is being clipped.

To configure a Webclip portlet with forms-based authentication

1. To begin configuring the Webclip portlet, follow step 1 through step 8 in "Configuring the Webclip Portlet Without Authentication" on page 143.
2. In the new browser window, scroll down to the **Access** property and click **Forms-based Auth**.
3. In the **Login URL** field, type the URL for the login page for the target Web site or Web application you want to clip.

4 Click **Capture**.

5 In the **Form URL** field of the new browser window, if the URL is not already filled in, type the URL of the login page and click **Load**.

The target page is loaded into the upper portion of the browser window.

6 Click the form to select it.

The content of the form is highlighted with a yellow border.

7 Fill out the login form but *do not* click the form's Submit button.

8 Click **Capture**.

9 Scroll down to the **Location** property and, in the **URL** field, type the URL of the target Web site or Web application from which you want to clip Web content and click **Load**.

The target page is loaded into the upper portion of the browser window.

10 Complete configuration of the Webclip portlet by following step 10 through step 16 in "Configuring the Webclip Portlet Without Authentication" on page 143.

Portlet Wiring

The webMethods Portal wiring technology allows portlet-specific properties, data, and events to be passed to another portlet to dynamically drive what information or business function is displayed in a resulting portlet or series of portlets.

Portlet-to-Portlet Wiring

Using portlet-to-portlet wiring, you can wire a property from one portlet to a property on another portlet. In this simple example, we wire a postal code property that is exposed in an Yahoo! Weather portlet to the postal code property that is exposed on a MapQuest portlet. While this example is not very useful, it illustrates the basic concept of portlet-to-portlet property-based wiring.

To wire two portlets together

1 Create a new portal page as described in "Creating a Portal Page" on page 109.

2 In the upper right-hand corner of the portal page, click the menu icon, and on the menu, click **Edit Portal Page**.

3 Add a MapQuest portlet to your portal page by following these steps:

a In the **Root** list of the **Available Portlets** panel, click **Portlets**.

b In the **Portlets** list of the **Available Portlets** panel, click **Internet - Tools**.

- **c** In the **Internet - Tools** list, drag the **MapQuest** portlet and drop it into one of the columns on your portal page.

4. Add a Yahoo! Weather portlet to your portal page by following these steps:

   - **a** In the **Root** list of the **Page Editor**, click **Portlets**.
   - **b** In the **Portlets** list of the **Page Editor**, click **Internet - News/Weather**.
   - **c** In the **Internet - News/Weather** list of the **Page Editor**, drag the **Yahoo! Weather** portlet and drop it into a column on your portal page.

5. On the left side of the page control area, click **Save**.

6. From the run-time view of your portal page, click ▤ (Popup Menu) at the right edge of the Yahoo! Weather portlet toolbar and then click **Properties**.

7. In the **Zip Codes** property of the Properties page, type a valid zip code value and, at the bottom of the page, click **Apply**.

8. From the run-time view of your portal page, click ▤ (Popup Menu) at the right edge of the MapQuest toolbar and then click **Wiring**.

   The wiring dialog page displays a list of properties on the MapQuest portlet that are available for wiring to other portlets.

9. In the **Portlet** list for the **Zip Code** property, select Yahoo! Weather.

10. In the **ZipCode** row, click **Browse**.

    The **Choose the Wired Property** dialog box contains a list of the properties of the Yahoo! Weather portlet that can be wired to properties in the MapQuest portlet.

11. Select the **Zip Codes** property and click **Select**.

12. On the Wiring page, click **Submit**.

    The run-time view of this page should display the MapQuest portlet with whatever value is configured in the Yahoo! Weather portlet.

13. To test the wiring functionality, go back to the Yahoo! Weather portlet and change the postal code. The MapQuest portlet should automatically be updated with the new postal code value.

webMethods

APPENDIX A

Property Editor Reference

Property Editor Reference Overview

Property editors provide the features that make up a property you are creating, allowing you to add text fields, radio buttons, lists, and other elements to a portlet. You can use property editors as the basis for portlet properties (see "Attributes of a Property" on page 46) or as building blocks in the Form Builder (see "Using the Form Builder" on page 65). The following sections describe some of the property editors that are available. Property editors are organized into the following Drawers:

- "Extended Drawer" on page 150
- "Portal Resources Drawer" on page 155
- "Simple Drawer" on page 178
- "Java Drawer" on page 206
- "Search Drawer" on page 224
- "wm_tabs Drawer" on page 226
- "Table Drawer" on page 227
- "Portlet Properties Drawer" on page 235
- "Internationalization Drawer" on page 236

Extended Drawer

The Extended Drawer in the Form Builder contains the following property editors.

- "Auth Scheme Property Editor" on page 150
- "File Encoding Property Editor" on page 151
- "File Size Property Editor" on page 152
- "Image Source Property Editor" on page 152
- "PCA Form Submit Property Editor" on page 153
- "PCA Register HTML Input Property Editor" on page 154
- "SQL DataSource Property Editor" on page 155

Auth Scheme Property Editor

The Auth Scheme property editor provides a drop-down list with a choice of available authentication schemes. You can modify the following properties in the Properties view.

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

File Encoding Property Editor

The File Encoding property editor provides a drop-down list with a choice of available file character encodings. You can modify the following properties in the Properties view.

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.

Property	Description
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

File Size Property Editor

The File Size property editor provides a formatted display for a property whose value represents the size of a file in bytes.

Image Source Property Editor

The Image Source property editor provides the ability for a user to choose an image from one of three options: an external Web site, the portal, or the current skin.

For this option...	The user does this...
Web resource URL	Types the image value as part of the URL. For example: `http://www.google.com/images/logo.gif`
Portal content	Clicks **Browse** to open a portal resource selector, browses to the image and then selects it. Alternatively, the user can click **Use Alias** and type the alias of the target image.
Skin Property	Types the property name of the image prefixed by the `skin:` scheme. For example: `skin:images/logo.gif`

You can modify the following properties in the Properties view.

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

PCA Form Submit Property Editor

By default, a PCA controller form allows you to submit the form data only to a single PCA method or redirect to a single PCA layout.The PCA Form Submit property editor allows you to have more than one action per form. This property editor is rendered as a Submit button. Then the user clicks this button, the form is submitted to the specified PCA Method or, if void, to a PCA Layout.

You can modify the following properties in the Properties view.

Property	Description
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
PCA Layout	A PCA Layout. Type the name of the PCA Layout to which the form should be submitted if no PCA Method is specified.
PCA Method	A PCA Method. Type the name of the PCA Method to which the form should be submitted. If you leave this property as `void`, the form is submitted to the layout specified in the PCA Layout property.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.

PCA Register HTML Input Property Editor

The PCA Register HTML Input property editor allows you to use bare HTML inputs inside a restful PCA form. By default only portal tags are allowed within the restful PCA form to build the form UI. But if you want to use bare HTML tags, such as <input name="myinput" type="text" />, you need to register this input with the PCA controller using this property editor. The name of the HTML element must be the same as the name of the PCA Register HTML Input property editor. For example:

```
<input name="myname" type="text"/>
<ui:propertyEditor>
    <util:param name='type' value='pcaregisterinput'></util:param>
    <util:param name='name' value='myname'></util:param>
</ui:propertyEditor>
```

If a bare HTML tag is not registered with PCA, the restful state is not properly handled, which may lead to multiple values for the "myname" property.

You can modify the following property in the Properties view.

Property	Description
Name	Type the name of the HTML element to be registered.

SQL DataSource Property Editor

The SQL DataSource property editor provides a drop-down list with a choice of SQL data sources that are currently configured in the portal (using the DataSource Administration portlet). This property editor is used when you add a DB Query to a portlet. You can modify the following properties in the Properties view.

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

Portal Resources Drawer

The Portal Resources Drawer provides property editors that make portal resources available in the form.

- "Directory Service Property Editor" on page 156
- "Portal Resource Property Editor" on page 157
- "Portal Resource (Popup) Property Editor" on page 159
- "Portal Resources List (Popup) Property Editor" on page 162

- "Principal Property Editor" on page 164
- "Principal (Popup) Property Editor" on page 166
- "Principal List (Popup) Property Editor" on page 169
- "Renderer Property Editor" on page 171
- "Shell Property Editor" on page 172
- "Skin Property Editor" on page 173
- "Thing Property Editor" on page 173
- "Thing (Popup) Property Editor" on page 176

Directory Service Property Editor

The Directory Services property editor provides a drop-down list with a choice of available directory services. You can modify the following properties in the Properties view.

Property	Description
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Style	The style to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.

Property	Description
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

Portal Resource Property Editor

The Portal Resource property editor provides a universal picker instance for picking portal resources in the portal. This property editor is similar to the Thing property editor ("Thing Property Editor" on page 173) but does not specify the root folders. You can navigate to items in a panel of available items on the left and then move them into the right panel to select them. Compare with "Portal Resource (Popup) Property Editor" on page 159.

You can modify the following properties in the Properties view:

Property	Description

Available Start	If this property has an entry, it specifies the path to an initial available container. You can root the property editor in a higher-level folder, such as the Root folder, but initially display a lower-level folder, such as the Portlets folder. The path is a set of comma-separated portal aliases, as in this example:
	`folder.root,folder.system,folder.shells`
	You can create multiple Available Start paths, with each path as an item in a list. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.
	`The root folder for each path must also appear in the Roots property.`

Base Types	If this property has an entry, the picker can only select items that are of the specified base types. By default, this property is empty. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Valid entries are folder, content, link, form, user, group, and role.

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Multiple	A boolean value set to True if the property editor can have multiple items selected and False if the editor cannot do so. The default is False. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Roots	Provides a list of roots to display in the Available Items pane. By default, the list is empty. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Valid entries include any portal alias. For example, folder.public.
Selectable	A JavaScript expression that allows fine-grained control of what resources a user can select. The expression should evaluate to true for a particular resource if the current user can select the resource. For example, to allow users to select only resources with `Folder` in the name, you could use this expression: `this.sName.indexOf('Folder') != -1` Within the JavaScript expression you can use the following properties to access information about each resource to which the expression is applied:

Property	Description
`this.sURI`	ID of the resource, for example: /meta/default/folder/0000000123
`this.sName`	Name of the resource, for example: Public Folders
`this.sDescription`	Description of the resource, for example: Publish public documents here
`this.sType`	Base-type name of the resource, for example: folder
`this.sXType`	Extended-type name of the resource, for example: wm_xt_webdavfolder
`this.nRights`	Current user's right bits for the resource, for example: 255

Property	Description
Unselectable	A JavaScript expression that allows fine-grained control of what resources a user can unselect. The expression should evaluate to true for a particular resource if the current user can unselect the resource. See the Selectable property for an example and further information.
Xtypes	If this property has an entry, the picker can only select items that are of the specified xtypes. By default, this property is empty. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Valid entries include any portal xtype or DBO name. For example, wm_xt_shortcut.

Portal Resource (Popup) Property Editor

The Portal Resource (Popup) property editor provides a Browse button that opens a portal resource selector. The user navigates to items in a panel of available items on the left and then moves them into the right panel to select them. Compare with "Portal Resource Property Editor" on page 157.

The Portal Resource (Popup) property editor also provides a Use Alias button. When clicked, the button opens a Select Alias window in which the user can select and test an alias for a target item.

You can modify the following properties in the Properties view:

Property	Description
Available Start	If this property has an entry, it specifies the path to an initial available container. You can root the property editor in a higher-level folder, such as the Root folder, but initially display a lower-level folder, such as the Portlets folder. The path is a set of comma-separated portal aliases, as in this example: `folder.root,folder.system,folder.shells` You can create multiple Available Start paths, with each path as an item in a list. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. `The root folder for each path must also appear in the Roots property.`
Base Types	If this property has an entry, the picker can only select items that are of the specified base types. By default, this property is empty. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Valid entries are folder, content, link, form, user, group, and role.

Property	Description
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Display Alias Button	A boolean value set to True to display the Use Alias button (the default) and False to hide the button. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Multiple	A boolean value set to True if the property editor can have multiple items selected and False if the editor cannot do so. The default is False. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Roots	Provides a list of roots to display in the Available Items pane. By default, this property is empty. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.
Selectable	A JavaScript expression that allows fine-grained control of what resources a user can select. The expression should evaluate to true for a particular resource if the current user can select the resource. For example, to allow users to select only resources with `Folder` in the name, you could use this expression: `this.sName.indexOf('Folder') != -1` Within the JavaScript expression you can use the following properties to access information about each resource to which the expression is applied:

Property	Description
`this.sURI`	ID of the resource, for example: /meta/default/folder/0000000123
`this.sName`	Name of the resource, for example: Public Folders

Property	Description	
	`this.sDescription`	Description of the resource, for example: Publish public documents here
	`this.sType`	Base-type name of the resource, for example: folder
	`this.sXType`	Extended-type name of the resource, for example: wm_xt_webdavfolder
	`this.nRights`	Current user's right bits for the resource, for example: 255
Style	The style to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.	
Unselectable	A JavaScript expression that allows fine-grained control of what resources a user can unselect. The expression should evaluate to true for a particular resource if the current user can unselect the resource. See the Selectable property for an example and further information.	
Xtypes	If this property has an entry, the picker can only select items that are of the specified xtypes. By default, this property is empty. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.	

Portal Resources List (Popup) Property Editor

The Portal Resources List (Popup) property editor provides an editable list of portal resources. You can allow the user to add, remove, reorder, or edit list items. You can modify the following properties in the Properties view.

0

Property	Description

Available Start	If this property has an entry, it specifies the path to an initial available container. You can root the property editor in a higher-level folder, such as the Root folder, but initially display a lower-level folder, such as the Portlets folder. The path is a set of comma-separated portal aliases, as in this example:
	`folder.root,folder.system,folder.shells`
	You can create multiple Available Start paths, with each path as an item in a list. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.
	`The root folder for each path must also appear in the Roots property.`

Base Types	If this property has an entry, the picker can only select items that are of the specified base types. By default, this property is empty. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Valid entries are folder, content, link, form, user, group, and role.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Roots	Provides a list of roots to display in the Available Items pane. By default, the list is empty. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Valid entries include any portal alias. For example, folder.public.

Property	Description
Selectable	A JavaScript expression that allows fine-grained control of what resources a user can select. The expression should evaluate to true for a particular resource if the current user can select the resource. For example, to allow users to select only resources with `Folder` in the name, you could use this expression: `this.sName.indexOf('Folder') != -1` Within the JavaScript expression you can use the following properties to access information about each resource to which the expression is applied:
	Property / **Description**
	`this.sURI` — ID of the resource, for example: /meta/default/folder/0000000123
	`this.sName` — Name of the resource, for example: Public Folders
	`this.sDescription` — Description of the resource, for example: Publish public documents here
	`this.sType` — Base-type name of the resource, for example: folder
	`this.sXType` — Extended-type name of the resource, for example: wm_xt_webdavfolder
	`this.nRights` — Current user's right bits for the resource, for example: 255
Show Order	A boolean value set to True to display the Up and Down buttons used for ordering list items, and False to hide the buttons. The default is True. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Size	An integer that specifies the size of the list box, in lines. If there are more list items, the box becomes scrollable.

Property	Description
Unselectable	A JavaScript expression that allows fine-grained control of what resources a user can unselect. The expression should evaluate to true for a particular resource if the current user can unselect the resource. See the Selectable property for an example and further information.
Xtypes	If this property has an entry, the picker can only select items that are of the specified xtypes. By default, this property is empty. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Valid entries include any portal xtype or DBO name. For example, wm_xt_shortcut.

Principal Property Editor

The Principal property editor provides a universal picker instance for picking users, groups and roles in the portal. This Property editor is similar to the Portal Resource property editor ("Portal Resource Property Editor" on page 157). You can navigate to items in a panel of available items on the left and then move them into the right panel to select them. Compare with "Principal (Popup) Property Editor" on page 166.

You can modify the following properties in the Properties view:

Property	Description

Available Start	If this property has an entry, it specifies the path to an initial available container. You can root the property editor in a higher-level folder, such as the Root folder, but initially display a lower-level folder, such as the Portlets folder. The path is a set of comma-separated portal aliases, as in this example:
	`folder.root,folder.system,folder.shells`
	You can create multiple Available Start paths, with each path as an item in a list. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.
	`The root folder for each path must also appear in the Roots property.`

Base Types	If this property has an entry, the picker can only select items that are of the specified base types. The default list is: Users, Groups, and Roles. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.

Property	Description
Multiple	A boolean value set to True if the property editor can have multiple items selected and False if the editor cannot do so. The default is False. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Roots	Provides a list of roots to display in the Available Items pane. The default list is: directory.system.root. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.
Selectable	A JavaScript expression that allows fine-grained control of what resources a user can select. The expression should evaluate to true for a particular resource if the current user can select the resource. For example, to allow users to select only resources with `Folder` in the name, you could use this expression: `this.sName.indexOf('Folder') != -1` Within the JavaScript expression you can use the following properties to access information about each resource to which the expression is applied:

Property	Description
`this.sURI`	ID of the resource, for example: /meta/default/folder/0000000123
`this.sName`	Name of the resource, for example: Public Folders
`this.sDescription`	Description of the resource, for example: Publish public documents here
`this.sType`	Base-type name of the resource, for example: folder
`this.sXType`	Extended-type name of the resource, for example: wm_xt_webdavfolder
`this.nRights`	Current user's right bits for the resource, for example: 255

Property	Description
Unselectable	A JavaScript expression that allows fine-grained control of what resources a user can unselect. The expression should evaluate to true for a particular resource if the current user can unselect the resource. See the Selectable property for an example and further information.
Xtypes	If this property has an entry, the picker can only select items that are of the specified xtypes. By default, this property is empty. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.

Principal (Popup) Property Editor

The Principal (Popup) property editor provides a Browse button that opens a universal picker instance for picking users, groups, or roles in the portal. You can navigate to items in a panel of available items on the left and then move them into the right panel to select them. Compare with "Principal Property Editor" on page 164.

The Principal (Popup) property editor also provides a Use Alias button. When clicked, the button opens a Select Alias window in which the user can select and test an alias for a target item.

You can modify the following properties in the Properties view:

Property	Description
Available Start	If this property has an entry, it specifies the path to an initial available container. You can root the property editor in a higher-level folder, such as the Root folder, but initially display a lower-level folder, such as the Portlets folder. The path is a set of comma-separated portal aliases, as in this example: `folder.root,folder.system,folder.shells` You can create multiple Available Start paths, with each path as an item in a list. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. `The root folder for each path must also appear in the Roots property.`
Base Types	If this property has an entry, the picker can only select items that are of the specified base types. The default list is: users, groups, and roles. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.

Property	Description
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Display Alias Button	A boolean value set to True to display the Use Alias button (the default) and False to hide the button. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Multiple	A boolean value set to True if the property editor can have multiple items selected and False if the editor cannot do so. The default is False. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Roots	Provides a list of roots to display in the Available Items pane. The default list is: directory.system.root. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.
Selectable	A JavaScript expression that allows fine-grained control of what resources a user can select. The expression should evaluate to true for a particular resource if the current user can select the resource. For example, to allow users to select only resources with `Folder` in the name, you could use this expression: `this.sName.indexOf('Folder') != -1` Within the JavaScript expression you can use the following properties to access information about each resource to which the expression is applied:

Property	Description
`this.sURI`	ID of the resource, for example: /meta/default/folder/0000000123
`this.sName`	Name of the resource, for example: Public Folders

Property	Description	
	`this.sDescription`	Description of the resource, for example: Publish public documents here
	`this.sType`	Base-type name of the resource, for example: folder
	`this.sXType`	Extended-type name of the resource, for example: wm_xt_webdavfolder
	`this.nRights`	Current user's right bits for the resource, for example: 255
Style	The style to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.	
Unselectable	A JavaScript expression that allows fine-grained control of what resources a user can unselect. The expression should evaluate to true for a particular resource if the current user can unselect the resource. See the Selectable property for an example and further information.	
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.	
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.	
Xtypes	If this property has an entry, the picker can only select items that are of the specified xtypes. By default, this property is empty. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.	

Principal List (Popup) Property Editor

The Principal List (Popup) property editor provides an editable list of users, groups, and roles. You can allow the user to add, remove, reorder, or edit list items. You can modify the following properties in the Properties view.

0

Property	Description
Available Start	If this property has an entry, it specifies the path to an initial available container. You can root the property editor in a higher-level folder, such as the Root folder, but initially display a lower-level folder, such as the Portlets folder. The path is a set of comma-separated portal aliases, as in this example: `folder.root,folder.system,folder.shells` You can create multiple Available Start paths, with each path as an item in a list. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. `The root folder for each path must also appear in the Roots property.`
Base Types	If this property has an entry, the picker can only select items that are of the specified base types. The default list is: Users, Groups, and Roles. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Valid entries are folder, content, link, form, user, group, and role.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Display Alias Button	A boolean value set to True to display the Use Alias button (the default) and False to hide the button. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Roots	Provides a list of roots to display in the Available Items pane. The default list is: directory.system.root. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Valid entries include any portal alias. For example, folder.public.

Property	Description
Selectable	A JavaScript expression that allows fine-grained control of what resources a user can select. The expression should evaluate to true for a particular resource if the current user can select the resource. For example, to allow users to select only resources with `Folder` in the name, you could use this expression: `this.sName.indexOf('Folder') != -1` Within the JavaScript expression you can use the following properties to access information about each resource to which the expression is applied:

Property	Description
`this.sURI`	ID of the resource, for example: /meta/default/folder/0000000123
`this.sName`	Name of the resource, for example: Public Folders
`this.sDescription`	Description of the resource, for example: Publish public documents here
`this.sType`	Base-type name of the resource, for example: folder
`this.sXType`	Extended-type name of the resource, for example: wm_xt_webdavfolder
`this.nRights`	Current user's right bits for the resource, for example: 255

Property	Description
Show Order	A boolean value set to True to display the Up and Down buttons used for ordering list items, and False to hide the buttons. The default is True. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Size	An integer that specifies the size of the list box, in lines. If there are more list items, the box becomes scrollable.
Unselectable	A JavaScript expression that allows fine-grained control of what resources a user can unselect. The expression should evaluate to true for a particular resource if the current user can unselect the resource. See the Selectable property for an example and further information.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.

Property	Description
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.
Xtypes	If this property has an entry, the picker can only select items that are of the specified xtypes. By default, this property is empty. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Valid entries include any portal xtype or DBO name. For example, wm_xt_shortcut.

Renderer Property Editor

The Renderer Property editor provides a drop-down list with a choice of available portal renderers. You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Skip	Provides a text field in which to type the ID of one renderer to be omitted from the list.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

Shell Property Editor

The Shell Property editor provides a drop-down list with a choice of available portal shells. You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Legacy Shells	A boolean value set to True if the property editor can display legacy shells and False if the editor cannot do so. The default is True. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Skip	Provides a text field in which to type the ID of one shell to be omitted from the list.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

Skin Property Editor

The Skin Property editor provides a drop-down list with a choice of available portal skins. You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Skip	Provides a text field in which to type the ID of one skin to be omitted from the list.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

Thing Property Editor

The Thing property editor provides a universal picker instance for picking objects in the portal. This Property editor is similar to the Portal Resource property editor ("Portal Resource Property Editor" on page 157) but specifies the root folders. You can navigate to items in a panel of available items on the left and then move them into the right panel to select them. Only one item can be selected at a time. If the right panel is occupied at the time a user moves an item into it, the previously selected item is removed. Compare with "Thing (Popup) Property Editor" on page 176.

You can modify the following properties in the Properties view:

Property	Description

Available Start	If this property has an entry, it specifies the path to an initial available container. You can root the property editor in a higher-level folder, such as the Root folder, but initially display a lower-level folder, such as the Portlets folder. The path is a set of comma-separated portal aliases, as in this example:
	`folder.root,folder.system,folder.shells`
	You can create multiple Available Start paths, with each path as an item in a list. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.
	`The root folder for each path must also appear in the Roots property.`

Base Types	If this property has an entry, the picker can only select items that are of the specified base types. By default, this property contains the Folder base type. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Multiple	A boolean value set to True if the property editor can have multiple items selected and False if the editor cannot do so. The default is False. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Roots	Provides a list of roots to display in the Available Items pane. The default list is: folder.root, folder.public, and user.current.home. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.

Property	Description
Selectable	A JavaScript expression that allows fine-grained control of what resources a user can select. The expression should evaluate to true for a particular resource if the current user can select the resource. For example, to allow users to select only resources with `Folder` in the name, you could use this expression: `this.sName.indexOf('Folder') != -1` Within the JavaScript expression you can use the following properties to access information about each resource to which the expression is applied:

Property	Description
`this.sURI`	ID of the resource, for example: /meta/default/folder/0000000123
`this.sName`	Name of the resource, for example: Public Folders
`this.sDescription`	Description of the resource, for example: Publish public documents here
`this.sType`	Base-type name of the resource, for example: folder
`this.sXType`	Extended-type name of the resource, for example: wm_xt_webdavfolder
`this.nRights`	Current user's right bits for the resource, for example: 255

Property	Description
Unselectable	A JavaScript expression that allows fine-grained control of what resources a user can unselect. The expression should evaluate to true for a particular resource if the current user can unselect the resource. See the Selectable property for an example and further information.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.

Property	Description
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.
Xtypes	If this property has an entry, the picker can only select items that are of the specified xtypes. By default, this property is empty. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.

Thing (Popup) Property Editor

The Thing (Popup) property editor provides a Browse button that opens a universal picker instance for picking objects in the portal. You can navigate to items in a panel of available items on the left and then move them into the right panel to select them. Only one item can be selected at a time. If the right panel is occupied at the time a user moves an item into it, the previously selected item is removed. Compare with "Thing Property Editor" on page 173.

The Thing (Popup) property editor also provides a Use Alias button. When clicked, the button opens a Select Alias window in which the user can select and test an alias for a target item.

You can modify the following properties in the Properties view:

Property	Description

Available Start	If this property has an entry, it specifies the path to an initial available container. You can root the property editor in a higher-level folder, such as the Root folder, but initially display a lower-level folder, such as the Portlets folder. The path is a set of comma-separated portal aliases, as in this example:
	`folder.root,folder.system,folder.shells`
	You can create multiple Available Start paths, with each path as an item in a list. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.
	`The root folder for each path must also appear in the Roots property.`

Base Types	If this property has an entry, the picker can only select items that are of the specified base types. By default, this property is empty. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.

Property	Description
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Display Alias Button	A boolean value set to True to display the Use Alias button (the default) and False to hide the button. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Multiple	A boolean value set to True if the property editor can have multiple items selected and False if the editor cannot do so. The default is False. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Roots	Provides a list of roots to display in the Available Items pane. The default list is: folder.root, folder.public, and user.current.home. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.
Selectable	A JavaScript expression that allows fine-grained control of what resources a user can select. The expression should evaluate to true for a particular resource if the current user can select the resource. For example, to allow users to select only resources with `Folder` in the name, you could use this expression:

```
this.sName.indexOf('Folder') != -1
```

Within the JavaScript expression you can use the following properties to access information about each resource to which the expression is applied:

Property	Description
`this.sURI`	ID of the resource, for example: /meta/default/folder/0000000123
`this.sName`	Name of the resource, for example: Public Folders

Property	Description	
	`this.sDescription`	Description of the resource, for example: Publish public documents here
	`this.sType`	Base-type name of the resource, for example: folder
	`this.sXType`	Extended-type name of the resource, for example: wm_xt_webdavfolder
	`this.nRights`	Current user's right bits for the resource, for example: 255
Style	The style to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.	
Unselectable	A JavaScript expression that allows fine-grained control of what resources a user can unselect. The expression should evaluate to true for a particular resource if the current user can unselect the resource. See the Selectable property for an example and further information.	
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.	
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.	
Xtypes	If this property has an entry, the picker can only select items that are of the specified xtypes. By default, this property is empty. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.	

Simple Drawer

The Simple Drawer provides property editors that make available a variety of form input mechanisms, and other common features.

- "Boolean Checkbox Property Editor" on page 179
- "Button Property Editor" on page 180

- "Checkbox Property Editor" on page 181
- "Checkbox Group Property Editor" on page 182
- "Confirm Password Property Editor" on page 183
- "Date Property Editor" on page 184
- "File Property Editor" on page 186
- "Hidden Property Editor" on page 186
- "HTML Area Property Editor" on page 187
- "Label Property Editor" on page 188
- "Label with Fallback Property Editor" on page 188
- "List Property Editor" on page 189
- "Localizable Label Property Editor" on page 190
- "Map Property Editor" on page 191
- "Multi-Select Property Editor" on page 191
- "Ordered Map Property Editor" on page 193
- "Password Property Editor" on page 194
- "Popup HTML Area Property Editor" on page 195
- "Popup Text Area Property Editor" on page 196
- "Radio Button Property Editor" on page 197
- "Radio Button Group Property Editor" on page 198
- "Single-Select Property Editor" on page 199
- "Submit Button Property Editor" on page 201
- "Swap-Box Property Editor" on page 201
- "Text Area Property Editor" on page 203
- "Text Property Editor" on page 205

Boolean Checkbox Property Editor

The Boolean Checkbox property editor provides a check box that passes a True or False state when the form is submitted. Compare with "Checkbox Property Editor" on page 181. This property editor does not include a feature for a text title. To add text, pair the Checkbox property editor with the "Label Property Editor" on page 188.

You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are none (the default), number, login, small, medium, and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
On-click Handler	A string. Type a JavaScript the property editor should execute when the button is clicked.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

Button Property Editor

The Button property editor places a button on the form. In the Value property, type the text that should appear on the button. You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are none (the default), number, login, small, medium, and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.

Property	Description
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
On-click Handler	A text string. Type a JavaScript the property editor should execute when the button is clicked. Do not use this property unless the DHTML Allowed property is set to True.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.
Value	Text string. Type the text that should appear on the face of the button.

Checkbox Property Editor

The Checkbox property editor places a check box on the form. If the check box is selected when the form is submitted, the Value is passed. Compare with "Checkbox Group Property Editor" on page 182. This property editor does not include a feature for a text title. To add text, pair the Checkbox property editor with the "Label Property Editor" on page 188.

You can modify the following properties in the Properties view:

Property	Description
Checked	A boolean value set to True if the check box is selected and False if the check box is cleared. The default is False. To display the choices, click the **Value** column and then click the button at the right edge of the column.
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are none (the default), number, login, small, medium, and large.

Property	Description
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
On-click Handler	A string. Type a JavaScript the property editor should execute when the button is clicked.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.
Value	A string. Type the value that is to be passed if the Checked property is True.

Checkbox Group Property Editor

The Checkbox Group property editor provides multiple check boxes and titles. By default, all check boxes are cleared. If a check box is selected when the form is submitted, the Values property value is passed. Compare with "Checkbox Property Editor" on page 181 and "Multi-Select Property Editor" on page 191.

The Titles property provides text for the titles; the Values property provides values to be passed. One check box is displayed for each value in the Values property. If there is a mismatch in the number of values between the Titles and Values properties:

If there are more...	The property editor does this...
Titles property values	Uses only enough Titles values to equal the number of Values values.
Values property values	Uses the Values property value in place of missing titles.

You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are none (the default), number, login, small, medium, and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Titles	Provides a list of titles to display with check boxes. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add entries in the order in which they should appear. If you leave this property empty, the default is to use values in the Values property.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.
Values	Provides a list of values associated with check boxes. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add values in the order in which they should appear.

Confirm Password Property Editor

The Password property editor provides a text field in which to type a confirmation password. Use this property editor in concert with the "Password Property Editor" on page 194. You set the Validation field parameter of the Confirm Password property editor to the name of the Password property editor and the Validation Message parameter to the message that should be displayed if the passwords do not match.

For example, assume that the Name field in the Password property editor is `LDAPpassword`. In the Confirm Password property editor, the Validation field should also be `LDAPpassword`.

You can modify the following additional property in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are none, number, login, small, medium (the default), and large.
On-blur Handler	A text string. Type a JavaScript the property editor should execute when focus is taken away from the input field.
On-enter Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses ENTER.
On-focus Handler	A text string. Type a JavaScript the property editor should execute when focus is given to the input field.
On-key-press Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses a key.
On-select Handler	A text string. Type a JavaScript the property editor should execute when the input field content is selected.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

Date Property Editor

The Date property editor provides a pop-up date and time editor. The user clicks an icon to open a calendar from which to choose a date and time. You can modify the pattern in which the date and time are displayed. You can modify the following properties in the Properties view:

Property	Description
Allow Manual Entry	A boolean value set to True (the default) if the user is allowed to type a date value into the field and False if the field is not editable. To display the choices, click the **Value** column and then click the button at the right edge of the column.
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are none, number, login, small, medium (the default), and large.
Default Value	A default date that appears when the form is first displayed. Type a date using the format in the Pattern property. If you leave the Default Value property empty, the field is empty by default.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Pattern	Type a pattern in which to display the date and time. The pattern should conform to the java.text.SimpleDateFormat class. The default is `yyyy-MM-dd HH:mm`.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

File Property Editor

The File property editor provides a file-upload input field. The user clicks a Browse button, opening a window in which to locate the file. This property editor supports the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are none, number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

Hidden Property Editor

The Hidden property editor places a hidden input field on the form. When a user submits a form, the browser sends the form field value to the server just the same as it would a standard text input field, even though the field isn't displayed to the user.

You can modify the following properties in the Properties view:

Property	Description
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.
Value	The hidden input value. Type the value to be passed by the form.

HTML Area Property Editor

The HTML Area property editor places an HTML editor on the form. You can specify the width of the editor in characters and the height in rows or in valid CSS units, such as pixels. Compare with "Popup HTML Area Property Editor" on page 195.

You can modify the following properties in the Properties view:

Property	Description
Cols	The width of the HTML editor in characters.
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are none, number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Height	The height of the HTML editor in valid CSS units, such as 300px.

Property	Description
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Rows	The height of the HTML editor in lines.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

Label Property Editor

The Label property editor provides a static text field that you can use as a label. Compare with "Localizable Label Property Editor" on page 190 and "Label with Fallback Property Editor" on page 188. You can modify the following property in the Properties view:

Property	Description
Value	A text string. Type a text string containing the text to be displayed.

Label with Fallback Property Editor

The Label with Fallback property editor provides a static text Value field that you can use as a label. If the Value field is empty, the property editor uses the localized text identified by the Fallback Resource Key field in the resource bundle class identified by Fallback Resource Bundle. Compare with "Label Property Editor" on page 188 and"Localizable Label Property Editor" on page 190. You can modify the following properties in the Properties view:

Property	Description
Fallback Resource Bundle	The class name of the resource bundle from which to take the label text.

Property	Description
Fallback Resource Key	The resource key that identifies a locale-specific object in the resource bundle.
Value	A text string. Type a text string containing the text to be displayed. Leave this field empty if you want to use a localized label in the resource bundle.

List Property Editor

The List property editor provides a editable list of strings. You can allow the user to add, remove, reorder, or edit list items. You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are none (the default), number, login, small, medium, and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Show Edit Buttons	A boolean value set to True to display the Add, Edit, and Remove buttons, and False to hide the buttons. The default is True. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Show Order Buttons	A boolean value set to True to display the Up and Down buttons used for ordering list items, and False to hide the buttons. The default is True. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Size	An integer that specifies the size of the list box, in lines. If there are more list items, the box becomes scrollable.

Property	Description
Titles	Provides an initial group of titles to display in the list. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add entries in the order in which they should appear.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

Localizable Label Property Editor

The Localizable Label property editor provides a a way to create a localizable label. You provide the class name of the resource bundle and the resource key that identifies the label text. Compare with "Label Property Editor" on page 188 and "Label with Fallback Property Editor" on page 188. You can modify the following properties in the Properties view:

Property	Description
Resource Bundle	The class name of the resource bundle from which to take the label text.
Value	The resource key that identifies a locale-specific object in the resource bundle.

Map Property Editor

The Map property editor provides a select box to which a user can add and edit name/value pairs, and remove them. Compare with "Ordered Map Property Editor" on page 193. You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are none, number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Show Edit Buttons	A boolean value set to True to display the Add, Edit, and Remove buttons, and False to hide the buttons. The default is True. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Size	An integer that specifies the size of the select box, in lines. If there are more mapping pairs, the box becomes scrollable.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

Multi-Select Property Editor

The Multi-Select property editor provides a multiple-selection box. If a list item is selected when the user submits the form, the Values property value is passed. Compare with "Checkbox Group Property Editor" on page 182 and "Multiple-Select Table Property Editor" on page 234.

The Titles property provides text for the titles; the Values property provides values to be passed. One list item is displayed for each value in the Values property. If there is a mismatch in the number of values between the Titles and Values properties:

If there are more...	The property editor does this...
Titles property values	Uses only enough Titles values to equal the number of Values values.
Values property values	Uses the Values property value in place of missing titles.

You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are none, number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Size	An integer that specifies the size of the list box, in lines. If there are more list items, the box becomes scrollable.
Styles	A list of CSS styles to apply to the list, one for each option. An example style is: `color:white;background-color:blue;`
Titles	A list of titles to display in the list. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add entries in the order in which they should appear. If you leave this property empty, the default is to use values in the Values property.
Values	A list of values associated with list items. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add values in the order in which they should appear.

Property	Description
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

Ordered Map Property Editor

The Ordered Map property editor provides a select box to which a user can add and edit name/value pairs, change the order of the pairs, and remove them. Compare with"Map Property Editor" on page 191.You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are none, number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Show Edit Buttons	A boolean value set to True to display the Add, Edit, and Remove buttons, and False to hide the buttons. The default is True. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Size	An integer that specifies the size of the select box, in lines. If there are more mapping pairs, the box becomes scrollable.

Property	Description
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

Password Property Editor

The Password property editor provides a text field in which to enter a password. Use this property editor in concert with the "Confirm Password Property Editor" on page 183. You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are none, number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-blur Handler	A text string. Type a JavaScript the property editor should execute when focus is taken away from the input field.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
On-enter Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses ENTER.
On-focus Handler	A text string. Type a JavaScript the property editor should execute when focus is given to the input field.
On-key-press Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses a key.
On-select Handler	A text string. Type a JavaScript the property editor should execute when the input field content is selected.

Property	Description
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

Popup HTML Area Property Editor

The Popup HTML Area property editor places an Edit WYSIWYG button on the form that opens an HTML editor in a pop-up window and an Edit Source button that opens a text editor. A Preview property defines the number of spaces for a preview of the value to the left of the buttons and the Value property contains the text of the title. Compare with "HTML Area Property Editor" on page 187.

You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are none, number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Preview Characters	An integer. Type an integer denoting the number of characters to be displayed in the preview preceding the buttons.
Value	A text string. Type a text string for the Preview area. If the text exceeds the value in Preview Characters, the text is truncated.

Property	Description
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

Popup Text Area Property Editor

The Popup Text Area property editor provides a multi-line text field that is made available by means of an Edit button. A Preview property defines the number of spaces for a preview to the left of the Edit button and the Value property contains the entire text.You can modify the following additional properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are none, number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Preview Characters	An integer. Type an integer denoting the number of characters to be displayed in the preview preceding the buttons.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.

Property	Description
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.
Value	A text string. Type a text string for the Preview area that precedes the Edit button. If the text exceeds the value in Preview Characters, the text is truncated.

Radio Button Property Editor

The Radio Button property editor places a radio button on the form. If the radio button is selected when the form is submitted, the value is passed. This property editor does not include a feature for a text title. To add text, pair the Radio Button property editor with the "Label Property Editor" on page 188.

This property editor works like the standard HTML radio button control, in that rendering multiple radio buttons in a form with the same Name value creates a radio button group. Only one radio button can be selected at one time. Compare with "Radio Button Group Property Editor" on page 198 and "Single-Select Table Property Editor" on page 227, in which radio buttons are already grouped.

You can modify the following properties in the Properties view:

Property	Description
Checked	A boolean value set to True if the radio button is selected and False if the radio button is cleared. The default is False. To display the choices, click the **Value** column and then click the button at the right edge of the column.
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are none (the default), number, login, small, medium, and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Name	The name assigned to this property editor on the form. Use this property to combine two or more Radio Button property editors into a group so that only can be selected at a time. Type the same Name value for each radio button on the form.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.

Property	Description
On-click Handler	A string. Type a JavaScript the property editor should execute when the button is clicked.
Value	A string. Type the value that is to be passed if the Checked property is True.

Radio Button Group Property Editor

The Radio Button Group property editor provides multiple radio buttons and titles. Only one radio button in the group can be selected at a time. If a radio button is selected when the form is submitted, the Values property value for that button is passed. Compare with "Radio Button Property Editor" on page 197 and "Single-Select Property Editor" on page 199.

The Titles property provides text for the titles; the Values property provides values to be passed. One radio button is displayed for each value in the Values property. If there is a mismatch in the number of values between the Titles and Values properties:

If there are more...	The property editor does this...
Titles property values	Uses only enough Titles values to equal the number of Values values.
Values property values	Uses the Values property value in place of missing titles.

You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are none (the default), number, login, small, medium, and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.

Property	Description
Titles	Provides a list of titles to display with radio buttons. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add entries in the order in which they should appear. If you leave this property empty, the default is to use values in the Values property.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.
Values	Provides a list of values associated with the radio buttons. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add values in the order in which they should appear.

Single-Select Property Editor

The Single-Select property editor provides a single-select drop-down list. The value of the selected list item is passed when the user submits the form. Compare with "Radio Button Group Property Editor" on page 198.

The Titles property provides text for the titles; the Values property provides values to be passed. One list item is displayed for each value in the Values property. If there is a mismatch in the number of values between the Titles and Values properties:

If there are more...	The property editor does this...
Titles property values	Uses only enough Titles values to equal the number of Values values.
Values property values	Uses the Values property value in place of missing titles.

By default, the Single-Select property editor is displayed as a drop-down list but you can use the Size property to change the display to a select box.

You can use the Show Other property to provide the user with the capability of typing in a value that is not in the list. If the user clicks the Other option, a user prompt window appears in which to type a value.

You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are none, number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Show Other	A boolean value set to True to cause the list to contain an Other option and False to hide the option. The default is False. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Size	An integer that specifies the size of the list box, in lines. If this property is empty, which is the default, the property editor is displayed as a drop-down list.
Titles	Provides a list of titles to display as list items. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add entries in the order in which they should appear. If you leave this property empty, the default is to use values in the Values property.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.
Values	Provides a list of values associated with the list items. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add values in the order in which they should appear.

Submit Button Property Editor

The Submit property editor provides a button that, when clicked, submits the form. The Submit button is available by default when you create a new form in the Form Builder. You can modify the following additional property in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are none (the default), number, login, small, medium, and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.
Value	A text string. Type the text that should appear on the face of the button.

Swap-Box Property Editor

The Swap-Box property editor provides a pair of list boxes in which you can place multiple items for selection. the left box contains items available for selection and the right list box contains items that have been selected. The user can move one or all items between the two boxes. The values of the items in the right list box are passed when the user submits the form.

The Titles property provides text for the titles; the Values property provides values to be passed. One list item is displayed for each value in the Values property. If there is a mismatch in the number of values between the Titles and Values properties:

If there are more...	The property editor does this...
Titles property values	Uses only enough Titles values to equal the number of Values values.
Values property values	Uses the Values property value in place of missing titles.

By default, the Swap-Box property editor makes the list boxes large enough to display all items, but you can use the Size property to control the maximum number of items to be displayed at one time; if there are more items than can be displayed, the list boxes become scrollable.

You can modify the following properties in the Properties view.

Property	Description
Available Title	A text string. Type the label for the left list box, which contains items that have not been selected. If you leave the property empty the default label is **Available**. Compare with the Selected Title property.
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Orientation	A value set to horizontal (the default) if the list boxes are displayed side by side and vertical if the list box of available items is displayed above the list box of selected items. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.

Property	Description
Selected Title	A text string. Type the label for the right list box, which contains items that have been selected. If you leave the property empty the default label is **Selected**. Compare with the Available Title property.
Size	An integer that specifies the size of the list box, in lines. If there are more list items, the box becomes scrollable.
Styles	A list of CSS styles to apply to the list, one for each option. An example style is: `color:white;background-color:blue;`
Titles	A list of titles to display in the list. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add entries in the order in which they should appear. If you leave this property empty, the default is to use values in the Values property.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.
Values	A list of values associated with list items. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add values in the order in which they should appear.

Text Area Property Editor

The Text Area property editor provides a multi-line text field. You can modify the following properties in the Properties view:

Property	Description
Cols	The width of the text field in characters.
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-blur Handler	A text string. Type a JavaScript the property editor should execute when focus is taken away from the input field.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
On-enter Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses ENTER.
On-focus Handler	A text string. Type a JavaScript the property editor should execute when focus is given to the input field.
On-key-press Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses a key.
On-select Handler	A text string. Type a JavaScript the property editor should execute when the input field content is selected.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Rows	The height of the text field in lines. The default height is two lines. If a user types enough text to fill more rows, the text area becomes scrollable.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.

Property	Description
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.
Value	A text string. Type text you want to appear in the text area by default. The user can delete or replace this text.

Text Property Editor

The Text property editor provides a one-line text input field. You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-blur Handler	A text string. Type a JavaScript the property editor should execute when focus is taken away from the input field.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
On-enter Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses ENTER.
On-focus Handler	A text string. Type a JavaScript the property editor should execute when focus is given to the input field.
On-key-press Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses a key.
On-select Handler	A text string. Type a JavaScript the property editor should execute when the input field content is selected.

Property	Description
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.
Value	A text string. Type text you want to appear in the text area by default. The user can delete or replace this text.

Java Drawer

The Java Drawer provides property editors named for Java object classes they represent. Many of these property editors are aliases for other property editors.

- "Byte Property Editor" on page 207
- "Char Property Editor" on page 208
- "com.webmethods.portal.service.meta2.thing.IThingID Property Editor" on page 209
- "com.webmethods.portal.service.meta2.thing.IThingIDList Property Editor" on page 212
- "com.webmethods.portal.system.IURI Property Editor" on page 214
- "Double Property Editor" on page 215
- "Float Property Editor" on page 216
- "Int Property Editor" on page 217
- "java.lang.Boolean Property Editor" on page 217
- "java.lang.Byte Property Editor" on page 218
- "java.lang.Character Property Editor" on page 218
- "java.lang.Double Property Editor" on page 218

- "java.lang.Float Property Editor" on page 218
- "java.lang.Integer Property Editor" on page 218
- "java.lang.Long Property Editor" on page 218
- "java.lang.Object Property Editor" on page 218
- "java.lang.Short Property Editor" on page 218
- "java.math.BigDecimal Property Editor" on page 219
- "java.math.BigInteger Property Editor" on page 220
- "java.net.URL Property Editor" on page 221
- "java.util.Calendar Property Editor" on page 221
- "java.util.Date Property Editor" on page 222
- "Long Property Editor" on page 222
- "Short Property Editor" on page 223

Byte Property Editor

The Byte property editor provides a one-line text field to contain a single byte. You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-blur Handler	A text string. Type a JavaScript the property editor should execute when focus is taken away from the input field.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
On-enter Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses ENTER.
On-focus Handler	A text string. Type a JavaScript the property editor should execute when focus is given to the input field.

Property	Description
On-key-press Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses a key.
On-select Handler	A text string. Type a JavaScript the property editor should execute when the input field content is selected.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression in the Validation Regular Expression property. The default message is: `Value must be a byte`
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. The default regular expression is: `(^$)\|(^-?\d{1,3}$)`

Char Property Editor

The Char property editor provides a one-line text field to contain a single character. You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-blur Handler	A text string. Type a JavaScript the property editor should execute when focus is taken away from the input field.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
On-enter Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses ENTER.
On-focus Handler	A text string. Type a JavaScript the property editor should execute when focus is given to the input field.

Property	Description
On-key-press Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses a key.
On-select Handler	A text string. Type a JavaScript the property editor should execute when the input field content is selected.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression in the Validation Regular Expression property. The default message is: `Value must be a single character`
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. The default regular expression is: `^.?$`

com.webmethods.portal.service.meta2.thing.IThingID Property Editor

This property editor provides a Browse button that opens a universal picker instance for picking objects in the portal. You can navigate to items in a panel of available items on the left and then move them into the right panel to select them. Only one item can be selected at a time. If the right panel is occupied at the time a user moves an item into it, the previously selected item is removed. Compare with "com.webmethods.portal.service.meta2.thing.IThingIDList Property Editor" on page 212.

The property editor also provides a Use Alias button. When clicked, the button opens a Select Alias window in which the user can select and test an alias for a target item.

You can modify the following properties in the Properties view:

Property	Description

Available Start	If this property has an entry, it specifies the path to an initial available container. You can root the property editor in a higher-level folder, such as the Root folder, but initially display a lower-level folder, such as the Portlets folder. The path is a set of comma-separated portal aliases, as in this example:
	`folder.root,folder.system,folder.shells`
	You can create multiple Available Start paths, with each path as an item in a list. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.
	`The root folder for each path must also appear in the Roots property.`

Base Types	If this property has an entry, the picker can only select items that are of the specified base types. By default, this property contains a base type of folder. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Display Alias Button	A boolean value set to True to display the Use Alias button (the default) and False to hide the button. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Multiple	A boolean value set to True if the property editor can have multiple items selected and False if the editor cannot do so. The default is True. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Roots	Provides a list of roots to display in the Available Items pane. The default list is: folder.root, folder.public, and user.current.home. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.

Property	Description
Selectable	A JavaScript expression that allows fine-grained control of what resources a user can select. The expression should evaluate to true for a particular resource if the current user can select the resource. For example, to allow users to select only resources with `Folder` in the name, you could use this expression: `this.sName.indexOf('Folder') != -1` Within the JavaScript expression you can use the following properties to access information about each resource to which the expression is applied:

Property	Description
`this.sURI`	ID of the resource, for example: /meta/default/folder/0000000123
`this.sName`	Name of the resource, for example: Public Folders
`this.sDescription`	Description of the resource, for example: Publish public documents here
`this.sType`	Base-type name of the resource, for example: folder
`this.sXType`	Extended-type name of the resource, for example: wm_xt_webdavfolder
`this.nRights`	Current user's right bits for the resource, for example: 255

Property	Description
Style	The style to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Unselectable	A JavaScript expression that allows fine-grained control of what resources a user can unselect. The expression should evaluate to true for a particular resource if the current user can unselect the resource. See the Selectable property for an example and further information.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.

Property	Description
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.
Xtypes	If this property has an entry, the picker can only select items that are of the specified xtypes. By default, this property is empty. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.

com.webmethods.portal.service.meta2.thing.IThingIDList Property Editor

This property editor provides a Browse button that opens a universal picker instance for picking objects in the portal. You can navigate to items in a panel of available items on the left and then move them into the right panel to select them. Multiple items can be selected at the same time. Compare with "com.webmethods.portal.service.meta2.thing.IThingID Property Editor" on page 209.

The property editor also provides a Use Alias button. When clicked, the button opens a Select Alias window in which the user can select and test an alias for a target item.

You can modify the following properties in the Properties view:

Property	Description

Available Start	If this property has an entry, it specifies the path to an initial available container. You can root the property editor in a higher-level folder, such as the Root folder, but initially display a lower-level folder, such as the Portlets folder. The path is a set of comma-separated portal aliases, as in this example:
	`folder.root,folder.system,folder.shells`
	You can create multiple Available Start paths, with each path as an item in a list. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.
	`The root folder for each path must also appear in the Roots property.`

Base Types	If this property has an entry, the picker can only select items that are of the specified base types. By default, this property contains a base type of folder. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.

Property	Description
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Display Alias Button	A boolean value set to True to display the Use Alias button (the default) and False to hide the button. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Multiple	A boolean value set to True if the property editor can have multiple items selected and False if the editor cannot do so. The default is False. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Roots	Provides a list of roots to display in the Available Items pane. The default list is: folder.root, folder.public, and user.current.home. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.
Selectable	A JavaScript expression that allows fine-grained control of what resources a user can select. The expression should evaluate to true for a particular resource if the current user can select the resource. For example, to allow users to select only resources with `Folder` in the name, you could use this expression: `this.sName.indexOf('Folder') != -1` Within the JavaScript expression you can use the following properties to access information about each resource to which the expression is applied:

Property	Description
`this.sURI`	ID of the resource, for example: /meta/default/folder/0000000123
`this.sName`	Name of the resource, for example: Public Folders

Property	Description	
	`this.sDescription`	Description of the resource, for example: Publish public documents here
	`this.sType`	Base-type name of the resource, for example: folder
	`this.sXType`	Extended-type name of the resource, for example: wm_xt_webdavfolder
	`this.nRights`	Current user's right bits for the resource, for example: 255
Style	The style to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.	
Unselectable	A JavaScript expression that allows fine-grained control of what resources a user can unselect. The expression should evaluate to true for a particular resource if the current user can unselect the resource. See the Selectable property for an example and further information.	
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.	
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.	
Xtypes	If this property has an entry, the picker can only select items that are of the specified xtypes. By default, this property is empty. To display the List Editor, click the **Value** column and then click the button at the right edge of the column.	

com.webmethods.portal.system.IURI Property Editor

This property editor provides a Browse button that opens a universal picker instance for picking objects in the portal. See "com.webmethods.portal.service.meta2.thing.IThingID Property Editor" on page 209.

Double Property Editor

The Double property editor provides a one-line text field to contain a double (double-precision floating point real number). You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-blur Handler	A text string. Type a JavaScript the property editor should execute when focus is taken away from the input field.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
On-enter Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses ENTER.
On-focus Handler	A text string. Type a JavaScript the property editor should execute when focus is given to the input field.
On-key-press Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses a key.
On-select Handler	A text string. Type a JavaScript the property editor should execute when the input field content is selected.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression in the Validation Regular Expression property. The default message is: `Value must be a double`
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. The default regular expression is: `(^$)|(^-?\d+\.?\d+$)`

Float Property Editor

The Float property editor provides a one-line text field to contain a float (single-precision floating point real number). You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-blur Handler	A text string. Type a JavaScript the property editor should execute when focus is taken away from the input field.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
On-enter Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses ENTER.
On-focus Handler	A text string. Type a JavaScript the property editor should execute when focus is given to the input field.
On-key-press Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses a key.
On-select Handler	A text string. Type a JavaScript the property editor should execute when the input field content is selected.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression in the Validation Regular Expression property. The default message is: `Value must be a float`
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. The default regular expression is: (^$)\|(^-?\d+\.?\d+$)

Int Property Editor

The Int property editor provides a one-line text field to contain an integer. You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-blur Handler	A text string. Type a JavaScript the property editor should execute when focus is taken away from the input field.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
On-enter Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses ENTER.
On-focus Handler	A text string. Type a JavaScript the property editor should execute when focus is given to the input field.
On-key-press Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses a key.
On-select Handler	A text string. Type a JavaScript the property editor should execute when the input field content is selected.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression in the Validation Regular Expression property. The default message is: `Value must be an integer`
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. The default regular expression is: `(^$)|(^-?\d{1,10}$)`

java.lang.Boolean Property Editor

The Boolean Checkbox property editor provides a check box that passes a True or False state when the form is submitted. This property editor has the same properties as

"Boolean Checkbox Property Editor" on page 179 except that the CSS Class property has a default value of none.

java.lang.Byte Property Editor

The java.lang.Byte property editor provides a one-line text field to contain a single byte. See "Byte Property Editor" on page 207.

java.lang.Character Property Editor

The java.lang.Character property editor provides a one-line text field to contain a single character. See "Char Property Editor" on page 208.

java.lang.Double Property Editor

The java.lang.Double property editor provides a one-line text field to contain a double (double-precision floating point real number). See "Double Property Editor" on page 215.

java.lang.Float Property Editor

The java.lang.Float property editor provides a one-line text field to contain a float (single-precision floating point real number). See "Float Property Editor" on page 216.

java.lang.Integer Property Editor

The java.lang.Integer property editor provides a one-line text field to contain an integer. See "Int Property Editor" on page 217.

java.lang.Long Property Editor

The Long property editor provides a one-line text field to contain a long integer. See "Long Property Editor" on page 222.

java.lang.Object Property Editor

The java.lang.Object property editor provides a one-line text input field. See "Text Property Editor" on page 205.

java.lang.Short Property Editor

The Short property editor provides a one-line text field to contain a short integer. See "Short Property Editor" on page 223.

java.math.BigDecimal Property Editor

The java.math.BigDecimal property editor provides a one-line text field to contain an immutable, arbitrary-precision signed decimal number. You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-blur Handler	A text string. Type a JavaScript the property editor should execute when focus is taken away from the input field.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
On-enter Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses ENTER.
On-focus Handler	A text string. Type a JavaScript the property editor should execute when focus is given to the input field.
On-key-press Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses a key.
On-select Handler	A text string. Type a JavaScript the property editor should execute when the input field content is selected.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression in the Validation Regular Expression property. The default message is: `Value must be a decimal`
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. The default regular expression is: `(^$)|(^-?\d+\.?\d+$)`

java.math.BigInteger Property Editor

The java.math.BigInteger property editor provides a one-line text field to contain an immutable, arbitrary-precision integer. You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-blur Handler	A text string. Type a JavaScript the property editor should execute when focus is taken away from the input field.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
On-enter Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses ENTER.
On-focus Handler	A text string. Type a JavaScript the property editor should execute when focus is given to the input field.
On-key-press Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses a key.
On-select Handler	A text string. Type a JavaScript the property editor should execute when the input field content is selected.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression in the Validation Regular Expression property. The default message is: `Value must be an integer`
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. The default regular expression is: `(^$)\|(^-?\d+$)`

java.net.URL Property Editor

The java.net.URL property editor provides a one-line text field to contain a URL. You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-blur Handler	A text string. Type a JavaScript the property editor should execute when focus is taken away from the input field.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
On-enter Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses ENTER.
On-focus Handler	A text string. Type a JavaScript the property editor should execute when focus is given to the input field.
On-key-press Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses a key.
On-select Handler	A text string. Type a JavaScript the property editor should execute when the input field content is selected.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression in the Validation Regular Expression property. The default message is: `Value must be a URL`
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. The default regular expression is: `(^$)\|(^.+://.+$)`

java.util.Calendar Property Editor

The java.util.Calendar property editor provides a pop-up date and time editor. See "Date Property Editor" on page 184.

java.util.Date Property Editor

The java.util.Date property editor provides a pop-up date and time editor. See "Date Property Editor" on page 184.

Long Property Editor

The Long property editor provides a one-line text field to contain a long integer. You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-blur Handler	A text string. Type a JavaScript the property editor should execute when focus is taken away from the input field.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
On-enter Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses ENTER.
On-focus Handler	A text string. Type a JavaScript the property editor should execute when focus is given to the input field.
On-key-press Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses a key.
On-select Handler	A text string. Type a JavaScript the property editor should execute when the input field content is selected.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.

Property	Description
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression in the Validation Regular Expression property. The default message is: `Value must be a long`
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. The default regular expression is: `(^$)\|(^-?\d+\.?\d+$)`

Short Property Editor

The Short property editor provides a one-line text field to contain a short integer. You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-blur Handler	A text string. Type a JavaScript the property editor should execute when focus is taken away from the input field.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
On-enter Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses ENTER.
On-focus Handler	A text string. Type a JavaScript the property editor should execute when focus is given to the input field.
On-key-press Handler	A text string. Type a JavaScript the property editor should execute when the input field is focused and the user presses a key.
On-select Handler	A text string. Type a JavaScript the property editor should execute when the input field content is selected.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.

Property	Description
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression in the Validation Regular Expression property. The default message is: `Value must be a short`
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. The default regular expression is: `(^$)|(^-?\d{1,5}$)`

Search Drawer

The Search Drawer provides property editors to specify portal search capabilities.

- "Search Library Property Editor" on page 224
- "Search Service Property Editor" on page 225

Search Library Property Editor

The Search Library property editor provides a drop-down list of available libraries for the specified search service. You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Search Service Name	Type the name of a valid portal search service, such as luceneSearch.

Property	Description
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

Search Service Property Editor

The Search Service property editor provides a drop-down list of search services available on the portal. You can modify the following properties in the Properties view:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

wm_tabs Drawer

The wm_tabs Drawer provides property editors to tab properties.

- "Tab Style Property Editor" on page 226

Tab Style Property Editor

The Tab Style property editor allows you to select a preconfigured tab style to use with the wm_tabs portlet. For example, if you create a portlet that displays a list of categories, and use the wm_tabs portlet within that portlet to display the category list, you might expose a `mytabstyle` property on the category portlet, and pass the value of that property to the wm_tabs portlet Tab Style property. This capability allows a user to publish an instance of the portlet and use the Tab Style property editor to change the instance's display from a list of tabs to a list of links separated by bars (|), or to whatever styles have been configured for the wm_tabs portlet).

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

Table Drawer

The Table Drawer provides the ability to add tables to a form.

- "Single-Select Table Property Editor" on page 227
- "Table Property Editor" on page 228
- "Multiple-Select Table Property Editor" on page 234

Single-Select Table Property Editor

The Single-Select property editor provides a table that can contain multiple pages of radio buttons and titles. Only one radio button in the table can be selected at a time. If a radio button is selected when the user submits the form, the Values property value for that button is passed. Compare with "Radio Button Property Editor" on page 197 and "Radio Button Group Property Editor" on page 198.

The Titles property provides text for the titles; the Values property provides values to be passed. One radio button is displayed for each value in the Values property. If there is a mismatch in the number of values between the Titles and Values properties:

If there are more...	The property editor does this...
Titles property values	Uses only enough Titles values to equal the number of Values values.
Values property values	Uses the Values property value in place of missing titles.

You can modify the following properties in the Properties view:

Property	Description
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Titles	Provides a list of titles to display with radio buttons. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add entries in the order in which they should appear. If you leave this property empty, the default is to use values in the Values property.

Property	Description
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.
Values	Provides a list of values associated with the radio buttons. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add values in the order in which they should appear.

Table Property Editor

The Table property editor allows much finer-grained control of the table settings than do the Single-Select or Multi-Select Table property editors. In particular, you can specify the list of rows displayed by the Table as one of the following data types:

java.lang.Object[]
java.lang.String (where rows are separated by line-feeds)
java.sql.ResultSet
java.util.Collection
java.util.Enumeration
java.util.Iterator
org.w3c.dom.Node (where each row is a child of the specified node)
org.w3c.dom.NodeList
org.w3c.dom.traversal.NodeIterator

You can configure the list of columns displayed. You can specify the content of a column using {x} expression notation, where x is a bean property of the row. For example, with a row object that has getID() and getTitle() methods, you could use the ID or Title properties in an expression (like {id} or {title}). If the row object is one of the following special types, however, an expression {x} is interpreted differently:

com.webmethods.service.view.IView: named view property
java.lang.Object[]: array index
java.lang.String: (where columns are separated by commas) column index
java.sql.ResultSet: column index/name
java.util.Collection: collection index
java.util.Enumeration: enumeration index
java.util.Iterator: iterator index
java.util.Map: map key
org.w3c.dom.Node: xpath expression

For example, if a row object is an Object[], an expression {0} refers to the first item in the array, {1} to the second item, and so on. If the row object is a Node, an expression {@id} refers to the "id" attribute of that node and an expression {name} refers to the "name" element child of that node.

You must specify the list of rows that the table displays using one of these properties: List, List Command, Slice, or Slice Command.

You can modify the following properties in the Properties view:

Property	Description
Alignments	A list of table column alignments, one for each column. Acceptable values are "" (the current user browser's default text alignment), `left`, `center`, or `right`. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add entries in the order in which they should appear.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Empty Message	A text string to display if the table is empty. The default is Empty.
Links	A list of expressions to use for links, one for each column. (You can also create links using the Values property; these links wrap the main content of a column as specified by the Values property.) If the expression is blank, no link is created for that column. If the expression is not blank, a link is created for the column, with the expression's value as the link reference. For example, for a column with a link expression of {url}, where the url property of each row is http://www.google.com/, http://www.yahoo.com/, and so forth, the link for the column in the first row is http://www.google.com/, the second is http://www.yahoo.com/, and so forth.To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add entries in the order in which they should appear.

Property	Description
List	The full list of rows to be displayed (See also, List Command, Slice, and Slice Command). If you edit the JSP page with a Table property editor in it, you can specify the list as any of the following data types; but using the visual properties pane you can only specify the list as a CSV string (where rows are separated by line-feeds): java.lang.Object[] java.lang.String (where rows are separated by line-feeds) java.sql.ResultSet java.util.Collection java.util.Enumeration java.util.Iterator org.w3c.dom.Node (each row is a child of the specified node) org.w3c.dom.NodeList org.w3c.dom.traversal.NodeIterator To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add entries in the order in which they should appear.
List Command	The Portal command name that returns the full list of rows to be displayed (See also, List, Slice, and Slice Command).
Multiple Selections	A boolean value set to True if the property editor is to display a check box for each row in the table and False if the property editor is to display a radio button for each row.To display the choices, click the **Value** column and then click the button at the right edge of the column. The name of the check box or radio button is specified by the Name property, the value is specified by the Selection Key property.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Page Size	The number of items to be displayed per page. If blank, this property defaults to user's preferred number of items per page.
Parameters	Additional parameters used by the List Command or Slice Command property. For example, the listChildren command has custom includeItems and depth parameters. To list the containers in a folder three levels deep you might specify the Parameters property as this: `includeItems=false&depth=3`
Portlet	ID of portlet that manages this table. Only in exceptional circumstances would you not leave this blank.

Property	Description
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Resource	If the command specified in the List Command or Slice Command property operates on a particular portal resource, specify the ID for that resource in the Resource property. For example, to list the children of the System folder, the entry in the Resource might be this: `folder.system`
Selection Key	An expression used to calculate the selection value of a row. For example, an expression {id} would use the ID property of a row as its selection value.
Show Column Headers	A boolean value set to True (the default) if the table should display column headers and False if column headers are to be hidden. To display the choices, click the **Value** column and then click the button at the right edge of the column. Use the Titles property to display the column header values.
Show Empty	A boolean value set to True (the default) if the table should display an Empty message if the table is empty, and False if a message should not be displayed. To display the choices, click the **Value** column and then click the button at the right edge of the column. Use the Empty Message property to contain the text of the message.
Show Footer	A boolean value set to True (the default) if the table should display footer with paging information and False if the footer should not be displayed. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Show Page Links	A boolean value set to True (the default) if the table should display links to individual pages of the table and False if the links should not be displayed. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Show Paging Header	A boolean value set to True (the default) if the table should display paging information above the table proper and False if the paging information should not be displayed. To display the choices, click the **Value** column and then click the button at the right edge of the column.

Property	Description
Show Total	A boolean value set to True (the default) if the table should display the total number of items in the table and False if the total number of items should not be displayed. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Show Total Selected	A boolean value set to True (the default) if the table should display the total number of items selected and False if the total number of items selected should not be displayed. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Slice	The list of rows to be displayed for the current page only. For example, if the table contains 1000 rows, but only rows 21 through 30 are currently displayed, the Slice property should contain the list of rows 21 through 30 only (See also, List, List Command, and Slice Command). To display the List Editor, click the Value column and then click the button at the right edge of the column. Add entries in the order in which they should appear.
Slice Command	The Portal command name that returns the list of rows to be displayed for the current page only (See also, List, List Command, and Slice Command).
Sort Key	The default sort key. See the Sort Keys property.
Sort Keys	A list of sort keys, one for each column. If the value is blank or unspecified for a particular column, the user cannot re-sort on that column. A sort key for a particular column typically matches the value expression (both might be {id}), but not necessarily so (the value might be {formattedDate} and the sort key might be {rawDate}). To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add entries in the order in which they should appear.
Sort Order	The default sort order, either ascending (the default) or descending. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Start Index	The default start index. The index of the first row is 1.
Table Style	The display style for the table. The Standard style (the default) is the table seen in most Portal tables. The Condensed style has less padding or no padding around each cell and the Table style displays a border along each cell edge. The actual appearance of the table style is dependent on the Portal skin in use. To display the choices, click the **Value** column and then click the button at the right edge of the column.

Property	Description
Titles	Provides a list of column headers to display. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add entries in the order in which they should appear. If you leave this property empty, the default is to use values in the Values property.
Tooltips	A list of expressions to use for ToolTips (hover text) for the cells in each column, one for each column. If the ToolTip entry for a column is left blank, the cells in the column will not have ToolTips.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.
Values	A list of expressions to use for the cell content, one for each column. For example, for a column with a value expression of {name}, where the "name" property of each row is Google, Yahoo, and so forth, the text displayed in the first row of the column is Google, the second is Yahoo, and so forth. If the value expression is <img src="{url}logo.gif" width="16" height="16" /> {name}, the text displayed for each column includes an icon in front of it. Any HTML in the property value is automatically escaped. To display raw HTML, add a caret symbol (^) in front of the property name. For example, to display an icon in some column for each row, where the "icon" property is "<img src='http://www.google.com/logo.gif' />", "<img src='http://www.yahoo.com/logo.gif' />",and so forth, you would use a value expression of {^icon}. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add values in the order in which they should appear.
View	If you use more than one table on the same page of a portlet, you must give each table a unique View property value. This value can be brief because it needs to be unique only on a particular portlet page. For example, the View property value of one table could be `one` and the value of the second table could be `two`.

Property	Description
Width	A list of table column widths, one for each column. Values can be a percentage (such as 50%), a pixel size (such as 20px) or left blank (which allows the browser to size the column). To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add values in the order in which they should appear.
Wrappings	A list of table wrapping values, one for each column. Acceptable values are to leave the entry blank (allow column content to wrap) or `nowrap` (do not allow column content to wrap). To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add values in the order in which they should appear.

Multiple-Select Table Property Editor

The Multiple-Select Table property editor provides a table that can contain multiple pages of check boxes and titles. If a list item is selected when the user submits the form, the Values property value is passed. Compare with "Checkbox Group Property Editor" on page 182 and "Multi-Select Property Editor" on page 191.

The Titles property provides text for the titles; the Values property provides values to be passed. One list item is displayed for each value in the Values property. If there is a mismatch in the number of values between the Titles and Values properties:

If there are more...	The property editor does this...
Titles property values	Uses only enough Titles values to equal the number of Values values.
Values property values	Uses the Values property value in place of missing titles.

You can modify the following properties in the Properties view:

Property	Description
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.

Property	Description
Titles	Provides a list of titles to display with check boxes. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add entries in the order in which they should appear. If you leave this property empty, the default is to use values in the Values property.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.
Values	Provides a list of values associated with the check boxes. To display the List Editor, click the **Value** column and then click the button at the right edge of the column. Add values in the order in which they should appear.

Portlet Properties Drawer

The Portlet Property Drawer provides property editors to display and label portlet properties.

- "Portlet Property Label Property Editor" on page 235
- "Portlet Property Value Property Editor" on page 236

Portlet Property Label Property Editor

The Portlet Property Label property editor provides the display name of a portlet property as a label. Use with "Portlet Property Value Property Editor" on page 236. This property editor uses these properties:

Property	Description
Name	Type the internal name of the portlet property.
Portlet URI	Type the URI for the portlet, as deployed on the Portal server. Use the default portlet instance alias, which takes the form /portlet/*portlet_name*.

Portlet Property Value Property Editor

The Portlet Property Value property editor displays a portlet property. Use with "Portlet Property Label Property Editor" on page 235. As an example of how these two property editors work, select an existing property from the **wmPortal Components** tab for the current portlet and drag it into a table cell in the Form Builder. This property editor uses the CSS Class property and the following properties:

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Name	Type the internal name of the portlet property to be displayed.
Portlet URI	Type the URI for the portlet, as deployed on the Portal server. Use the default portlet instance alias, which takes the form /portlet/*portlet_name*.

Internationalization Drawer

The Internationalization Drawer provides property editors for use in internationalizing forms.

- "Country/Region ID Chooser Property Editor" on page 236
- "Locale Chooser Property Editor" on page 237

Country/Region ID Chooser Property Editor

The Country/Region ID Chooser property editor provides a drop-down list from which a user can select a country or region. You can modify the following properties in the Properties view.

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

Locale Chooser Property Editor

The Local Chooser property editor provides a drop-down list from which a user can select a locale. Language packs that are currently installed on the Portal server are highlighted in blue. You can modify the following properties in the Properties view.

Property	Description
CSS Class	The CSS class to be applied to the output of the property editor. To display the list, click the **Value** column and then click the button at the right edge of the column. Valid list items are number, login, small, medium (the default), and large.
Disabled	A boolean value set to True to disable the property editor and False (the default) to enable it. To display the choices, click the **Value** column and then click the button at the right edge of the column.

Property	Description
Language Packs	A boolean value set to True to limit the list to language packs that are currently installed and False (the default) to display all language packs. To display the choices, click the **Value** column and then click the button at the right edge of the column.
On-Change Handler	A JavaScript the property editor should execute if its value changes. Type the JavaScript in the **Value** column.
Read Only	A boolean value set to True if the property editor should be read only and False (the default) if the editor can be written to. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Show Default	A boolean value set to True if the property editor displays **Default** as the default setting for the drop-down list and False if it displays the first locale value by default. The default is True. To display the choices, click the **Value** column and then click the button at the right edge of the column.
Validation Failure Message	An error message that is displayed when a user types an input that fails validation by the regular expression you have created in the Validation Regular Expression property. Type the message in the **Value** column.
Validation Regular Expression	A regular expression against which to validate inputs to the property editor. Click the **Value** column and then click the button at the right edge of the column to display a Regular Expression Editor that can help you create and test a regular expression.

APPENDIX B

Building Portlets from the Command Line

Using Apache ANT to Build Portlets

Important! The utility and examples shown here are not official parts of the webMethods product line. The utility and examples are provided strictly "as is," and are not eligible for technical support through webMethods Technical Services. webMethods makes no guarantees or warranties pertaining to the functionality, scalability, robustness, or degree of testing of the utility and examples, and assumes absolutely no liability for any damages relating to them. Customers are strongly advised to consider the utility and examples as "working examples" from which they should build and test their own solutions.

Apache Ant is a Java-based build tool that is used within the Portlet Developer to build portlets. You can also use Ant to build portlets from the command line. To build a portlet in this way, you need to place the directory containing the portlet source files into the /mycomponents directory, which has the following location:

webMethods_install_dir/Portal/mycomponents

You can download Apache Ant, and find documentation on how to use it, at http://ant.apache.org.

\

To set up a build for portlets

1. Install and configure Apache Ant on the machine where you intend to build portlets.
2. Create a new system variable:

   ```
   ANT_HOME=ant_install_dir
   ```

   where *ant_install_dir* is the directory where you have installed Ant.
3. Update the path variable to include this entry:

   ```
   PATH=%PATH%;%ANT_HOME%/bin
   ```
4. (Optional) Copy or move the /mycomponents directory to a location within your project source tree.

5 If you move the directory in step 4, make the following changes to the build.properties in that same directory.

Change this line...	In this way...

`portal.home=..`	To point to the location *`webMethods_install_dir`*`/Portal`. For example, on Windows, the line might become:
	`portal.home=C:\\Program Files\\`
	`webMethods6\\Portal`

`dist.dir=${portal.home}`	To point to a directory into which the finished portlets should be written. For example, on Windows, the line might become: `dist.dir=C:\\projects\\portal`

6 (Optional) Create category directories in the /mycomponent directory.

For example, you might create the following directory structure for your portlets, shells, and skins:

```
c:/projects/portal/mycomponents
    /portletscategory_1
        /portletssubcategory_1
            /my_portlet_1
            /my_portlet_2
        /portletssubcategory_2
            /my_portlet_3
    /portletscategory_2
        /my_shell_1
        /my_skin_1
        /my_pages_1
```

7 Place the components into the directory structure of the /mycomponents directory.

8 If you have not already done so, move to the /mycomponents directory.

9 Type the following command:

```
ant |[dist]|[clean]|
```

where:

This option...	Does this...
	(No option) Builds all components in place within the directory structure of the mycomponents directory.

This option...	Does this...
`dist`	Builds all components and places them within the *dist.dir*/components directory.
`clean`	Cleans the component source and build folders, removing generated .class and packaged component files (.pdp, .cdp, .war, .skin, .lp, .dccb).

Examples

You can also build a category or a single component if you specify the category path or the component home directory path as the *basedir* property:

To build just portletssubcategory2 components (from the sample structure in step 6), the command might be:

```
ant -Dbasedir=portletscategory1/portletssubcategory2
```

To clean a simple my_portlet3 component, the command might be:

```
ant -Dbasedir=porteltscategory1/porteltssubcategory2/my_portlet3 clean
```

To incorporate the component build into your product build, use an ant call with this syntax:

```
<target name="components">
   <ant dir="${<mycomponents path>}" target="dist" inheritAll="false"
   inheritRefs="false"/>
</target>
```

An example of this syntax is as follows:

```
<target name="components">
   <ant dir="c:/projects/portal/mycomponents" target="dist"
   inheritAll="false" inheritRefs="false">
       <property name="dist.dir" value="c:/projects/portal/dist"/>
   </ant>
</target>
```

Index