Novell exteNd Director

5.0

www.novell.com

PORTAL GUIDE

Novell®

Legal Notices

Novell, Inc.

1800 South Novell Place

Provo, UT 85606

www.novell.com

exteNd Director *Portal Guide*

December 2003

Online Documentation: To access the online documemntation for this and other Novell products, and to get updates, see www.novell.com/documentation.

Novell Trademarks

ConsoleOne is a registered trademark of Novell, Inc.
eDirectory is a trademark of Novell, Inc.
GroupWise is a registered trademark of Novell, Inc.
exteNd is a trademark of Novell, Inc.
exteNd Composer is a trademark of Novell, Inc.
exteNd Director is a trademark of Novell, Inc.
iChain is a registered trademark of Novell, Inc.
jBroker is a trademark of Novell, Inc.
NetWare is a registered trademark of Novell, Inc.
Novell is a registered trademark of Novell, Inc.
Novell eGuide is a trademark of Novell, Inc.

SilverStream Trademarks

SilverStream is a registered trademark of SilverStream Software, LLC.

Third-Party Trademarks

Acrobat, Adaptive Server, Adobe, AIX, Autonomy, BEA, Cloudscape, DRE, Dreamweaver, EJB, HP-UX, IBM, Informix, iPlanet, JASS, Java, JavaBeans, JavaMail, JavaServer Pages, JDBC, JNDI, JSP, J2EE, Linux, Macromedia, Microsoft, MySQL, Navigator, Netscape, Netscape Certificate Server, Netscape Directory Server, Oracle, PowerPoint, RSA, RSS, SPARC, SQL, SQL Server, Sun, Sybase, Symantec, UNIX, VeriSign, Windows, Windows NT

All third-party trademarks are the property of their respective owners.

Third-Party Software Legal Notices

The Apache Software License, Version 1.1

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)."

Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Apache" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.

5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Autonomy

Castor

Redistribution and use of this software and associated documentation ("Software"), with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain copyright statements and notices. Redistributions must also contain a copy of this document.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name "ExoLab" must not be used to endorse or promote products derived from this Software without prior written permission of Intalio Inc. For written permission, please contact info@exolab.org.

4. Products derived from this Software may not be called "Castor" nor may "Castor" appear in their names without prior written permission of Intalio Inc. Exolab, Castor and Intalio are trademarks of Intalio Inc.

5. Due credit should be given to the ExoLab Project (http://www.exolab.org/).

THIS SOFTWARE IS PROVIDED BY INTALIO AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTALIO OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Indiana University Extreme! Lab Software License

Version 1.1.1

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Indiana University Extreme! Lab (http://www.extreme.indiana.edu/)."

Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Indiana University" and "Indiana University Extreme! Lab" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact http://www.extreme.indiana.edu/.

5. Products derived from this software may not use "Indiana University" name nor may "Indiana University" appear in their name, without prior written permission of the Indiana University.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS, COPYRIGHT HOLDERS OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

JDOM.JAR

Phaos

Sun

W3C

3. Notice of any changes or modifications to the files, including the date changes were made. (We recommend you provide URIs to the location from which the code is derived.)

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR DOCUMENTATION.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to the software without specific, written prior permission. Title to copyright in this software and any associated documentation will at all times remain with copyright holders.

Contents

About This Book

Purpose

This book explains how use the Novell® exteNd Director™ Portal subsystem and the exteNd Director implementation of Java Portlet 1.0.

Audience

This book is for programmers working on exteNd Director applications. It assumes familiarity with Java programming, HTML, XML, and XSL and provides examples using all these technologies.

Portal Concepts

Provides an overview of portal concepts

1 About Portal Applications

This chapter provides an introduction to exteNd Director portal applications. It contains the following sections:

- About the Portal subsystem
- Anatomy of a portal application
- Portal application resources
- Creating a portal application project

 For more information about exteNd Director projects in general, see the chapters that cover working with projects in *Developing exteNd Director Applications*.

About the Portal subsystem

exteNd Director applications often include a **portal** Web site. The portal is the presentation layer for an application, providing the interface through which users access and interact with Web content generated by portlets.

To implement this functionality, exteNd Director provides a Portal subsystem that allows you to:

- Design and run Java Portlet 1.0-compliant portlets
- Create, maintain, and manage portal pages
- Manage access to portal content

The Portal subsystem also provides complete integration with the Directory, Security, and User subsystems, which handle authentication, authorization, and user profiling.

Portal components are deprecated exteNd Director provides runtime support for portal components created in earlier versions of the product. However, you should use portlet technology for all new development.

Web presentation services

The Portal subsystem implements the following Web presentation services:

Portal service	Description
Portal Aggregator	Renders a response to a portal request by: ◆ Displaying the content of a portlet directly in a Web client ◆ Aggregating the content of one or more portlets on a portal page
Page Manager	Manages page information and page categories for PID pages
Portal Personalizer	Provides a graphical interface for creating personal pages and applying themes portal-wide 📖 For more information, see the chapter on personalizing your portal.
Portal Administration tool	Provides a graphical interface for portal administrators to create container and shared pages, assign permissions for page access, and manage the Portal 📖 For more information, see the chapter on administering portal pages.

Portal APIs

The Portal subsystem provides APIs for:

- Developing Java Portlet 1.0-compliant portlets that generate content in portal pages—or dispatch content generation to JSP pages or servlets
- Working with page layouts, as described in the section on working with the layout API.
- Implementing portlet-level decorators and decorator services, as described in the chapter on working with portal decorators.
- Working with portal options, as described in the section on working with the portal option API.
- Working with portal themes, as described in the section on working with the theme API.
- Working with portal categories, as described in the chapter on working with page categories

Anatomy of a portal application

This section describes the key functional elements of exteNd Director portal applications.

Portal aggregation servlet

Each exteNd Director portal application includes a base servlet called **EboPortalAggregationServlet** that controls all portal and portlet requests.

This servlet listens on the key **portal** in the request URL, as specified by the **PortalPathEntryPointKey** descriptor in web.xml:

```
<context-param>
    <param-name>PortalPathEntryPointKey</param-name>
    <param-value>portal</param-value>
</context-param>
```

Application requests

This servlet responds to the following kinds of application requests:

Type of Request	Request URL
Run a portlet directly in a browser	Syntax *servlet_url*/*portlet_path*/*name of portlet* *servlet_url* http://*host*/*context*/portal *portlet_path* portlet Example: http://localhost/ExpressPortal/portal/portlet/WelcomeMessagePortlet
Run a component directly in a browser	Syntax *servlet_url*/*comp_path*/*name of component* *servlet_url* http://*host*/*context*/portal *comp_path* comp Example: http://localhost/ExpressPortal/portal/comp/MyComponent

Type of Request	Request URL
Run a component directly in a browser with decoration	Syntax *servlet_url*/*component_path*/*name of component* *servlet_url* http://*host*/*context*/portal *component_path* component Example: http://localhost/ExpressPortal/portal/component/MyComponent
Run the Portal Personalizer	Syntax *servlet_url*/*portlet_path*/Personalize *servlet_url* http://*host*/*context*/portal *portlet_path* portlet Example: http://localhost/ExpressPortal/portal/portlet/Personalize
Run the Portal Administrator	Syntax *servlet_url*/*portlet_path*/PortalPageAdmin *servlet_url* http://*host*/*context*/portal *portlet_path* portlet Example: http://localhost/ExpressPortal/portal/portlet/PortalPageAdmin
Display a container page that displays the content of a specified shared page	Syntax *servlet_url*/cn/*container page name*/*name of shared page* *servlet_url* http://*host*/*context*/portal Example: http://localhost/ExpressPortal/portal/cn/MyContainerPage/OurSharedPage

Type of Request	Request URL
Display a container page that displays the content of a specified personal page	Syntax *servlet_url*/cn/*container page name*/up_*name of personal page* *servlet_url* http://*host*/*context*/portal Example: http://localhost/ExpressPortal/portal/cn/MyContainerPage/up_MyPersonalPage
Display a container page that displays the content of its default shared page	Syntax *servlet_url*/cn/*container page name* *servlet_url* http://*host*/*context*/portal Example: http://localhost/ExpressPortal/portal/cn/MyContainerPage
Display a shared page	Syntax *servlet_url*/pg/*shared page name* *servlet_url* http://*host*/*context*/portal Example: http://localhost/ExpressPortal/portal/pg/OurSharedPage
Display a shared page surrounded by the user's default container page	Syntax *servlet_url*/pgcn/*shared page name* *servlet_url* http://*host*/*context*/portal Example: http://localhost/ExpressPortal/portal/pgcn/OurSharedPage
Display a personal page	Syntax *servlet_url*/pg/up_*personal page name* *servlet_url* http://*host*/*context*/portal Example: http://localhost/ExpressPortal/portal/pg/up_MyPersonalPage

Type of Request	Request URL
Display a personal page surrounded by the user's default container page	Syntax *servlet_url*/pgcn/up_*personal page name* *servlet_url* http://*host*/*context*/portal Example: http://localhost/ExpressPortal/portal/pgcn/up_MyPersonalPage
Display a PID page	Syntax *servlet_url*/*page_path*/MyPID.html *servlet_url* http://*host*/*context*/portal *page_path* pages Example: http://localhost/ExpressPortal/portal/pages/MyPID.html

Resource servlet

Each exteNd Director portal application also includes a servlet called **resource**. This servlet can serve up any resource in the resource set associated with the portal application. The portal resource path is defined in **web.xml** as follows:

```
<context-param>
    <param-name>PortalResourcePath</param-name>
    <param-value>$CONTEXT_URL$/resource</param-value>
    <description>
        Path at which resource set resources are served up from the
         controller.
    </description>
</context-param>
```

For example, the following URL could be used to access an image file **MyImage.jpg** in the resource set for an exteNd Director WAR project called **MyWAR** running on the server **localhost**:

```
http://localhost/MyWAR/resource/images/MyImage.jpg
```

Portal Web tier

Portal applications can include the Portal Web tier, which provides the functionality that allows end-users and portal administrators to interact with the portal. The Portal Web tier includes tools for logging in, customizing portal pages, and creating new users. It can be the springboard for your own applications.

exteNd Director ships with a portal application called Express Portal which includes a Portal Web tier.

 For more information about Express Portal, see Chapter 2, "About the Express Portal".

 To learn how end-users can work with the Portal Web tier, see the chapter on personalizing your portal.

 To learn how portal administrators can use the Portal Web tier, see the chapter on administering the portal.

Resource set

Portal applications can include an exteNd Director resource set, which organizes resources used by exteNd Director subsystems, including:

- Descriptors for portlets, pages, pageflows, workflows, forms, and rules
- Images
- WSDL files
- XSL style sheets
- HTML pages

A key feature of the resource set is that it enables dynamic loading of these resources during development, obviating the need for frequent redeployments and speeding the testing cycle.

 To learn more about resource sets, see the sections on managing application resources in *Developing exteNd Director Applications*.

Shared libraries

Portal applications can use shared libraries, which contain JAR files and classes that encapsulate exteNd Director services. In a shared library environment, these JARs and classes are installed on your application server so they can be shared across all Web applications deployed on that server. It is important to note, however, that only one portal application can be deployed to the application server in a shared library environment.

 For more information about shared libraries, see the section about shared library configurations in *Developing exteNd Director Applications*.

Portal application resources

An exteNd Director portal application can contain a combination of standard J2EE resources along with value-added resources that extend functionality.

J2EE resources The J2EE resources include:

- Servlets
- JavaServer Pages (JSP pages)
- Java utility classes
- Custom tag libraries

Value-added resources The exteNd Director value-added resources include:

Resource	Description
Portlets	Java classes that generate dynamic content on portal pages and PID pages. Portlets are governed by the Java Portlet 1.0 specification. When a page is requested at runtime, the Portal Aggregator aggregates and renders the content of all portlets on the page.

Resource	Description
Portal pages	Customizable pages whose content is generated by portlets. exteNd Director provides graphical design tools that allow portal administrators and end users to create portal pages. Portal applications can include three types of portal pages: ◆ Personal pages ◆ Shared pages ◆ Container pages **NOTE:** Unlike the other resources listed in this table, portal pages are not defined at design time within the exteNd Director development environment. Instead, they are created at runtime within a running Portal application. 📖 For more information on portal pages, see the chapter on working with portal pages.
PID pages	HTML or XML pages that contain exteNd Director portlets and components (defined by s3-component tags) and therefore require processing by the Portal Aggregator.
Static pages	HTML pages that contain static content.
Style sheets (XSL)	Style sheets that transform XML pages into HTML (or another output format). XSL is reusable for any page that uses the same XML elements.
Themes	Visual characteristics that apply globally to an entire exteNd Director portal application. These settings can potentially change the appearance of portal pages, PID pages, JSP pages, and portlets. Themes provide a simple way to ensure a consistent appearance throughout a portal application.
Layouts	Templates that define how a set of selected portlets should appear on a page. Each personal page, shared page, or container page in an exteNd Director portal application uses a portal layout to specify how the selected portlets and components should be arranged on the page.
Decorators	Java classes that decorate the dynamic content generated by a portlet. The decorator controls the appearance of the title bar, body, and footer for the portlet. The portlet data (the dynamic content) is inserted in the body of the portlet.

Resource	Description
Portal options	Images or text links that appear in the title bar of a portlet. Users interact with these controls to change portlet behavior at runtime. exteNd Director ships with several predefined portal options: ◆ Reload Pageflow ◆ Help ◆ Edit ◆ Minimize ◆ Maximize ◆ Restore ◆ Remove You can also add your own custom options.
Media	Images, sounds, and other media files required for the application.

Creating a portal application project

To create a project that includes a portal, see the chapter on creating exteNd Director projects in *Developing exteNd Director Applications.*

2 About the Express Portal

This chapter provides an introduction to the Express Portal. It contains the following sections:

What is Express Portal?

To give you a head start in building applications, exteNd Director provides a default project called Express Portal, which is a complete Portal application based on the standard exteNd Director project template. You can use this application out of the box to:

- Explore the features of an exteNd Director portal
- Learn how to build a portal application
- Build your own customized production-quality portal by adding and modifying portlets and pages, and customizing the presentation layer

How is Express Portal installed?

The Express Portal is available in several modalities, depending on how you install the exteNd™ product suite:

Type of exteNd suite installation	Project available on file system	Application deployed to exteNd Application Server
Express		
Custom		
Server		
Tools		

About the Express Portal project

The Express Portal project is a standalone exteNd Director WAR project that encapsulates the custom logic and resources of the Express Portal application. This project is based on the **Standard Director Template**, which is installed at:

```
exteNd suite install directory\Director\templates\Director
```

 For more information about project templates, see the chapters that cover working with projects in *Developing exteNd Director Applications*.

If you install the exteNd suite using the Express or Custom install, the Express Portal project is added to your suite install directory. You can then open the project in exteNd Director and customize it with your own business logic—for example, by adding new portlets, pageflows, forms, and other Web resources such as JSPs and servlets.

 To access the Express Portal project in exteNd Director, see "Opening the Express Portal project" on page 35.

Express Portal project specifications

The Express Portal project has the following specifications:

Specification	Express Portal
Type of project	exteNd Director Project WAR that includes: ◆ A portal ◆ A portlet runtime ◆ A resource set 📖 For more information, see the chapter on using a resource set in an exteNd Director application.
Name of project	ExpressPortal
Directory location	*exteNd suite install directory*\Projects\ExpressPortal
exteNd Director database	*exteNd suite install directory*\MySQL\data\expressportal
JDBC connection pool	JDBC/ExpressPortal
Server profile	Novell exteNd 5_x
Deployment plan	ExpressPortal_DeplPlan.xml
Resource set	ExpressPortal-resource
Uses shared libraries?	No 📖 For more information, see the section on shared library configurations.

Specification	Express Portal
Subsystems included	All: ◆ CMS Administration Console (Web tier) ◆ Content Management ◆ Director Administration Console (Web tier) ◆ Directory ◆ Framework ◆ Portal (Web tier) ◆ Rule ◆ Search ◆ User ◆ WebDAV ◆ Workflow 📖 For more information about exteNd Director subsystems, see the overview chapter in *Developing exteNd Director Applications* .

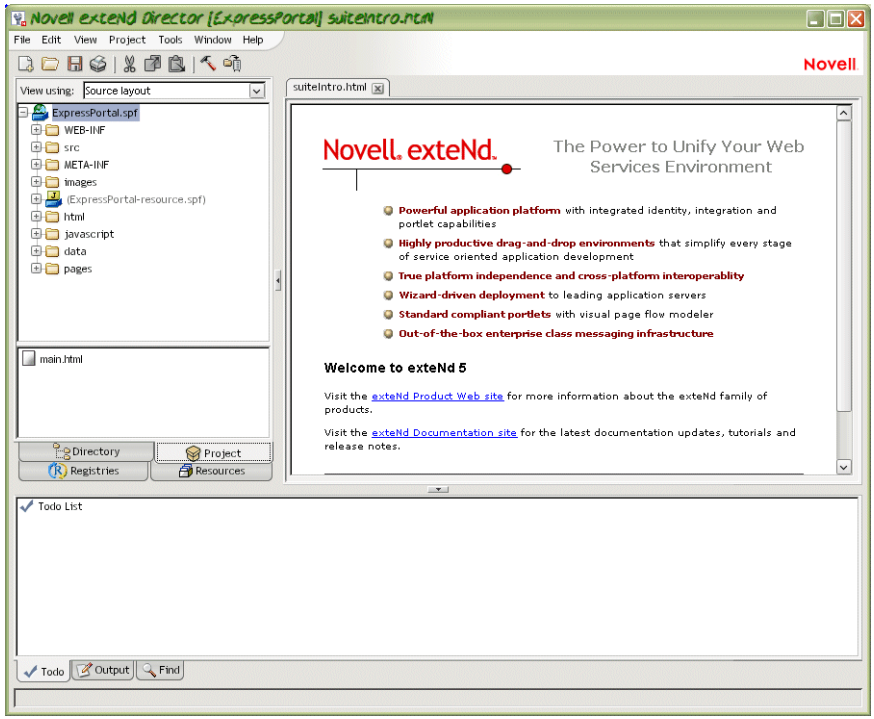
Opening the Express Portal project

If you installed the exteNd suite using the **Express** or **Custom** install option, you can open the ExpressPortal project in exteNd Director.

➢ **To open the Express Portal project:**

1 Start exteNd Director from the program group in the Windows **Start Menu** by selecting **Novell exteNd 5.0>Director>Director Designer**.

The first time you start exteNd Director, it opens the ExpressPortal project automatically on your desktop:



2 If the ExpressPortal project does **not** open automatically, follow these steps:

2a Select **File > Open Project**.

The Open Project dialog appears.

2b Navigate to the Express Portal project:

exteNd suite install directory\Projects\ExpressPortal

For example, if you installed the exteNd suite in c:\Program Files\Novell\exteNd5, you would navigate to:

c:\Program Files\Novell\exteNd5\Projects\ExpressPortal

2c Select **ExpressPortal.spf** and click **Open**.

The Express Portal project opens in exteNd Director.

Using the Express Portal project with other projects

In a shared library environment, only one portal application can be deployed to the application server at a time. Because the Express Portal project does **not** use shared libraries, you can deploy other portal applications to the server where Express Portal is running.

NOTE: You must use a shared library environment if you plan on using Portlet Application WARs.

What you can do with the Express Portal project

You can use the Express Portal project in several ways:

- **As a model portal application. in exteNd Director**. By exploring the structure and function of the Express Portal project, you can learn by example how to develop your own portal applications in exteNd Director.
- **As a starting point for a developing a production-quality application**. You can customize the Express Portal project by adding your own portlets, pages, and other Web application resources.

Changing the application context If you want to use the ExpressPortal project as the starting point for developing your own application, you need to change the application context. To do this, you need to replace the string ExpressPortal with the context name for your application. You need to do this in several places:

- You need to set the name of the **deployment context**. This varies from one application server to another.

  TIP: For the exteNd Application Server, you need to set the name of the **deployedObject** and **url** in the deployment plan. You can optionally change the **warJarName** as well, but if you do this, you must also change the Archive File Path in the Project Settings for your project.
- In the **web.xml** file for the project, you need to change the **display-name** element.
- In the **config.xml** file for the Portal subsystem, you need to change the value of the **portal.context** property.
- In the **config.xml** file in WEB-INF/conf, you need to change the name of the key for the resource set (**ExpressPortal/resourceset**) to specify your context rather than ExpressPortal. You do not need to change the value for this key.

These properties must all have matching values for the context name.

To ensure that your application runs as expected, you should use a new database for **exteNd Director** data.

📖 To get started, see the chapters on working with projects in *Developing exteNd Director Applications*.

About the Express Portal application

The Express Portal application is a working portal that you can use as soon as you install exteNd Director. It provides the full complement of functions contained in the exteNd Director Portal subsystem.

If you installed the exteNd suite using the **Express** install option, the Express Portal is deployed to the exteNd Application Server at installation time and ready to run. To get started, see "Starting the Express Portal application" on page 37.

If you installed the exteNd suite using the **Custom** install option, you need to deploy the Express Portal application to your exteNd Application Server before you can run it. See "Deploying the Express Portal" on page 41.

Starting the Express Portal application

The following procedure assumes that the Express Portal has been deployed to your exteNd Application Server. If you need to deploy the Express Portal, see "Deploying the Express Portal" on page 41.

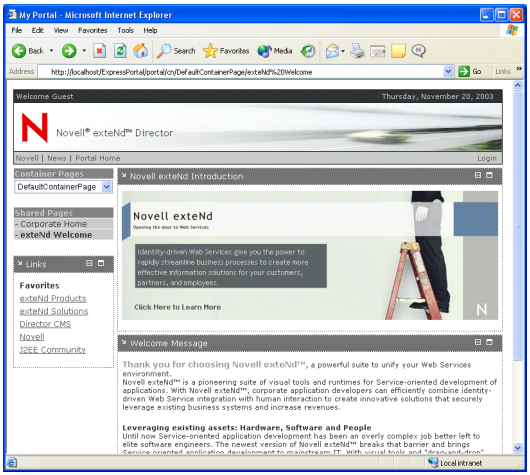
➢ **To start the Express Portal application:**

1 Start your server.

2 Open a browser and enter the following URL:

```
http://server name/ExpressPortal/portal
```

The default portal page opens in your browser, allowing you to enter the portal Web tier as a **guest** user:



Guest users have limited access to the exteNd Director portal. To take full advantage of its features—for example, to personalize the portal as an end-user or administrator—you must log in as an authorized user.

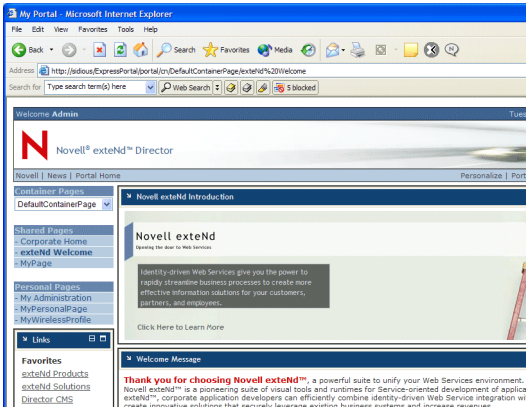
3 Click **Login** in the upper right corner of the page.

The exteNd Director Login portlet opens in your browser and prompts you to log in to the portal:

4 Enter your user name and password, then click **Login** or press the **Enter** key.

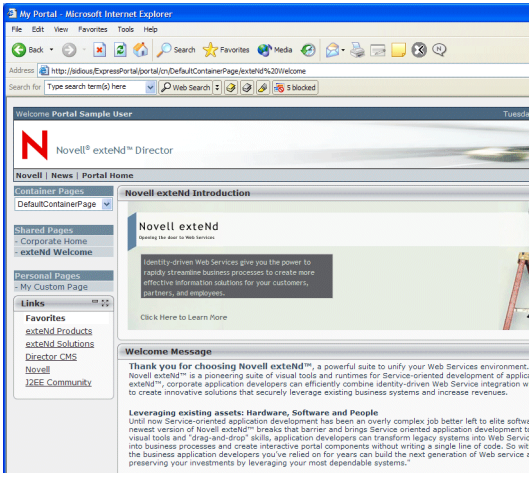
The default Express Portal page opens in your browser.

If you are a portal administrator (member of the **PortalAdmin** group with **PROTECT** permission) or a locksmith user, you'll see a default portal page with content tailored to administrators:



To learn about portal administration capabilities, see the chapter on administering the portal.

If you are any other type of user, you'll see a default portal page with content designed for users who do not have administrative privileges:



To learn how to customize your portal environment, see the chapter on personalizing your portal.

Now you are ready to work with the Express Portal application. To get started, see "What you can do with the Express Portal application" on page 40.

What happens when your session times out If you've logged in to the Portal, the portal tracks your session and notifies you if the session has timed out. When your session times out, the Portal redirects you to a session timeout page (which is rendered by the SessionHandler portlet). This page provides a link that allows you to return to the previous page with a new session.

NOTE: The web.xml file for the portal application includes a context parameter called UseSessionHandlerPortlet that allows you to control the behavior of session timeouts. If you do not want the SessionHandler portlet to manage timeouts, you can set this parameter to false. The default setting is true.

What you can do with the Express Portal application

Different levels of access are available for the Express Portal application. All users can work with Express Portal to personalize their private portal environments; administrative functions are more restrictive, available only to portal administrators and shared page owners.

A portal administrator is a user who has been assigned to the PortalAdmin administrative group with PROTECT permissions. The portal administrator can assign users to be owners of shared pages.

The following table summarizes how different types of users can work with Express Portal:

Task	Portal administrator	Shared page owner	All users	How?
Create and modify Personal Pages				1 Select **Personalize** 2 Follow procedures in chapter on personalizing your portal.
Create and modify Container Pages				1 Select **Portal Administration** 2 Follow procedures in the section on creating and maintaining container pages.
Create Shared Pages				1 Select **Portal Administration** 2 Follow procedures in the section on creating shared pages.

Task	Portal administrator	Shared page owner	All users	How?
Modify Shared Pages		 **NOTE:** These users can modify only the shared pages that they own		1 Select **Portal Administration** 2 Follow procedures in the sections on adding content to a shared page, modifying the layout of a shared page, and arranging content on the shared page.
Assign pages to users and groups				1 Select **Portal Administration** 2 Follow procedures in the section on assigning pages to users and groups.
Assign shared page owners				1 Select **Portal Administration** 2 Follow procedures in the section on assigning shared page owners.
Apply a theme to your portal				1 Select **Personalize** 2 Follow procedures in section on setting the theme for your portal

Deploying the Express Portal

You deploy the Express Portal the same way you deploy other exteNd Director applications, as described in the chapter on deploying exteNd Director applications in *Developing exteNd Director Applications*.

IMPORTANT: In a shared library environment, only one portal application can be deployed to the application server. Because the Express Portal application does **NOT** use shared libraries, it can coexist with other applications on your server.

 For more information on configuring a shared library environment, see the chapter on reconfiguring exteNd Director projects in *Developing exteNd Director Applications*.

Undeploying the Express Portal

You undeploy the Express Portal the same way you undeploy other exteNd Director applications.

3 Working with Portal Pages

This chapter explains how to use portal pages to personalize content in your portal applications. It covers the following topics:

About portal pages

Portal pages are customizable pages that you can include in Portal applications to present information tailored to your end-user audience. exteNd Director provides Web-based utilities for creating custom portal pages:

Name of utility	Description	Reference
Portal Personalizer	Lets any portal user create **personal pages** with personalized content and layout	Chapter on *Personalizing Your Portal*

Name of utility	Description	Reference
Portal Administration tool	Lets portal administrators create: ◆ **Container pages**, used for branding corporate sites ◆ **Shared pages**, used for sharing common information among users and groups	Chapter on *Administering the Portal*

Each utility provides a graphical user interface to help you build portal pages using portlet technology with minimal if any Java programming required.

In addition, exteNd Director supplies a set of prebuilt portlets that provide out-of-the-box functions to add to your portal pages for rapid development and testing of application logic.

When you need to take a more specialized approach, you can develop your own portlets using the following tools:

- Pageflow design tools
- Portlet Wizard
- exteNd Director portlet API

 For more information about these tools see the chapter on developing portlets.

 For more information about portlets, see the chapter about portlets.

Types of portal pages

exteNd Director supports several types of portal pages:

Type	Description	Reference
Container page	Page that wraps shared and personal pages with a consistent look and feel Container pages often are used for branding Web sites with a corporate identity	See "Working with container pages" on page 53.
Shared page	Page that provides content that can be shared among users and groups	See "Working with shared pages" on page 58.

Type	Description	Reference
Personal page	Private pages with personalized content for individual users	See "Working with personal pages" on page 61.

Because each type of page can be bound selectively to users and groups in an organization, portal pages are useful for building intranet sites and other portal applications that require personalized content.

How page types interact

When you view a portal page, you see content that is aggregated from your container page and the associated shared or personal page, as described in "How content is determined for the current user" on page 49.

Container pages and shared pages provide the mechanisms for branding all content in an organization with a standard look and feel.

Typically, portal administrators use **container pages** to brand—or wrap—personal pages and shared pages with a corporate identity and with controls for navigating the portal. exteNd Director provides several system portlets to support this functionality:

- Page Navigation portlet can be added to a container page to automatically generate links to all pages the current user has permission to view—including personal pages, shared pages, and other container pages.
- Portal Page Controller portlet designates where on the container page to display the content of a selected shared or personal page.
- Page Header portlet renders custom headers on container pages and can be configured to also provide navigation links.

 For more information, see "Working with container pages" on page 53 and the chapter on *system portlets for portal pages*.

Administrators use **shared pages** to disseminate content that is meant to be shared by specific users in an organization. For more information, see "Working with shared pages" on page 58.

Default portal pages

The portal administrator must define at least one container page for each portal application. exteNd Director ships with a container page and shared page, which are designated in web.xml as the portal's default pages, as follows:

```
<context-param>
    <param-name>PortalDefaultContainerPage</param-name>
    <param-value>DefaultContainerPage</param-value>
    <description>The name of the portal's default container page. A
container page is the page he portal uses to display User and
Shared pages.</description>
</context-param>
<context-param>
    <param-name>PortalDefaultSharedPage</param-name>
    <param-value>DefaultSharedPage</param-value>
    <description>The name of the default shared page.</description>
</context-param>
```

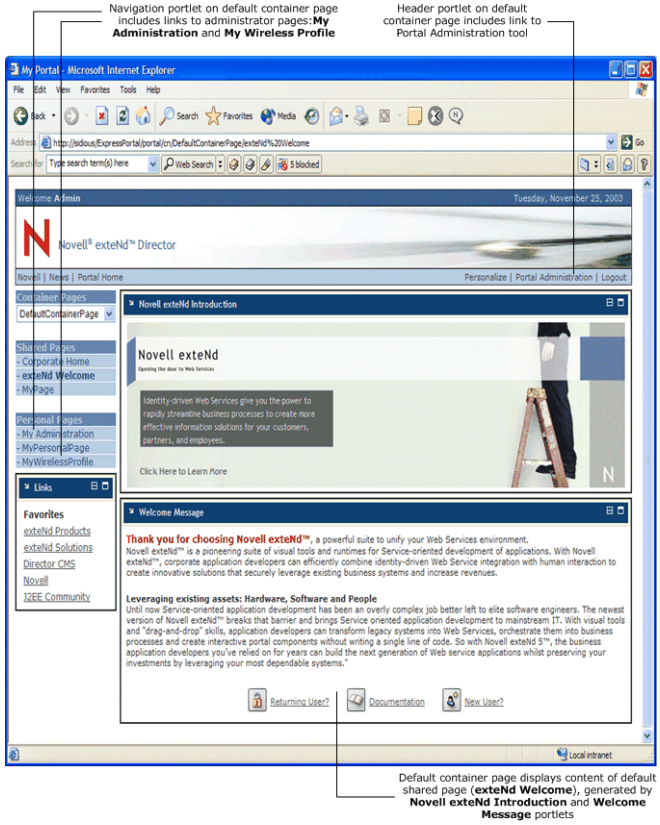
As portal administrators create new container and shared pages, they can change these settings to designate their own pages as defaults for the portal.

Users can modify the default container and shared pages for their own portal sessions by using the Portal Personalizer tool, as described in the chapter on personalizing your portal.

IMPORTANT: As shipped, the default container and shared pages do not require authentication and, therefore, are available to all users. It is recommended that portal administrators **NOT** lock down these default pages to ensure that all users are able to access the portal.

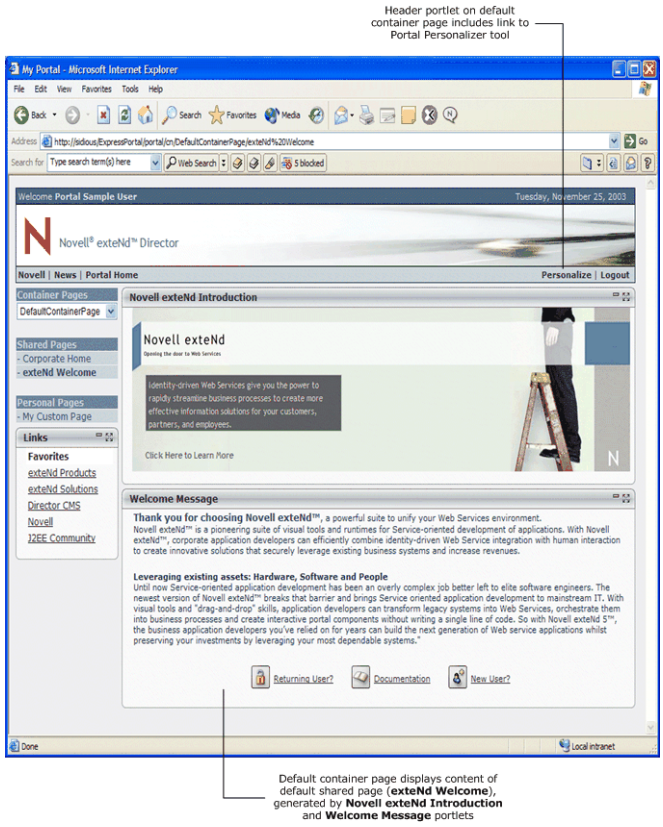
When you log in to the portal for the first time, you see a default portal page that aggregates the content of the default container page and default shared page.

Default portal page for administrators If you are a portal administrator or locksmith user, you'll see this default portal page when you first log in to the portal:



Note that the page includes content designed for portal administrators, including a link to the Portal Administration tool, a utility for creating container and shared pages described in the chapter on *administering the portal*.

Default portal page for non-administrators If you are **not** a portal administrator or locksmith user, you'll see this default portal page when you first log in to the portal:

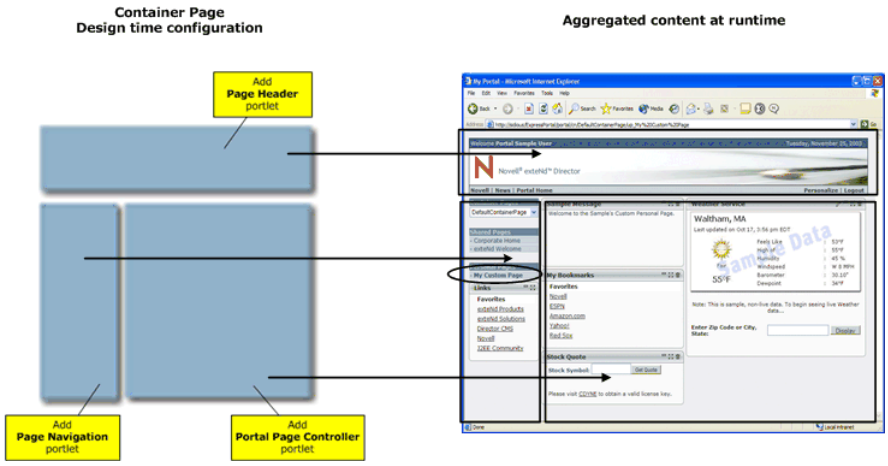


Note that the page includes content designed for users who are not administrators, including a link to the Portal Personalizer tool, a utility for creating personal pages (described in the chapter titled *Personalizing Your Portal*).

How content is determined for the current user

The exteNd Director Portal provides a **Portal Aggregator** to render content for the user in the current session. The content rendered is the aggregate of content from the user's container page, and associated personal or shared page.

For example, the following diagram shows how the Portal determines the content to display for a sample container page configuration:



In this example, a portal administrator configures a container page with a Header-Navigation-Content layout and three system portlets:

- Page Header
- Page Navigation
- Portal Page Controller

When a user selects the link to **My Personal Page** from the navigation frame of this container page at runtime, a render request is generated. The Portal responds to the request by:

1. Determining the user's current container page
2. Determining the selected personal or shared page—in this case, My Custom Page

   NOTE: My Custom Page contains four portlets: Sample Message, My Bookmarks, Stock Quote, and Weather Service.
3. Aggregating content as follows:

   3a Gets the content generated by the Page Header portlet on the user's current container page and displays it in the Header frame

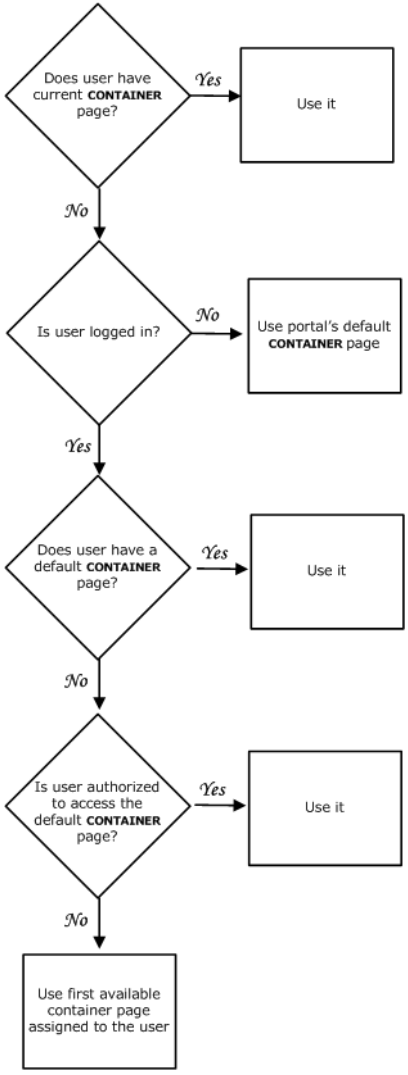
   3b Gets the content generated by the Page Navigation portlet on the user's current container page and displays it in the Navigation frame

3c Gets the content generated by the portlets on My Personal Page and displays it via the Portal Page Controller portlet in the Content frame.

📖 To learn more about content aggregation logic, see "Determining the container page" on page 50 and "Determining the current shared or personal page" on page 52.

Determining the container page

When the user makes a request to the Portal, the Portal determines which **container** page should be displayed, using the following logic:



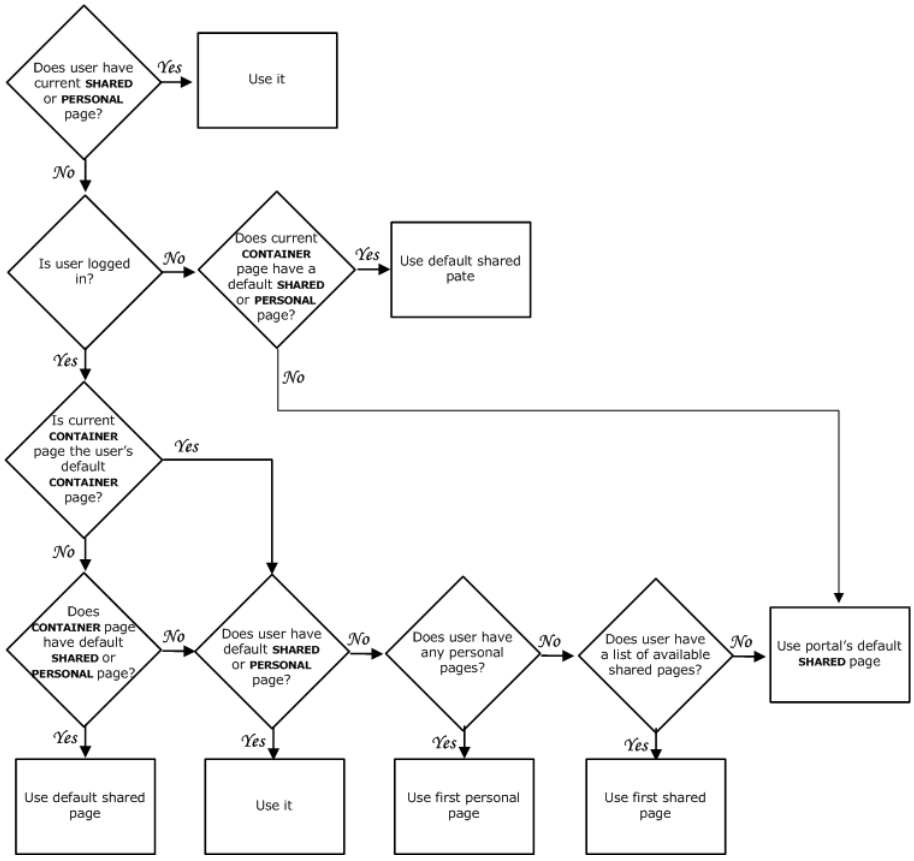
As the flow diagram shows, the Portal Aggregator checks for the following container pages:

Container page	Description
Current	Container page selected by the user in the current session
User default	Default container page set by the user in the Portal Personalizer, as described in the chapter on *personalizing your portal*.
First available	First container page assigned to the user

Portal default	The default container page defined in your project's web.xml file as **PortalDefaultContainerPage**. Here is a sample descriptor:
	`<context-param>`
	`  <param-name>PortalDefaultContainerPage</param-name>`
	`  <param-value>DefaultContainerPage</param-value>`
	`  <description>Portal's default container page.</description>`
	`</context-param>`

Determining the current shared or personal page

After determining the container page for the current user, the Portal Aggregator uses the following logic to determine which shared or personal page to display in the content area of the container page:



As the flow diagram shows, the Portal Aggregator checks for the following shared or personal pages:

Page	Description
Current	Shared or personal page selected by the user in the current session
User default	Default shared or personal page set by the user in the Portal Personalizer, as described in *Personalizing Your Portal*.

Page	Description
Container default	Default shared page assigned to a container page by the portal administrator in the Portal Administration tool, as described in *Administering the Portal*.
First available page	First shared or personal page assigned to the user

Portal default shared	The default shared page defined in your project's web.xml file as **PortalDefaultSharedPage**. Here is a sample descriptor:
	`<context-param>`
	`  <param-name>PortalDefaultSharedPage</param-name>`
	`  <param-value>DefaultSharedPage</param-value>`
	`  <description>Portal's default shared page.</description>`
	`</context-param>`

Working with container pages

Container pages are the conduits for displaying shared and personal pages with content suitable for particular groups of users. Administrators create container pages with images, logos, text, and navigation controls to brand the portal with a corporate identity.

Every portal application requires that at least one container page is defined. exteNd Director ships with a default container page, as described in "Default container page" on page 57.

Container pages must be defined by a portal administrator—that is, a user who is a member of the **PortalAdmin** group. The administrator uses the Portal Administration tool to:

- Specify the contents and layout of container pages
- Assign container pages to groups and users.

  NOTE: The assignment gives users **View** permission only, allowing them to access container pages, but not modify them.

Container pages are not tightly bound to layouts. That means portal administrators can switch layouts for container pages without disrupting the page contents. When the administrator applies a new layout to a page, any previously selected portlets are automatically displayed using the new layout.

To learn how to use the Portal Administration tool to create container pages, see the chapter on administering the portal.

Building blocks

The portal administrator can build container pages with system portlets supplied with exteNd Director or with custom portlets. System portlets provide the following out-of-the-box functions:

Container page function:	Provided by:	Required?
Designate a content area	Portal Page Controller portlet	**Yes**
Navigate available pages in your portal	Page Navigation portlet or Page Header portlet	No, but recommended
Customize page headers	Page Header portlet	No

IMPORTANT: You must add one Portal Page Controller portlet to every container page to designate an area for displaying the content of shared pages or personal pages.

 For more information about these portlets, see Chapter 24, "System Portlets for Portal Pages".

Required content area

The portal administrator **must** add one **Portal Page Controller** portlet to every container page to designate an area for displaying the content of a shared or personal page. If this portlet does not appear on a container page, the Portal cannot render the content of shared pages or personal pages on the container page.

 To learn how to add system portlets to container pages, see the chapter on administering the portal.

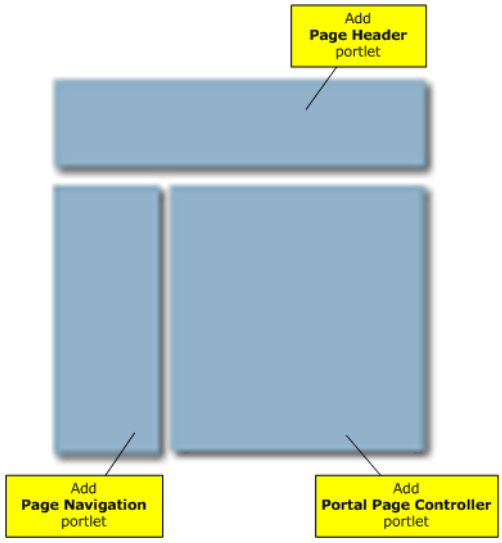
Commonly used layouts for container pages

exteNd Director provides pre-built layouts that can be used on all portal pages. This section describes the layouts that are most commonly used for container pages. Portal administrators can change these layouts at any time without losing content.

 To learn how to apply page layouts, see the chapter on working with portal layouts.

Header Nav Content layout

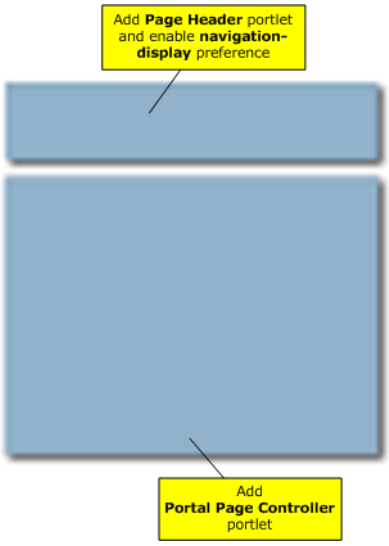
As its name suggests, the **Header Nav Content** layout provides header, navigation, and content frames. In addition to adding custom portlets to this layout, the portal administrator can add system portlets as follows:



For more information, see Chapter 24, "System Portlets for Portal Pages".

Header Content layout

The **Header Content** layout provides header and content frames. In addition to adding custom portlets to this layout, the portal administrator can add system portlets as follows:

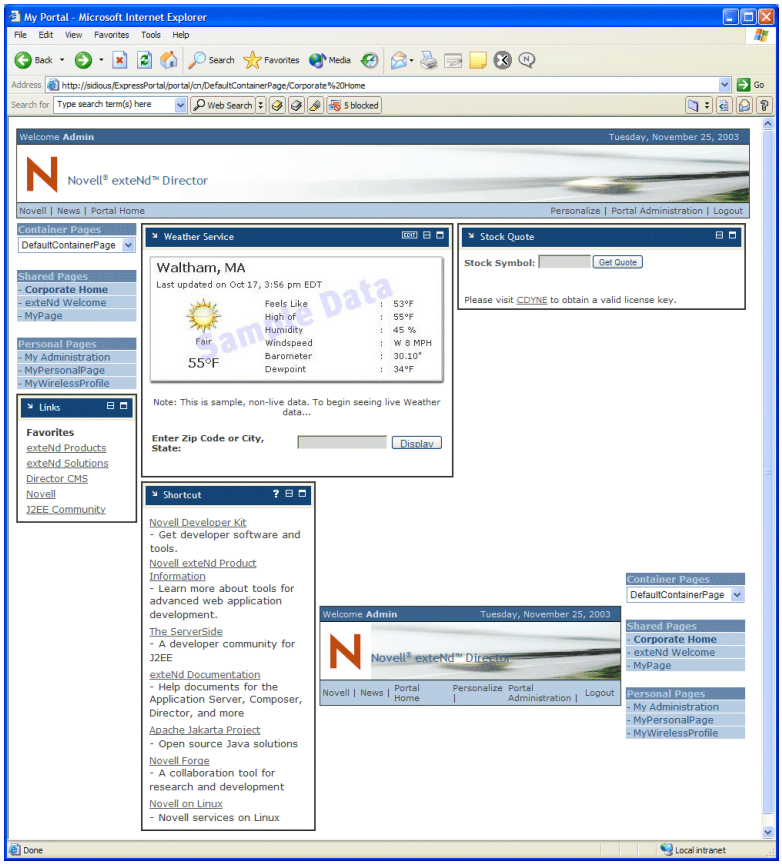


In this layout, you can enable navigation functionality in the Page Header portlet by setting its **navigation-display** preference to **true**.

📖 For more information, see Chapter 24, “System Portlets for Portal Pages”.

Default container page

exteNd Director ships with a default container page. When you log in to the portal for the first time, the default container page is displayed:



As you can see, the default container page displays the content of the default shared page **Corporate Home** in its content frame.

This page is designated in **web.xml** as the default container page for exteNd Director portal applications. Here is the descriptor:

```
<context-param>
  <param-name>PortalDefaultContainerPage</param-name>
  <param-value>DefaultContainerPage</param-value>
  <description>Portal's default container page.</description>
</context-param>
```

The administrator can modify the contents and layout of the default container page and change the portal default by editing the **PortalDefaultContainerPage** parameter in the application's web.xml file.

Working with shared pages

Shared pages provide content to be shared by multiple users. Shared pages must be created by the portal administrator—that is, any user who belongs to the **PortalAdmin** group.

Shared pages are designed to work with container pages. Typically, a portal administrator creates container pages that provide links to the shared pages you are authorized to access and display the content of shared pages you select.

📖 For more information about how container pages work with shared pages, see "How content is determined for the current user" on page 49 and "Working with container pages" on page 53.

The administrator uses the Portal Administration tool to specify the contents and layout of shared pages and to assign shared pages to users and groups. There are two types of permissions for shared page access:

Share page permission	What it allows
View	User can access the shared page, but not modify its content or layout.
Ownership	User can access **and** modify the content and layout of the shared page.

Shared pages are not tightly bound to layouts. That means page owners can switch layouts for their pages without disrupting the page contents. When the owner applies a new layout to a page, any previously selected portlets are automatically displayed using the new layout.

📖 For more information about using the Portal Administration tool for creating shared pages, see the chapter on administering the portal.

Ownership of shared pages

The portal administrator can designate any number of users and/or groups as owners of a shared page. Owners can modify the content and layout of shared pages, and assign of shared pages to users and groups. When multiple owners make changes to a shared page, the last set of edits prevails.

Modifying preferences of portlet registrations on shared pages

Portal administrators and shared page owners can modify portlet preferences on shared pages using the Portal Administration tool, as described in the chapter on administering the portal.

At runtime, any user with access to a shared page can modify the preferences of portlet registrations that have been added to that page if:

- The portlet administrator has enabled the Edit option
- The portlet developer has implemented Edit mode, either explicitly or using the default implementation, as described in the section on default implementation for Edit mode.

Shared page hierarchies

The portal administrator can use the Portal Administration tool to assign a parent page to any shared page. The parent must also be a shared page. When the Page Navigation portlet displays links to shared pages that have parents, the children appear indented under the parent page as a visual indicator of the relationship.

Child pages do not inherit content, preferences, or settings from their parent pages. Conversely, parent pages do not automatically display the content of child pages along with their own content.

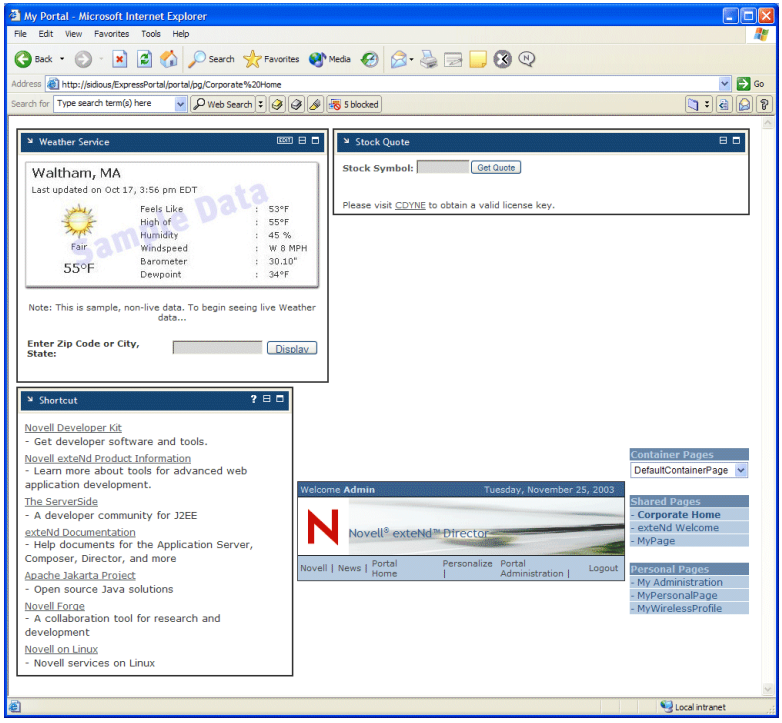
Creating shared page hierarchies is helpful for grouping pages, especially when there are a large number of pages in the list of available shared pages.

NOTE: Portal administrators and locksmith users are authorized to access every shared page defined for a portal. Therefore, when logging in as the portal administrator or locksmith user, be aware that the number of available shared pages in a navigation or selection list will be extremely long.

 To learn how to assign parents to shared pages, see the chapter on administering the portal.

Default shared page

exteNd Director ships with a default shared page called **Corporate Home**:



This page is designated in **web.xml** as the default shared page for exteNd Director portal applications. Here is the descriptor:

```
<context-param>
   <param-name>PortalDefaultSharedPage</param-name>
   <param-value>Corporate Home</param-value>
   <description>Portal's default shared page.</description>
</context-param>
```

When you log in to the portal for the first time, the default shared page is displayed in the content area of the default container page. The administrator can modify the contents and layout of the default shared page and change the portal default by editing the **PortalDefaultSharedPage** parameter in the application's web.xml file.

Working with personal pages

Personal pages can be created by any portal user to show personalized content. Each personal page is private and can be accessed only by the creator. Portal users can create any number of personal pages by using the **Portal Personalizer** to specify page contents and layout. You can place multiple instances of any registered portlet on a single personal page.

 For complete details on using the Portal Personalizer to create personal pages, see the chapter on personalizing your portal.

Personal pages are not tightly bound to layouts. That means users can switch layouts for their pages without disrupting the page contents. When the user applies a new layout to a page, any previously selected portlets are automatically displayed using the new layout.

URLs for accessing portal pages

exteNd Director provides URLs for accessing container pages, shared pages, and personal pages directly, and displaying them in your browser.

URL for container pages

You can use the following URL to access container pages on your server:

`http://`*`server/project context`*`/portal/cn/`*`container page name`*

For example, if your server is **localhost** and your project context is **MyWAR**, you can access the default container page by entering this URL in your browser:

```
http://localhost/MyWAR/portal/cn/DefaultContainerPage
```

The exteNd DirectorPortal responds to a container page URL request by displaying the aggregate of content from the requested container page along with content from the associated shared or personal page, as described in "How content is determined for the current user" on page 49.

You can instruct the Portal to display the content of a specific shared or personal page with your container page by using these URLs:

To request:	Use this URL:
Shared page	`http://`*`server/project context`*`/portal/cn/`*`container page name/shared page name`* Example: `http://localhost/MyWAR/portal/cn/MyContainerPage/MySharedPate`

To request:	Use this URL:

Personal page	`http://server/project context/portal/cn/container page name/up_personal page name`
	Example:
	`http://localhost/MyWAR/portal/cn/MyContainerPage/up_MyUserPage`

URL for shared pages

You can use the following URL to access shared pages on your server:

`http://server/project context/portal/pg/shared page name`

For example, if your server is **localhost** and your project context is **MyWAR**, you can access the default shared page by entering this URL in your browser:

```
http://localhost/MyWAR/portal/pg/DefaultSharedPage
```

URL for personal pages

You can use the following URL to access personal pages on your server:

`http://server/project context/portal/pg/up_personal page name`

For example, if your server is **localhost** and your project context is **MyWAR**, you can access a personal page called MyUserPage by entering this URL in your browser:

```
http://localhost/MyWAR/portal/pg/up_MyUserPage
```

Categorizing pages

The portal administrator can assign categories to shared and container pages to filter access to content within access control lists (ACLs). See Chapter 8, “Working with Page Categories”.

4 Working with Portal Layouts

This chapter explains how to use portal layouts to control the appearance of personal, shared, and container pages. It contains the following sections:

About portal layouts

A *portal layout* is a template that defines how a set of selected portlets should appear on a page. Each personal, shared, and container page in an exteNd Director portal application uses a portal layout to specify how the selected portlets should be arranged on the page. exteNd Director keeps the user's selected portlets separate from the portal layout, so that a page's layout can be changed without affecting the selected portlets.

Predefined layouts

exteNd Director ships with several predefined layouts. The following layouts are available to all users:

Layout	Description	Preview
1 Column	Creates a single column. The portlet flow is top to bottom.	
2 Columns	Creates two columns of equal size. The portlet flow is top to bottom within the columns.	
2 Columns 30/70	Creates two columns of proportional size. The first column occupies 30% of the page width and the second column occupies 70% of the page width.	
3 Columns	Creates three columns of equal size. The portlet flow is top to bottom within the columns.	
Classic Portal Layout	Creates a header area, three columns, and a footer area.	
Header Content	Creates a header area and a content area.	
Header Nav Content	Creates a header area, a navigation bar, and a content area.	

Layout	Description	Preview
Header Nav Content Footer	Creates a header area, a navigation bar, a content area, and a footer area.	

The following layouts are available only to layout administrators:

Layout	Description	Preview
My Novell Layout	Creates a sample XHTML layout.	

Creating your own layouts

You can also create your own layouts. To do this, you use the Portal Layout and Portal Layout Definition Wizards in exteNd Director, as described in Chapter 16, “Creating Custom Layouts”.

Files associated with portal layouts

Each layout in a portal application must provide each of the following files:

File(s)	Description
“Layout descriptor file” on page 67	Describes the layout. This XML file specifies the following information: ◆ Display name ◆ Description ◆ Preview image file ◆ Reference to the layout definition file ◆ Layout definition format (XML or XHTML) You can give the layout descriptor file any name you like.

File(s)	Description
“Layout definition file” on page 68	Specifies the physical characteristics of the layout in XML or XHTML format. This file provides the following information: ◆ Sections within the layout ◆ Width of the main page ◆ Width of each section ◆ How portlets should flow within a particular section (left to right or top to bottom). The layout definition file typically has the same name as the descriptor file, with Def (for definition) appended. For example, if the descriptor file is called HeaderContent.xml, the layout definition file might be called HeaderContentDef.xml. This naming convention is recommended, but not required.
Supporting graphics files	Define preview images for the layout.

The layout descriptor and layout definition files must be stored in the **portal-layout** directory within a resource set.

The internal identifier for a layout is the name of its descriptor file, without the XML extension.

 For more information about where files are located in a resource set, see the section on subdirectories for resources and Java classes in *Developing exteNd Director Applications*.

Layout descriptor file

Each layout used by a portal application must have a layout descriptor file. You can give the layout descriptor file any name you like. The layout descriptor file specifies the following information for the layout:

Element name	Description
portal-layout	The root node of a portal layout
display-name	The name used to identify the layout in the user interface of a portal application.
description	The description of the layout.
preview-image	A small image showing what the layout might look like. The path to this image could be a fully qualified URL or an URL that contains portal replacement strings. For more information on portal replacement strings, see Chapter 23, "Portal Replacement Strings".
layout-definition-file	The path to the layout definition file. This file contains the actual layout.
layout-definition-format	The format of the layout. A layout definition can use either of the following formats: ◆ text/xml (for XML) ◆ text/xhtml (for XHTML) HTML is not a supported layout definition format.
run-role-map	The set of roles that can run a portal page with this layout. If you do not specify a run-role-map, all users can run portal pages with this layout. **IMPORTANT:** If you specify a run-role-map for a layout, you should specify the equivalent list-role-map.
list-role-map	The set of roles that can view this layout from a selection list. If you do not specify a list-role-map, all users can view this layout.

Here is an example of a layout descriptor file for a layout called Header Content. The layout descriptor file is called HeaderContent.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE portal-layout PUBLIC "-//SilverStream Software, LLC//DTD
Portal Layout 5.0//EN" "portal-layout_5_0.dtd">
<portal-layout>
<display-name>Header Content</display-name>
<description>Header and content sections</description>
```

```
<preview-image>$RESOURCE_URL$/portal-
layout/images/HeaderContent.gif</preview-image>
<layout-definition-file>HeaderContentDef.xml</layout-definition-
file>
<layout-definition-format>text/xml</layout-definition-format>
</portal-layout>
```

The string $RESOURCE_URL$ shown above is an example of a **portal replacement string**. Replacement strings allow you to include runtime, context-based information in a portal application. Replacement strings are keywords that reference things that can change dynamically at runtime, such as a user's current theme or the currently rendered portlet ID. When these keywords are detected at runtime, they are replaced based on information from the current HTTP request or the current portal context.

 For more information about portal replacement strings, see Chapter 23, "Portal Replacement Strings".

Layout definition file

The layout definition file for a portal layout describes the sections within the layout, as well as the width of the main page and the width of each section. The layout definition also specifies how portlets should flow within a particular section (left to right or top to bottom).

A layout definition can use either of the following formats:

- **text/xml** (for XML)
- **text/xhtml** (for XHTML)

HTML is not a supported layout definition format.

You specify the format you plan to use in the **<layout-definition-format>** element of the layout descriptor file.

XML format

If you use XML to specify the layout definition, you must use the following elements to define the layout:

Element name	Description
portal-layout-def	The root node of a portal layout definition
section-container	An area within the layout that can contain one or more sections. When a section-container is of type **row**, the generated HTML for the element is a table row (<tr>). When a section-container is of type **column**, the generated HTML for the element is a table cell (<td>).
s3-section	A section within a section-container. Each s3-section can display one or more portlets. S3-sections are used at runtime to allow users to select the portlets they want in those sections. Each s3-section specifies: ◆ A **flow** attribute, which indicates whether portlets within the section should flow from top to bottom (**vertical** flow) or from left to right (**horizontal** flow). ◆ The **width** and **style** settings. ◆ An identifier for the section.
s3-component	Allow you to place content anywhere in your XHTML layout.

Here is an example of an XML layout definition file for a layout called HeaderContent. The layout definition file is called HeaderContentDef.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE portal-layout-def PUBLIC "-//SilverStream Software,
LLC//DTD Portal Layout Def 4.0//EN" "portal-layout-def_4_0.dtd">
<portal-layout-def>
<section-container type="row">
<s3-section flow="vertical" id="1" style="padding-
bottom:5px;padding-right:5px;" width="100%"/>
</section-container>
<section-container type="row">
<s3-section flow="vertical" id="2" style="padding-bottom:5px;"
width="100%"/>
</section-container>
</portal-layout-def>
```

 To see how each element in a portal layout is translated into HTML, see **PortalLayout.xsl** in the portal-style folder of the resource set in your Web application.

XHTML format

If you specify the layout definition in XHTML, you have much more control over the layout than you would with XML. XHTML layouts can be used for personal, shared, and container pages. For example, you might use an XHTML layout to brand a Web site with a particular corporate look. If you define the branding in a container page, each personal page displayed within this container page will have the branding.

When you use the XHTML format, the markup is well-formed. Here are some guidelines you should follow to ensure that your document is well-formed:

- Each tag you specify must have a corresponding end tag
- The case for the begin and end tags must match
- Attribute values must be enclosed in quotes

To reserve space for users to place their selected portlets, you need to place **<s3-section>** tags throughout the XHTML document. XHTML layouts can use scoped paths to dynamically point to portal resources. For example, the following scoped path shows how to access an image in the resource set.

```
<img src="${Portal/Uri/Context/resource/
   portal-layout/images/mynovell_logo.jpg}"/>
```

The XHMTL layout can only contain HTML, HEAD, and BODY tags when it is used as a portal container page.

Here is an example of an XHTML layout definition file for a layout called MyNovell. (It is a portal container page.) The layout definition file is called MyNovell.html:

```
<html>
<head>
<title>MyNovell.com</title>
<s3-component id="ThemeTester" mode="link-only"/>
<script language="JavaScript" type="text/JavaScript">
 onLoadFct = [];
 function compareOnLoadPriority(a, b) {
  if (a != undefined &amp;&amp; b != undefined) {
   if ( a.priority &lt; b.priority ) return -1
   if ( a.priority &gt; b.priority ) return 1
  }
  return 0
 }
function runOnLoad() {
  var lg = onLoadFct.length;
  if (lg &gt; 0) {
   if (lg &gt; 1) onLoadFct.sort(compareOnLoadPriority);
   var i = 0;
   while(i &lt; lg) {
    if (onLoadFct[i] != undefined) {
     (eval(onLoadFct[i].fct))();
    }
    i++;
   }
```

```
  }
 }
function setOnLoad(priority,fct) {
  var pos = onLoadFct.length;
  onLoadFct[onLoadFct.length] = {}
  onLoadFct[pos].priority = priority;
  onLoadFct[pos].fct = fct;
 }
</script>
</head>
<body margintop="0" bgcolor="#336699" marginleft="0"
onLoad="runOnLoad()">
<table border="0" width="100%" bgcolor="#336699">
  <tr>
    <td>
<table border="0" width="100%">
        <tr>
          <td width="300"><img
src="${Portal/Uri/Context/resource/portal-
layout/images/mynovell_logo.jpg}"/></td>
          <td width="65"><a href="../../portlet/Personalize"><img
border="0" src="${Portal/Uri/Context/resource/portal-
layout/images/mynovell_personalize.gif}"/></a></td>
          <td width="65"><a href=""><img border="0"
src="${Portal/Uri/Context/resource/portal-
layout/images/mynovell_myportal.gif}"/></a></td>
          <td></td>
        </tr>
      </table>
</td>
  </tr>
  <tr>
    <td>

    <s3-component id="PhoneList" instance="page_phome"/>
    <table border="0" height="500" cellpadding="0" cellspacing="0"
width="100%">
        <tr>
          <td rowspan="3" width="165">
            <table width="100%" height="95%" border="0">
              <tr>
                <td valign="top">
                  <!-- Here is section 1 where portal components
will go at runtime -->
                  <s3-section id="1" style="padding-bottom:5px;"/>
                </td>
              </tr>
            </table>
          </td>
          <td width="65" valign="bottom"
style="background:url(${Portal/Uri/Context/resource/portal-
layout/images/mynovell_topleft.gif}) no-repeat top left;"></td>
```

```
          <td height="42"
style="background:url(${Portal/Uri/Context/resource/portal-
layout/images/mynovell_top.gif}) top left;"></td>
          <td width="61" height="42" valign="bottom"
style="background:url(${Portal/Uri/Context/resource/portal-
layout/images/mynovell_topright.gif}) no-repeat top right;"></td>
        </tr>
        <tr>
          <td
style="background:url(${Portal/Uri/Context/resource/portal-
layout/images/mynovell_left.gif}) repeat-y top left;"></td>
          <td bgcolor="white" valign="top">
            <!-- Here is section 2 where portal components will go
at runtime -->
  <s3-section id="2"/>
          </td>
          <td
style="background:url(${Portal/Uri/Context/resource/portal-
layout/images/mynovell_right.gif}) repeat-y top right;"></td>
        </tr>
        <tr>
          <td width="65" height="66" valign="top"
style="background:url(${Portal/Uri/Context/resource/portal-
layout/images/mynovell_bottomleft.gif}) no-repeat top left;"></td>
          <td
style="background:url(${Portal/Uri/Context/resource/portal-
layout/images/mynovell_bottom.gif}) repeat-x bottom left;"></td>
          <td width="61" height="66" valign="top"
style="background:url(${Portal/Uri/Context/resource/portal-
layout/images/mynovell_bottomright.gif}) no-repeat top
right;"></td>
        </tr>
      </table>
</td>
  </tr>
</table>
</body>
</html>
```

Working with the layout API

The exteNd Director API provides two interfaces for working with layouts:

- EbiLayoutManager
- EbiLayoutInfo

EbiLayoutManager provides methods for accessing EbiLayoutInfo objects. EbiLayoutInfo provides methods for retrieving various kinds of information about a layout, including its display name and description. The EbiLayoutInfo interface also provides access to the URIs for a layout's preview and thumbnail images.

To access an EbiLayoutManager object, you need to use the getLayoutManager() method on the EboFactory class for the Portal subsystem.

5 Working with Portal Themes

This chapter explains how to use portal themes to control the look and feel of an exteNd Director portal application. It contains the following sections:

About themes

A *portal theme* is a set of visual characteristics that apply to an entire exteNd Director portal application. Once you set the theme for a portal application, the settings associated with the theme apply globally. These settings can potentially change the appearance of portal pages (both PID pages and JSP pages) and portlets, as well as the appearance of personal, shared, and container pages. Themes provide a simple way to ensure a consistent appearance throughout a portal application.

Predefined themes

exteNd Director ships with several predefined themes:

- Basic Blue
- Black N Blue
- Dotted Border

- JellyBean
- Professional
- Titanium

These themes are defined in the **portal_core_resource.jar**.

TIP: To preview these themes, display the theme tester page by entering the following URL in your browser:

```
http://host/project context/portal/pages/ThemeTester.html
```

Files associated with themes

A portal theme consists of several files:

File(s)	Description
"Theme descriptor file" on page 77	Describes the theme. This file provides a display name, description, preview image file, and thumbnail image file for the theme. The theme descriptor file must be named **theme.xml**. Each theme used by a portal application must have a separate theme.xml file. See "Theme descriptor file" on page 77.
"Theme style sheet" on page 79	Contains all of the attributes and properties that define the appearance of a theme when it is rendered at display time. The theme CSS file must be named **theme.css**. Each theme used by a portal application must have a separate theme.css file. See "Theme style sheet" on page 79.
"Theme image files" on page 84	Define various images that are used by the theme. These graphics files are stored in a directory called **images**.

The theme.css and theme.xml files, and the images directory are all stored in a *theme folder*. The theme folder is a subdirectory of the **portal-theme** directory within a resource set. The name of the theme folder provides a key for the theme and uniquely identifies the theme.

For example, here is what the directory structure might look like for the BasicBlue theme:

 For more information about where files are located in a resource set, see the chapter on using the Resource Set in an exteNd Director application in *Developing exteNd Director Applications*.

Custom themes

You can create custom themes using the Portal Themes Wizard in the exteNd Director development environment, as described in the chapter on creating custom themes.

Theme descriptor file

Each theme in your portal application must have a theme descriptor file called **theme.xml**. The theme descriptor file specifies the following information for the theme:

Element name	Description
portal-theme	The root node of the portal theme.

Element name	Description
display-name	The name used to identify the theme in the user interface of a portal application.
description	The description of the theme.
preview-image	An image showing what the theme will look like at display time. The theme selector displays preview images for each theme, as described in the section on setting the theme for your portal. The path to this image can be a fully qualified URL or an URL that contains portal replacement strings. 📖 For more information, see the chapter on portal replacement strings.
thumbnail-image	A smaller image showing what the theme will look like at display time. This image is typically used in user interfaces listing available themes. The path to this image can be a fully qualified URL or an URL that contains portal replacement strings. 📖 For more information, see the chapter on portal replacement strings.

Here is the descriptor file for the BasicBlue theme:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE portal-theme PUBLIC "-//SilverStream Software, LLC//DTD
Portal Theme 5.0//EN" "portal-theme_5_0.dtd">
<portal-theme>
  <display-name>Basic Blue</display-name>
  <description>Basic Blue theme, smooth plastic</description>
  <preview-image>$RESOURCE_URL$/portal-
theme/BasicBlue/images/preview.gif
  </preview-image>
<thumbnail-image>$RESOURCE_URL$/portal-
theme/BasicBlue/images/thumbnail.gif
  </thumbnail-image>
</portal-theme>
```

Each theme that ships with exteNd Director comes with its own theme.xml file, as described in “Files associated with themes” on page 76.

Creating custom portal themes

📖 See “Creating a portal theme” on page 272.

Theme style sheet

Each theme in your application must have a style sheet called **theme.css**. This file is a standard cascading style sheet (CSS) that will work in Netscape 6 and Microsoft Internet Explorer 5 (and higher versions). Each theme that ships with exteNd Director comes with its own theme.css file.

The style sheet defines a set of classes, some for exteNd Director styles and others for Java Portlet 1.0-compliant portlet styles. These styles can be used to alter the display of portal pages—and of the portlets and XForms 1.0-compliant Web forms that appear on portal pages.

You can create your own styles by defining new classes and editing the settings for the predefined exteNd Director styles.

To locate theme style sheets, see "Files associated with themes" on page 76.

Standard exteNd Director style classes

Standard exteNd Director style definitions in theme.css files contain an **nv-** label.

Standard style types

There are several types of exteNd Director style classes, each designated by a unique prefix:

Style class prefix	Style class description	Example
.nv-	Style that can be applied anywhere within a portal page, portlet, or XForms 1.0-compliant Web form (XForm)	`.nv-fontSmall {font-size : 9pt;}`
***HTML tag*.nv-**	Style that applies to the specified HTML tag	`A.nv-Anchor:hover { text-decoration : underline; color : #3C5279; }` **NOTE:** This style applies to the HTML anchor tag **A**
.nvi-	Style that applies to XForms	`.nvi-link-style:hover { text-decoration : underline; color : #3B5367; background-color : transparent;`

Standard style groups

Here is a list of the predefined exteNd Director style definitions, grouped according to the types of display characteristics they alter:

Style group	Description	Classes defined
Fonts	Define font styles that can be applied anywhere in a portal page or portlet. These styles include settings for the **font-family** and **font-style** HTML properties.	◆ .nv-font ◆ .nv-fontExtraSmall ◆ .nv-fontSmall ◆ .nv-fontMedium ◆ .nv-fontLarge ◆ .nv-fontExtraLarge
Foreground colors	Define foreground color styles that can be applied anywhere in a portal page or portlet where the **color** HTML property is valid. The numbered color styles are ordered from dark to light. The .nv-color1 style defines the darkest color, and the .nv-color10 style defines the lightest color.	◆ .nv-color1 ◆ .nv-color2 ◆ .nv-color3 ◆ .nv-color4 ◆ .nv-color5 ◆ .nv-color6 ◆ .nv-color7 ◆ .nv-color8 ◆ .nv-color9 ◆ .nv-color10
Background colors	Define background color styles that can be applied anywhere in a portal page or portlet where the **background-color** HTML property is valid. The numbered color styles are ordered from dark to light. The .nv-color1 style defines the darkest color, and the .nv-color10 style defines the lightest color.	◆ .nv-backgroundColor1 ◆ .nv-backgroundColor2 ◆ .nv-backgroundColor3 ◆ .nv-backgroundColor4 ◆ .nv-backgroundColor5 ◆ .nv-backgroundColor6 ◆ .nv-backgroundColor7 ◆ .nv-backgroundColor8 ◆ .nv-backgroundColor9 ◆ .nv-backgroundColor10

Style group	Description	Classes defined
Border colors	Define border color styles that can be applied anywhere in a portal page or portlet where the **border-color** HTML property is valid. The numbered color styles are ordered from dark to light. The .nv-color1 style defines the darkest color, and the .nv-color10 style defines the lightest color.	◆ .nv-borderColor1 ◆ .nv-borderColor2 ◆ .nv-borderColor3 ◆ .nv-borderColor4 ◆ .nv-borderColor5 ◆ .nv-borderColor6 ◆ .nv-borderColor7 ◆ .nv-borderColor8 ◆ .nv-borderColor9 ◆ .nv-borderColor10
Page-level styles	Define **font-family**, **color**, **background**, and **background-color** settings for the BODY tag.	◆ BODY
Paragraph-level styles	Define **font-size**, **font-weight**, and **color** settings for the P tag.	◆ .nv-paragraphTextBody ◆ .nv-paragraphTextHeader
Table-level styles	Define settings for the TABLE tag.	◆ .nv-table ◆ .nv-table-header ◆ .nv-table-row-even ◆ .nv-table-row-odd
Form styles	Define **font-size**, **font-weight,** and **color** settings for form fields, labels, and buttons.	◆ nv-formFieldLabel ◆ nv-formField ◆ nv-formButton
Anchor styles	Define **text-decoration** and **color** settings for the A tag. These settings include pseudo-classes that apply to the various user states associated with the A tag.	◆ A.nv-Anchor, A.nv-Anchor:active, A.nv-Anchor:link, A.nv-Anchor:visited ◆ A.nv-Anchor:hover

Style group	Description	Classes defined
exteNd Director Portlet styles	Define various settings associated with the decoration of portlets. You will not ordinarily need to use these styles when developing custom portlets.	• .nv-componentContainer • .nv-titleBarContainer • .nv-titleBarBorderLeft • .nv-titleBarContentLeft • .nv-titleBarContentRight • .nv-titleBarBorderRight • .nv-bodyContainer • .nv-bodyBorderLeft • .nv-bodyBorderRight • .nv-footerContainer • .nv-footerBorderLeft • .nv-footerContentLeft • .nv-footerContentRight • .nv-footerBorderRight • A.nv-titlebar-link • A.nv-titlebar-link:active, A.nv-titlebar-link:link, A.nv-titlebar-link:visited • A.nv-titlebar-link:hover

Java Portlet 1.0-compliant portlet style definitions

exteNd Director also supports standard CSS classes defined by the Java Portlet 1.0 specification.

These classes are defined with the prefix **.portlet**, as in this example:

```
.portlet-font {
  font-family : Verdana, Arial, Helvetica, sans-serif;
  font-size : 10pt;
}
```

 For a list and description of each style definition, see CSS Style Definitions in the Java Portlet 1.0 specification.

The styles defined for portlets give the theme developer a great deal of control over the runtime display of portlets on a page.

 For details on using the portlet styles to theme-enable a portlet, see "Creating portlets that are theme-enabled" on page 87.

Commenting out properties in styles

By convention, each of the standard exteNd Director style classes specifies values for a common set of HTML properties. However, a class definition need not specify values for all of the possible HTML properties normally associated with the class.

Some of the standard classes for the installed themes use a non-standard prefix to comment out certain property settings. When the browser sees a prefix that it does not recognize, it simply ignores the property. The following example uses a prefix **x-** to comment out the visibility property of the footer container style:

```
.nv-footerContainer {
      x-visibility : hidden;
      font-size:2pt;
      height:16px;
      width : 100%;
}
```

Creating a custom theme CSS file

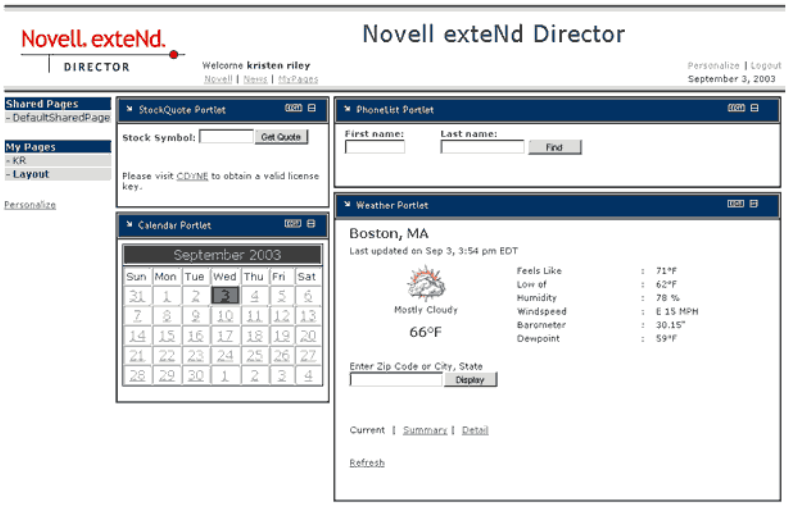
See "Creating a theme CSS file" on page 274.

Theme image files

A theme can include preview and thumbnail images, which give the user an idea what the theme will look like. A theme can also include images that actually appear when the theme is displayed at runtime. These can include background and foreground images, as well as theme option images, which control the appearance of portal options.

Theme preview image file

The theme preview image file is typically a screen shot showing what a theme looks like at display time. Here is the preview image file for the BlackNBlue theme:



The size of the preview image should be 320x320.

Theme thumbnail image file

The thumbnail image file is a smaller image that shows what a theme looks like. Here is the thumbnail image for the BlackNBlue theme:

This image is typically used to list available themes in a user interface. The size of this preview image should be 45x45.

Theme option image files

Themes can alter the appearance of portal options. For example, suppose a portlet title bar provides buttons to allow the user to edit, minimize, or restore the portlet. You can associate several images with each button so that you can change the button image in response to user events. A theme can alter the image for each user event, giving a consistent look when the user changes the theme.

Each theme can supply a unique set of images for the various events associated with a portal option. However, the file names for these images should be the same in each portal theme. By using a consistent naming convention across themes, you ensure that each user event will have an image to display regardless of which theme has been selected.

For example, you can create an image called **edit_onmouseover.gif** to handle the edit user event for the Edit option associated with portlet Edit mode. In this case each theme supported by the application should include a file with the same name so that the onmouseover event always has an image to display.

Creating pages that are theme-enabled

Portal pages can provide support for themes. To allow a PID or JSP page to alter its appearance when a user selects a new theme, you need to make some changes to the source for the page.

To create a page that is theme-enabled, you need to:

- Include a <link> tag in the head section of the page that references the currently selected theme CSS file.

  📖 For details about including the link in a JSP page, see "Including the CSS link in a JSP page" on page 86. For details about including the link in a PID page, see "Including the CSS link in a PID page" on page 86.

- Add HTML tags or portlets to the page that take advantage of the classes defined in the theme CSS file. By mapping your tags to classes in the CSS file, you can ensure that these tags will alter their appearance according to the display characteristics of the currently selected theme. For example, you might want to map each anchor tag in your page to the A.nv-Anchor class in the CSS file:

  ```
  <p><a class="A.nv-Anchor" href="http://www.novell.com">link</a>
  that changes styles on a hover.</p>
  ```

  You can use the standard Director class definitions (those that have names that contain **nv** labels) or use custom class definitions that you define in the CSS file.

Including the CSS link in a JSP page

To include a <link> tag in the head section of a JSP page that references the currently selected theme, you need to use the getThemeLink tag.

Here's an example that shows how this is done:

```
<html>
<head>
<title>My Theme-Enabled Page</title>

<%@ taglib uri="/portal" prefix="ep" %>
<ep:getThemeLink />
</head>
...
```

The getThemeLink tag causes a <link> tag to be added to the generated HTML for the page. The <link> tag includes a path to the CSS file for the theme. The theme folder in the path is set to the selected theme. For example, when the user selects the BasicBlue theme, the <link> tag looks something like this:

```
<html>
<head>
<title>My Theme-Enabled Page</title>
<lnk rel='stylesheet' type="text/css"
href='http://localhost/Director/MyEar/Portal/main/resource/portal-
theme/BasicBlue/theme.css'/>
</head>
...
```

Including the CSS link in a PID page

To include a <link> tag in the head section of a PID page that references the currently selected theme, you need to use the s3-component tag to add the **PortalUrlHelper** portlet to the page. The argument to the PortalUrlHelper portlet includes the syntax for the <link> tag.

Here's an example that illustrates how this is done:

```
<html>
<head>
<title>My Theme-Enabled Page</title>
<s3-component id="PortalUrlHelper" instance="Helper"
SUBST_STRING="<link rel='stylesheet' type='text/css'
href='$THEME_URL$/theme.css'/>"/>
</head>
...
```

The PortalUrlHelper portlet causes a <link> tag to be added to the generated HTML for the page. The <link> tag includes a path to the CSS file for the theme. The theme folder in the path is set to the selected theme. For example, when the user selects the BasicBlue theme, the <link> tag looks something like this:

```
<html>
<head>
<title>My Theme-Enabled Page</title>
<lnk rel='stylesheet' type="text/css"
href='http://localhost/Director/MyEar/Portal/main/resource/portal-
theme/BasicBlue/theme.css'/>
</head>
...
```

Creating portlets that are theme-enabled

A portlet can use most of the style definitions in the theme.css file. For example, an HTML portlet might use one or more font or color styles to alter the appearance of text or controls. In the example shown below, the HTML table generated by a portlet uses a theme-specific font style. It also uses two numbered color styles to alternate the color of rows:

```
<table class="nv-fontExtraSmall" width="99%" border="0"
cellspacing="0" cellpadding="1">

 <tr class="nv-table-row-even">
  <td width="40%"> Bill Lumbergh</td>
  <td width="40%">781-555-1171</td>
  <td width="20%"><div align="center"></div>
  </td>
 </tr>
 <tr class="nv-table-row-odd">
  <td> Peter Gibbons </td>
  <td>781-555-3457</td>
  <td><div align="center"></div>
  </td>
 </tr>
 <tr class="nv-table-row-even">
  <td> Michael Bolton </td>
  <td>781-555-3566</td>
  <td><div align="center"></div>
  </td>
 </tr>
 <tr class="nv-table-row-odd">
  <td> Milton Waddams </td>
  <td>781-555-3442</td>
  <td><div align="center"></div>
  </td>
 </tr>
 <tr class="nv-table-row-even">
```

```
  <td> Samir Nayeenanajar </td>
  <td>781-555-3316</td>
  <td><div align="center"></div>
  </td>
 </tr>
</table>
```

An HTML portlet could use string concatenation techniques in the doView() method to generate this data. For an XML portlet, the styles would be specified in an XSL style sheet, as shown below:

```
<!-- Alternate the color of each row in the table -->
<xsl:template match="employee">
 <xsl:choose>
  <xsl:when test="position() mod 2 = 1">
   <xsl:call-template name="employee_data">
   <xsl:with-param name="StyleClass">nv-backgroundColor3</xsl:with-
param>
   </xsl:call-template>
  </xsl:when>

 <xsl:otherwise>
   <xsl:call-template name="employee_data">
    <xsl:with-param name="StyleClass">nv-backgroundColor5
      </xsl:with-param>
    </xsl:call-template>
 </xsl:otherwise>
 </xsl:choose>
</xsl:template>
```

How the default decorator class renders a portlet

The runtime display of a portlet is controlled by a portal decorator. A *portal decorator* is a Java class that builds the various display elements for a portlets on a page. exteNd Director provides a *default decorator* class called EboDefaultPortalDecorator, which is in the com.sssw.portal.core package.

To render the title bar, body, and footer for a portlet, the default decorator generates three separate HTML tables. Each of these tables uses class definitions provided in the theme.css file to ensure that the display is appropriate for the currently selected theme.

The following extract from the source code for EboDefaultPortalDecorator class shows how the standard portlet class definitions are used to render a portlet. Each class name defined in the theme.css file is highlighted in the source code:

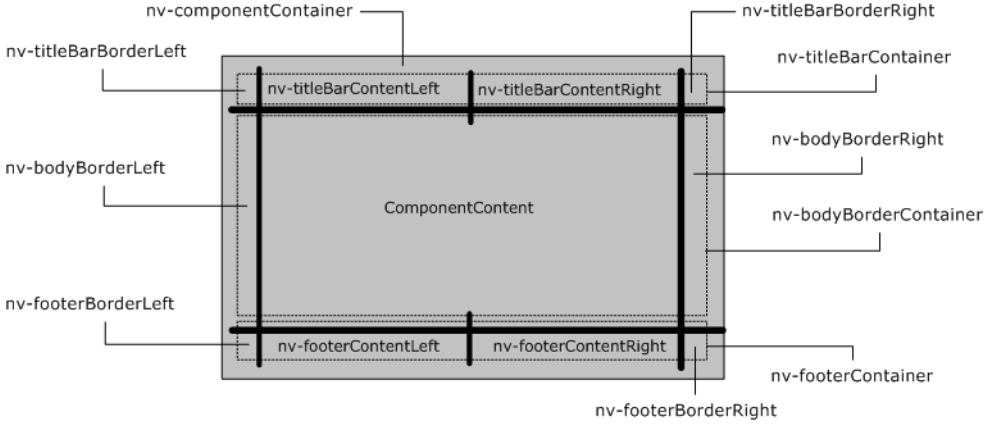
```
package com.sssw.portal.core;

import com.sssw.portal.api.*;

public class EboDefaultPortalDecorator extends
EboComponentDecorator{
```

```
public String decorateComponentData(EbiPortalContext context,
String componentID, String componentData) {

  return "<DIV class=nv-componentContainer>"
    + getTitleBarFragment(context)
    + getBodyFragment(context, componentData)
    + getFooterFragment(context)
    + "</DIV>";
 }


public String getTitleBarFragment(EbiPortalContext context){

  StringBuffer buffer = new StringBuffer("");

  // Build the title bar for this component
  // It is a table which consists of 1 row with 4 columns
(leftBoder, leftContent, RightContent, rightBorder)
  buffer.append("<table class=\"nv-titleBarContainer\" border=\"0\"
cellspacing=\"0\" cellpadding=\"0\">\n");
  buffer.append("  <tr>\n");
  buffer.append("   <td class=\"nv-titleBarBorderLeft\"></td>\n");
  buffer.append("   <td class=\"nv-titleBarContentLeft\">" +
   getTitleBarContentLeft(context) + "</td>\n");
  buffer.append("   <td class=\"nv-titleBarContentRight\">" +
   getTitleBarContentRight(context) + "</td>\n");
  buffer.append("   <td class=\"nv-titleBarBorderRight\"></td>\n");
  buffer.append("  </tr>\n");
  buffer.append("</table>\n");

  return buffer.toString();

}


public String getBodyFragment(EbiPortalContext context, String
contentFragment){

  StringBuffer buffer = new StringBuffer();


  buffer.append("<table class=\"nv-bodyContainer\" border=\"0\"
cellspacing=\"0\" cellpadding=\"0\">\n");
  buffer.append("  <tr>\n");
  buffer.append("   <td class=\"nv-bodyBorderLeft\"></td>\n");
  buffer.append("   <td>" + contentFragment + "</td>\n");
  buffer.append("   <td class=\"nv-bodyBorderRight\"></td>\n");
  buffer.append("  </tr>\n");
  buffer.append("</table>\n");

  return buffer.toString();
}
```

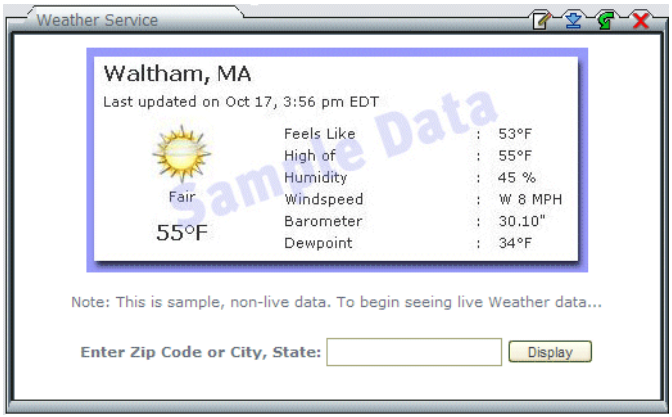
```
public String getFooterFragment(EbiPortalContext context){
  StringBuffer buffer = new StringBuffer();


  buffer.append("<table class=\"nv-footerContainer\" border=\"0\"
   cellspacing=\"0\" cellpadding=\"0\">\n");
  buffer.append("  <tr>\n");
  buffer.append("   <td class=\"nv-footerBorderLeft\"></td>\n");
  buffer.append("   <td class=\"nv-footerContentLeft\">" +
   getFooterContentLeft(context) + "</td>\n");
  buffer.append("   <td class=\"nv-footerContentRight\">" +
   getFooterContentRight(context) + "</td>\n");
  buffer.append("   <td class=\"nv-footerBorderRight\"></td>\n");
  buffer.append("  </tr>\n");
  buffer.append("</table>\n");

  return buffer.toString();
    }
...
```

How the CSS file changes the appearance of a portlet

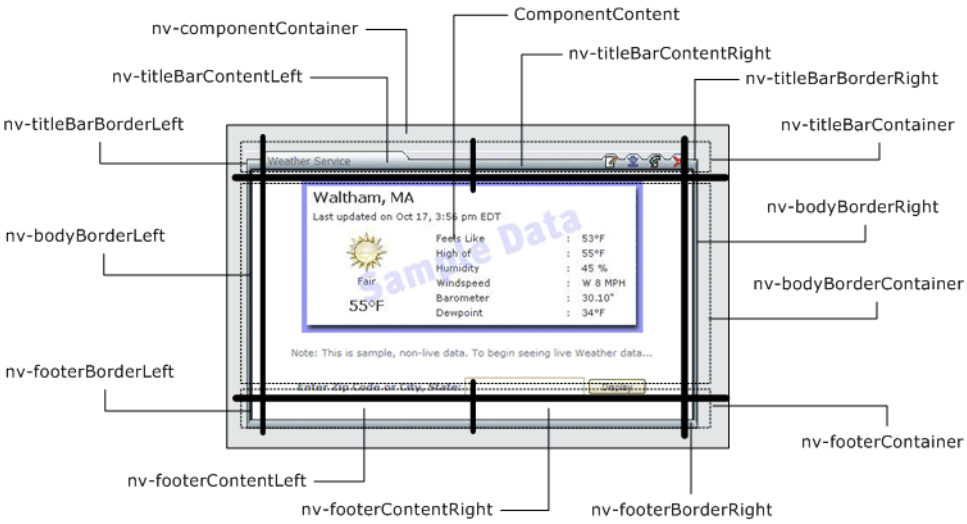
The CSS file for a theme divides the visual display of a portlet into several sections, as shown below:



How does this display scheme apply to a real-world example? Consider the Weather Service portlet, displayed here in the Titanium theme:



The following illustration shows how each display section of the Weather Service portlet is rendered in the Titanium theme:



The theme.css file for the Titanium theme specifies background images for most of the display sections for a portlet:

Class	Description	Image
.nv-componentContainer	Component container	None
.nv-titleBarContainer	Title bar container	

Class	Description	Image
.nv-titleBarBorderLeft	Left border for the title bar	
.nv-titleBarContentLeft	Left content for the title bar	
.nv-titleBarBorderRight	Right border for the title bar	
.nv-titleBarContentRight	Right content for the title bar	
.nv-titleBarContainer	Container for the title bar	
.nv-bodyContainer	Body container	None
.nv-bodyBorderLeft	Left border for the body	
.nv-bodyBorderRight	Right border for the body	
.nv-footerContainer	Footer container	
.nv-footerBorderLeft	Left border for the footer	
.nv-footerContentLeft	Left content for the footer	
.nv-footerContentRight	Right content for the footer	
.nv-footerBorderRight	Right border for the footer	
A.nv-titlebar-link	Anchor style for title bar links	None
A.nv-titlebar-link:active, A.nv-titlebar-link:link, A.nv-titlebar-link:visited	Anchor style for title bar links that are in the active, link, or visited state	None
A.nv-titlebar-link:hover	Anchor style for title bar links that are in the hover state	None

Working with the theme API

The exteNd Director API provides two interfaces for working with themes:

- EbiThemeManager
- EbiThemeInfo

EbiThemeManager provides methods for accessing EbiThemeInfo objects. EbiThemeInfo provides methods for retrieving various kinds of information about a theme, including its display name and description. The EbiThemeInfo interface also provides access to the URIs for a theme's preview and thumbnail images.

To access an EbiThemeManager object, you need to use the getThemeManager() method on the EboFactory class for the Portal subsystem.

6 Working with Portal Decorators

This chapter explains how to use portal decorators to control the appearance of portlets. It contains the following sections:

About portal decorators

A *portal decorator* is an XSL style sheet that decorates the dynamic content generated by a portlet.

When a portal request is received at runtime, the Portal Aggregator responds by building an XML document that describes the content to be generated on the requested page. For each portlet on the page, the Portal Aggregator determines whether decoration is required and if so, applies the portal decorator style sheet specified as PortalDecoratorStyle in web.xml.

Here is an excerpt of the XML generated for the Stock Quote portlet that is displayed on the Corporate Home shared page in the Express Portal application. Three options are decorated—Minimize, Restore, and Maximize—as shown highlighted in the XML code:

```
<portlet-decoration>
  <display-name>Stock Quote</display-name>
  <options>
    <option>
      <option-id>minimize</option-id>
      <display-name>Minimize</display-name>
      <option-text>Min</option-text>
```

```
      <option-link>?urlType=Render&amp;amp;wsrp-
windowstate=minimized&amp;amp;novl-
regid=StockQuotePortlet&amp;amp;novl-
inst=CorporateHome_StockQuotePortlet
      </option-link>
      <link-target/>
      <tool-tip>Minimize this content</tool-tip>
      <hide-states>
        <hide-state>minimized</hide-state>
        <hide-state>maximized</hide-state>
      </hide-states>
      <images>
        <normal>http://localhost:8080/Director/resource/portal-
theme/DottedBorder/images/minimize.gif</normal>
        <onmouseout>http://localhost:8080/Director/resource/portal-
theme/DottedBorder/images/minimize.gif</onmouseout>
        <onmousedown>http://localhost:8080/Director/resource/portal
-theme/DottedBorder/images/minimize.gif</onmousedown>
        <onmouseover>http://localhost:8080/Director/resource/portal
-theme/DottedBorder/images/minimize.gif</onmouseover>
      </images>
    </option>
    <option>
      <option-id>restore</option-id>
      <display-name>Restore</display-name>
      <option-text>Restore</option-text>
      <option-link>?urlType=Render&amp;amp;wsrp-
windowstate=normal&amp;amp;novl-
regid=StockQuotePortlet&amp;amp;novl-
inst=CorporateHome_StockQuotePortlet
      </option-link>
      <link-target/>
      <tool-tip>Restore window state</tool-tip>
      <hide-states>
        <hide-state>normal</hide-state>
      </hide-states>
      <images>
        <normal>http://localhost:8080/Director/resource/portal-
theme/DottedBorder/images/restore.gif</normal>
        <onmouseout>http://localhost:8080/Director/resource/portal-
theme/DottedBorder/images/restore.gif</onmouseout>
        <onmousedown>http://localhost:8080/Director/resource/portal
-theme/DottedBorder/images/restore.gif</onmousedown>
        <onmouseover>http://localhost:8080/Director/resource/portal
-theme/DottedBorder/images/restore.gif</onmouseover>
      </images>
    </option>
    <option>
      <option-id>maximize</option-id>
      <display-name>Maximize</display-name>
      <option-text>Max</option-text>
      <option-link>?urlType=Render&amp;amp;wsrp-
windowstate=maximized&amp;amp;novl-
```

```
regid=StockQuotePortlet&amp;amp;novl-
inst=CorporateHome_StockQuotePortlet
      </option-link>
      <link-target>_new</link-target>
      <tool-tip>Maximize this portlet</tool-tip>
      <hide-states>
        <hide-state>maximized</hide-state>
      </hide-states>
      <images>
        <normal>http://localhost:8080/Director/resource/portal-
theme/DottedBorder/images/maximize.gif</normal>
        <onmouseout>http://localhost:8080/Director/resource/portal-
theme/DottedBorder/images/maximize.gif</onmouseout>
        <onmousedown>http://localhost:8080/Director/resource/portal
-theme/DottedBorder/images/maximize.gif</onmousedown>
        <onmouseover>http://localhost:8080/Director/resource/portal
-theme/DottedBorder/images/maximize.gif</onmouseover>
      </images>
    </option>
  </options>
</portlet-decoration>
```

What is decorated

The following content is decorated:

- Title bar
- Borders around portlet body and footer
- Portal options

Window state influences how portlets are decorated. For example, the portlet body and footer are **not** decorated when the portlet's window state is **minimized** because these elements are not displayed in that state.

When is decoration required?

Decoration is required in the following situations:

- When the title bar is enabled in a portlet
- When options are enabled in a portlet

Using the default decorator for the Portal subsystem

By default, exteNd Director provides a portal decorator style sheet called **PortalDefaultDecorator.xsl**, located in the portal-style directory of the resource set. This style sheet is specified as the default decorator style sheet in web.xml, as follows:

```
<context-param>
        <param-name>PortalDecoratorStyle</param-name>
        <param-value>PortalDefaultDecorator</param-value>
        <description>This stylesheet is used to decorate the
portlet data. Titlebar, border, options, etc...</description>
</context-param>
```

The default portal decorator style sheet specifies three areas for each portlet—title bar, portlet body, and footer—each in a separate HTML table. Inside each table, the style sheet uses CSS classes to specify the HTML elements to be generated. The CSS classes in the decorator style sheet correspond to the same CSS classes in the theme style sheet. This association ensures that the decorated conforms to the display characteristics defined for the currently selected theme.

For example, the table tag for the title bar area specifies **s3-titleBarBorderLeft** as one of its classes. The display characteristics of this class are defined in the CSS file for the theme, as in this example from the BasicBlue theme.css file:

```
.nv-titleBarBorderLeft, .s3-titleBarBorderLeft {
      background-image : url(images/TitlebarLeft.gif);
      background-repeat : no-repeat;
      background-position : top left;
      width : 12px;
}
```

For more information on the interaction between themes and portlet decorators, see Chapter 5, “Working with Portal Themes”.

Title bar This excerpt shows how the default decorator style sheet specifies decoration for the title bar:

```
<!-- TitleBar Fragment -->
<table border="0" cellpadding="0" cellspacing="0" class="s3-
titleBarContainer" width="100%">
    <tr>
        <td class="s3-titleBarBorderLeft">Â </td>
        <td class="s3-titleBarContentLeft" nowrap="true">
            <xsl:value-of select="display-name"/>
        </td>
        <td class="s3-titleBarContentRight">
            <xsl:apply-templates select="options"/>
        </td>
        <td class="s3-titleBarBorderRight">Â </td>
    </tr>
</table>
```

Portlet body This excerpt shows how the default decorator style sheet specifies decoration for the portlet body:

```
<!-- Portlet Body -->
<table border="0" cellpadding="0" cellspacing="0" class="s3-
bodyContainer" width="100%">
    <tr>
        <td class="s3-bodyBorderLeft">Â </td>
        <td>
            <xsl:apply-templates select="../portlet-data"/>
        </td>
        <td class="s3-bodyBorderRight">Â </td>
    </tr>
</table>
```

Footer This excerpt shows how the default decorator style sheet specifies decoration for the footer:

```
<!-- Footer Fragment Nonbreaking Space:   -->
<table border="0" cellpadding="0" cellspacing="0" class="s3-
footerContainer" width="100%">
    <tr>
        <td class="s3-footerBorderLeft">Â </td>
        <td class="s3-footerContentLeft">Â </td>
        <td class="s3-footerContentRight">Â </td>
        <td class="s3-footerBorderRight">Â </td>
    </tr>
</table>
```

Creating a custom decorator

To create a custom decorator, you must create an XSL style sheet and substitute the name of your style sheet in web.xml. Your custom style sheet must use the same classes as those specified in the theme.css files, the style sheets for themes. The following procedure explains how to create a custom decorator by copying the default decorator style sheet.

➢ **To create a custom decorator:**

1. Start exteNd Director and open the project of interest.
2. Open **PortalDefaultDecorator.xsl** from the portal-style directory in the project's resource set.
3. Save the style sheet with a new name in the same directory.
4. Modify the style sheet as desired, preserving the classes that correspond to theme classes.

5. Open **web.xml** in your project's WEB-INF directory, right-click **PortalDecoratorStyle**, and select **Properties**.

   The PortalDecoratorStyle property sheet opens.

6. In the property sheet, click the **Context Parameter** tab.
7. Substitute the name of your decorator style sheet in the **Parameter value** field in place of PortalDecoratorStyle.
8. Save web.xml and redeploy your project.

7 Working with Portal Options

This chapter explains how to use portal options to control the appearance of the controls in the title bar of a portlet. It contains the following sections:

About portal options

A **portal option** is an image or text string that appears in the title bar of a portlet. Each portal option specifies a **link** that controls which **action** will be performed when the user clicks on the option. For example, a portlet title bar can provide buttons to allow the user to edit, minimize, or restore the portlet. Each button can have several images associated with it, so that the button image changes in response to user events.

Themes can alter the appearance of portal options. A theme can provide an image for each user event associated with a portal option, giving a consistent look when the user changes the theme.

Once you've defined a portal option, you can specify that this option is supported by a particular portlet.

Predefined options

exteNd Director ships with several predefined options:

Option	Description
Edit	Displays a screen to edit portlet preferences. This option puts the portlet in Edit mode, if the doEdit() method is supported; otherwise, it displays the default portlet preference sheet.
Help	Provides additional information about the content
Maximize	Maximizes the portlet so it occupies the entire browser page
Minimize	Minimizes this content generated by the portlet, leaving only the title bar visible
Remove	Removes portlet content from the page. This option applies only to personal pages.
Restart	Restarts the current pageflow 📖 A pageflow models a set of user interactions within a portlet. For more information, see the chapter on working with pageflows in the *Pageflow and Form Guide*.
Restore	Restores a minimized or maximized portlet to its normal window state

All of these options are defined in the **portal_core_resource.jar**.

Custom options

You can also create your own custom options. To do this, you use the Portal Option Wizard in the exteNd Director development environment.

📖 For details on using the Portal Option Wizard, see Chapter 18, "Creating Custom Options".

Files associated with portal options

Each option in a portal application must provide each the following files:

File(s)	Description
"Portal option descriptor file" on page 106	Describes the option. This file provides the following information: ◆ Display name ◆ Description ◆ Link ◆ Tool tip ◆ Text substitution if no image for the title bar ◆ **show by default** flag ◆ Order on title bar ◆ Set of theme-based images for the option The portal option descriptor file can have any name you like.
"Portal option image files" on page 108	Define images for the option.

The portal option descriptor is stored in the **portal-option** directory within a resource set.

 For more information about where files are located in a resource set, see the section on subdirectories for resources and Java classes in *Developing exteNd Director Applications*.

Portal option links

The link for a portal option can specify one or more of the following predefined portal URL query parameters:

Parameter	Description
urlType	Specifies type of portlet URL to be generated by the option. The following types are supported: ◆ **Action** URL that triggers a portlet action request, as defined by the Portlet specification ◆ **Render** URL that triggers a portlet render request, as defined by the Portlet specification ◆ **Remove** URL that triggers a request to remove content from the page. Although this option is not defined by the Portlet specification, it is recognized by the portal.
wsrp-mode	Specifies portlet mode to set. The following portlet-compliant modes are supported: ◆ **view** Portlet generates markup that reflects its current window state (the default mode) ◆ **edit** Portlet generates content that allows users to edit portlet preferences at runtime ◆ **help** Portlet provides help information **NOTE:** If wsrp-mode is not specified, the view mode is assumed.
wsrp-windowstate	Specifies portlet window state to set. The following portlet window states are supported: ◆ **normal** Portlet may share the portal page with other portlets and therefore assumes it has limited display space ◆ **minimized** Portlet renders minimal output or no output at all. Renders title bar with options, if enabled, but no content. ◆ **maximized** Portlet occupies most or all of the page real estate so it generates richer content than it would in the normal state
novl-regid	Specifies the portlet registration ID. The novl-regid parameter is required whenever the wsrp-mode, wsrp-windowstate, and urlType parameters are used.
novl-inst	Specifies the instance name for a portlet.

Here are some examples of portal option links that illustrate how the various parameters are used:

Option	Link example
Edit	`<link>?urlType=Render&amp;wsrp-mode=edit&amp;wsrp-windowstate=normal&amp;novl-regid=$COMPONENT_ID$&amp;novl-inst=$COMPONENT_INSTANCE_ID$</link>`
Help	`<link>?urlType=Render&amp;wsrp-mode=help&amp;wsrp-windowstate=normal&amp;novl-regid=$COMPONENT_ID$&amp;novl-inst=$COMPONENT_INSTANCE_ID$</link>`
Maximize	`<link>?urlType=Render&amp;wsrp-windowstate=maximized&amp;novl-regid=$COMPONENT_ID$&amp;novl-inst=$COMPONENT_INSTANCE_ID$</link>`
Minimize	`<link>?urlType=Render&amp;wsrp-windowstate=minimized&amp;novl-regid=$COMPONENT_ID$&amp;novl-inst=$COMPONENT_INSTANCE_ID$</link>`
Remove	`<link>?urlType=Remove&amp;novl-regid=$COMPONENT_ID$&amp;novl-inst=$COMPONENT_INSTANCE_ID$</link>`
Restart	`<link>?urlType=Action&amp;rlf=true&amp;novl-regid=$COMPONENT_ID$&amp;novl-inst=$COMPONENT_INSTANCE_ID$</link>`
Restore	`?urlType=Render&amp;wsrp-windowstate=normal&amp;novl-regid=$COMPONENT_ID$&amp;novl-inst=$COMPONENT_INSTANCE_ID$</link>`

The strings $COMPONENT_ID$ and $COMPONENT_INSTANCE_ID$ shown above are examples of portal replacement strings. Replacement strings allow you to include runtime, context-based information in a portal application. Replacement strings are keywords that reference things that can change dynamically at runtime, such as a user's current theme or the currently rendered component ID. When these keywords are detected at runtime, they are replaced based on information from the current HTTP request or the current portal context.

For more information about portal replacement strings, see Chapter 23, "Portal Replacement Strings".

Modes and window states

Mode and window state influence the content generated by portlets. For example, a portlet in edit mode might generate a list of properties that allows a user to customize the portlet's behavior. In help mode, a portlet might display a brief explanation of the its features. To determine the current mode for a portlet, the portlet container calls **getPortletMode()** on the request object.

When a user interacts with a portlet, its **window state** may change. For example, when the user minimizes a portlet, the state changes to minimized, which indicates that the portlet's title bar should be displayed, but not its content.

Window states are defined as constants on the EbiPortalContext interface.

Portal option descriptor file

The portal option descriptor file provides several elements that describe the portal option. Each option used by a portal application must have a separate portal option file.

Each element in the portal option descriptor file is described below:

Element name	Description
portal-option	The root node of the portal option
display-name	The name used to identify the option in the user interface of a portal application. This string can be localized.
description	The description of the option. This string can be localized.
link	The link associated with the option. The link can be a fully qualified or relative URL. It can also contain portal replacement strings. The link for a portal option can also use JavaScript. To use JavaScript in a link, start the link with: `"javascript:"`  For more information on portal replacement strings, see Chapter 23, "Portal Replacement Strings".
link-target	The target for the link. The target can specify the name of a frame, or specify one of the following standard values for the TARGET attribute of an HTML hyperlink: ◆ _top ◆ _self ◆ _new ◆ _blank

Element name	Description
tool-tip	The tip that appears when the user hovers over the link
text	Text for the link, if no image is provided for the **normal** event
show-by-default	Indicator that this option should show in the title bar by default even if the portlet does not include this option in its descriptor
order	Relative placement in the order of options from left to right. The smaller the integer the farther left it is positioned in the tool bar. If two options have the same order number, they are placed in random order. For example, if a particular portlet has 1,2,2, and 4 as the order numbers for its options, the option with number 1 is placed first and the one with the number 4 is placed last. The two options that have the number 2 are randomly placed between 1 and 4.
images	The set of images associated with a portal option. Each theme can supply a unique set of images for the various events associated with a portal option. These events map to JavaScript event handlers. The following event names can be specified for an image: ◆ normal ◆ onmouseout ◆ onmouseover ◆ onmousedown The path to each image can be a fully qualified URL or an URL that contains portal replacement strings. For more information on portal replacement strings, see Chapter 23, “Portal Replacement Strings”.
hide-states	One or more states in which this option should not be displayed. For example, when a portlet is maximized, you would typically want to hide the maximize option on the title bar. exteNd Director lets you specify any of the predefined portlet window states as hide-states. These states are: ◆ normal ◆ minimized ◆ maximized

Here is an example of a descriptor file for a portal option called maximize:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE portal-option PUBLIC "-//SilverStream Software, LLC//DTD
Portal Option 5.0//EN" "portal-option_5_0.dtd">
```

```
<portal-option>
  <display-name>Maximize</display-name>
  <description>Maximizes the portlet, giving it the entire browser
page</description>
  <link>?urlType=Render&amp;amp;wsrp-windowstate=maximized&amp;amp;novl-
regid=$COMPONENT_ID$&amp;amp;novl-inst=$COMPONENT_INSTANCE_ID$</link>
  <link-target>_new</link-target>
  <tool-tip>Maximize this portlet</tool-tip>
  <text>Max</text>
  <show-by-default>true</show-by-default>
  <order>25</order>
  <images>
   <image event="normal">$THEME_URL$/images/maximize.gif</image>
   <image
event="onmouseout">$THEME_URL$/images/maximize.gif</image>
   <image
event="onmouseover">$THEME_URL$/images/maximize.gif</image>
   <image
event="onmousedown">$THEME_URL$/images/maximize.gif</image>
  </images>
  <hide-states>
   <state>maximized</state>
  </hide-states>
</portal-option>
```

Portal option image files

Each theme can supply a unique set of images for the various events associated with a portal option. However, the file names for these images should be the same in each portal theme. By using a consistent naming convention across themes, you can ensure that each user event will have an image to display, regardless of which theme has been selected. For example, the Edit option associated with a portlet could have an image called edit_onmouseover.gif to handle the edit user event. In this case, each theme supported by the application would need to include a file with the same name. This would ensure that the onmouseover event always has an image to display for each theme.

Theme-enabling portal options

To theme-enable a portal option, you need to use the $THEME_URL$ replacement string keyword in the path to each option image in the portal option descriptor file, as shown below:

```
<images>
  <image event="normal">$THEME_URL$/images/edit.gif</image>
  <image
event="onmouseout">$THEME_URL$/images/edit_mouseout.gif</image>
  <image
event="onmouseover">$THEME_URL$/images/edit_mouseover.gif</image>
  <image
event="onmousedown">$THEME_URL$/images/edit_mousedown.gif</image>
</images>
```

Each supported theme would provide a unique set of images with these names, as in this example for the BasicBlue theme:

```
...\portal-theme\
        \BasicBlue
            theme.xml
            theme.css
            \images
               edit.gif
               edit_mousedown.gif
               edit_mouseout.gif
               edit_mouseover.gif
               ...etc.
```

Specifying options for portlets

Once you've defined a portal option, you can specify that this option is supported by a particular portlet. Portlet options are specified in **novell-portlet.xml**, the Novell extension to the portlet deployment descriptor.

A portlet can reuse the complete definition of one or more portal options simply by specifying the IDs for these options:

```
<portlet name="AppletLauncher">
   <supported-option>edit</supported-option>
   <supported-option>help</supported-option>
...
</portlet>
```

You can also specify options for portlets at registration time.

Working with the portal option API

The exteNd Director API provides two interfaces for working with options:

- EbiOptionManager
- EbiOptionInfo

EbiOptionManager provides methods for accessing EbiOptionInfo objects. EbiOptionInfo provides methods for retrieving various kinds of information about a portal option, including the property setting values specified in the portal option descriptor.

To access an EbiOptionManager object, you need to use the getOptionManager() method on the EboFactory class for the Portal subsystem.

8 Working with Page Categories

This chapter explains how portal administrators can use page categories to filter access to portal pages. It covers the following topics:

Page categories and portal navigation

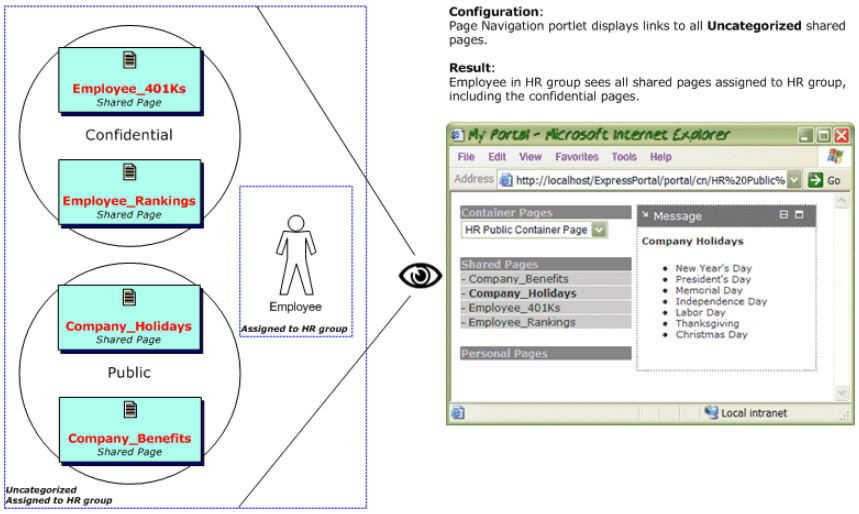
Portal administrators can use page categories in conjunction with the Page Navigation and Page Header portlets to filter access to individual pages within ACL groups. This level of control is useful when you need to override access permissions assigned to users or groups—for example, if a user is a member of two groups, but should not be allowed to view specific shared pages assigned to one of the groups.

The Page Navigation and Page Header portlets can facilitate navigation of a portal application from a single container page. When placed on a container page, these portlets can be configured to display navigation links to pages available to the current user based on categories.

A usage case

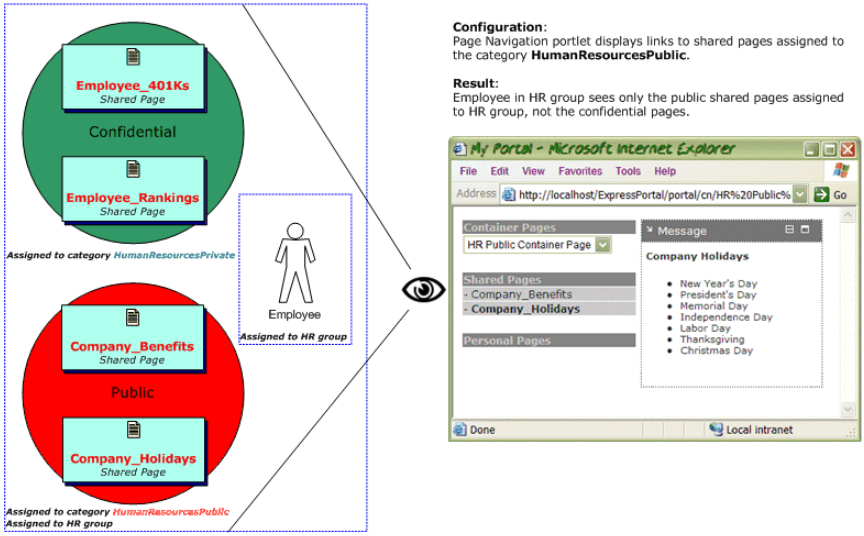
Consider this scenario: The Human Resources department in a corporation has an ACL group called **HR** and a portal that contains a mix of confidential and public pages. All employees in a corporation are assigned to the HR ACL group, but only a small subset of employees—authorized staff in the Human Resources department—can view the confidential pages.

When all Human Resources portal pages are uncategorized, they can be viewed by every employee, as shown:



To restrict access to the confidential portal pages, the portal administrator creates two page categories: **HumanResourcesPublic** for the public HR pages and **HumanResourcesPrivate** for the private HR pages.

The administrator then configures the Page Navigation portlet on a container page to display links only to pages assigned to the category HumanResourcesPublic. By assigning this container page to the unauthorized employees in the HR group, the administrator allows them to view public HR pages, but restricts their access to the private ones, as shown:



Filtering page navigation by category

Here are the steps a portal administrator must follow to filter page navigation by category:

1. Create one or more page categories, as described in "Creating page categories" on page 114.
2. Assign categories to container pages and shared pages, as described in "Assigning categories to pages" on page 117.
3. Configure the Page Navigation or Page Header portlet on a container page to restrict access to pages by category, as described in "Filtering navigation links by category" on page 118.
4. Assign the container page or shared page to the appropriate users or groups.

Creating page categories

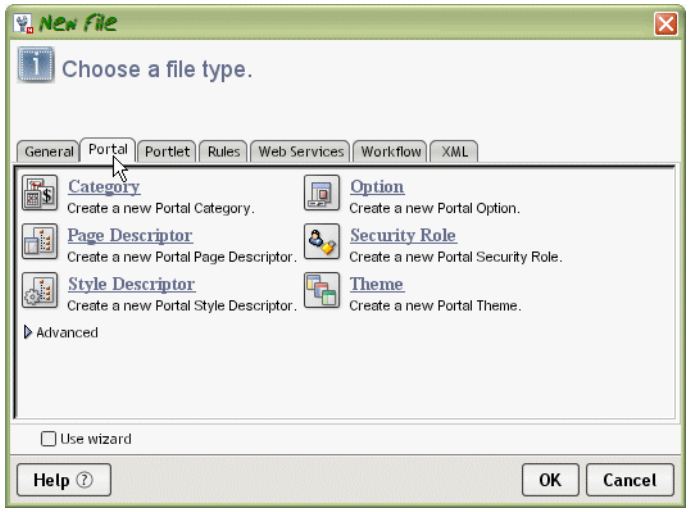
This procedure describes how to create page categories in exteNd Director.

➢ **To create a page category:**

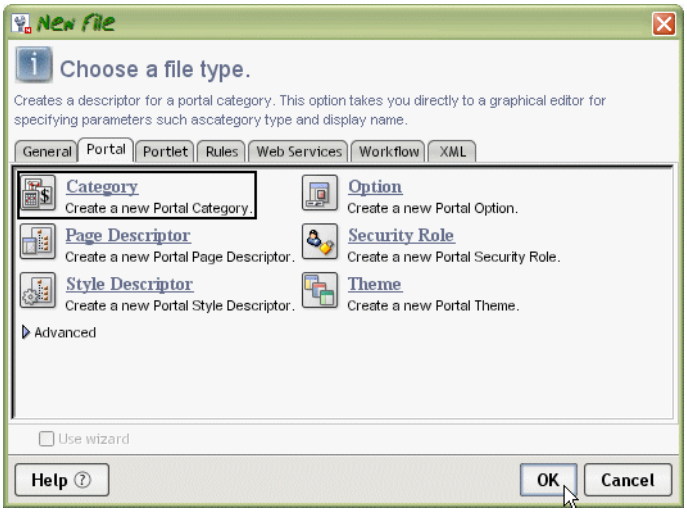
1. With your project open in the exteNd Director development environment, select **File>New>File**.

   The **New File** dialog opens.

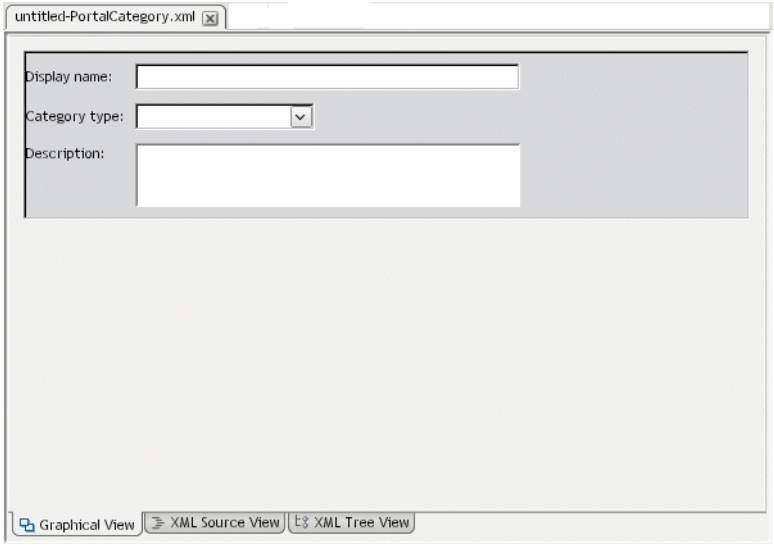
2. In the New File dialog, select the **Portal** tab.



3 Select **Category** and click **OK**.



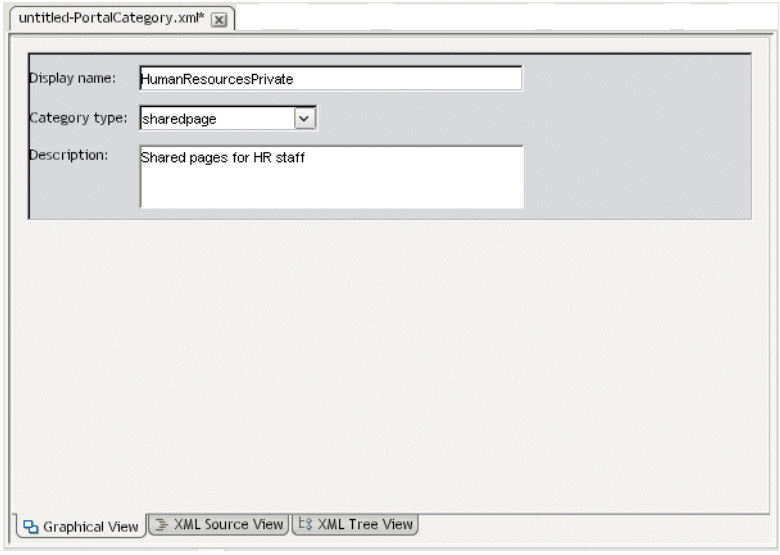
An untitled portal category descriptor opens in graphical view in exteNd Director.



4 Enter the following information in the portal category descriptor fields:

Field	What to specify
Display name	Enter a meaningful display name for the category
Category type	From the drop-down menu, select the type of page you want to categorize: ◆ **page** for personal pages ◆ **sharedpage** for shared pages ◆ **containerpage** for container pages
Description	Enter a description of the category

For example, here are the values entered for a category that will be used to restrict access to shared pages designed only for Human Resources staff:



5 Select **File>Save** to save the category descriptor.

The **Save As** dialog opens. By default, the category will be saved as an XML descriptor file in the **portal-category** folder of the resource set. The file name will be the same as the category name.

6 Keep the defaults and click **Save** in the Save As dialog.

In the example above, the category descriptor is saved in a file called **HumanResourcesPrivate.xml**. Here is what the descriptor looks like:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE portal-category PUBLIC "-//SilverStream Software,
LLC//DTD Portal Category 5.0//EN" "portal-category_5_0.dtd">
<portal-category>
     <display-name>HumanResourcesPrivate</display-name>
     <description>Shared pages for HR staff</description>
     <category-type>sharedpage</category-type>
</portal-category>
```

Now you are ready to assign the category to one or more pages, as described in "Assigning categories to pages" on page 117.

Assigning categories to pages

The following procedure describes how to assign categories to pages. You must assign the category to pages of the same type as the category. For example, if you created a category for shared pages, you can assign that category only to shared pages—and not to personal or container pages.

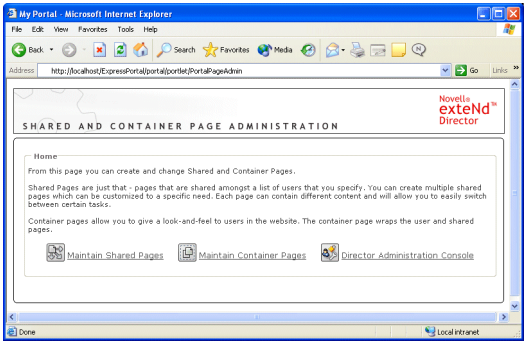
Only portal administrators can assign pages to categories.

➢ **To assign categories to pages:**

1 Start the Portal Administrator tool.

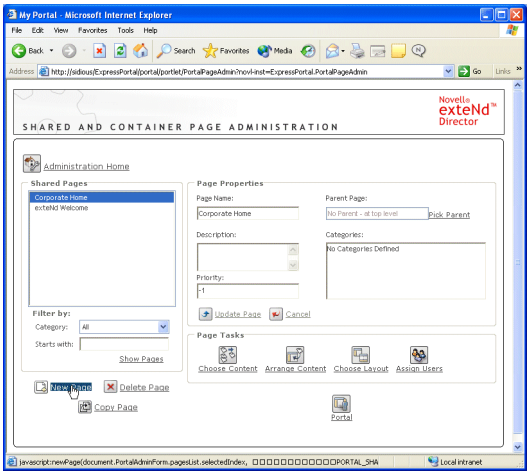
For instructions, see the section on starting the Portal Administration tool.

The Shared and Container Page Administration page opens in your browser:



2 Click **Maintain Shared Pages** or **Maintain Container Pages**, depending on the type of page you want to categorize.

The appropriate section of the Portal Administration tool opens in your browser. Here is an example of a shared page administration page:



3 If you need to search for the page of interest, enter search criteria and click **Show Pages**.

TIP: You can streamline your search by selecting a category of pages or by entering the initial letters of the page name in the **Starts with** field.

4 Select the page of interest from the list of pages returned, then check the categories you want to assign to the page, as in this example:

5 Click **Update Page**.

The page appears under its new category (or categories) in the list of pages.

6 Select **Administration Home** and leave the Portal Administration tool running for the next procedure, "Filtering navigation links by category" on page 118.

Filtering navigation links by category

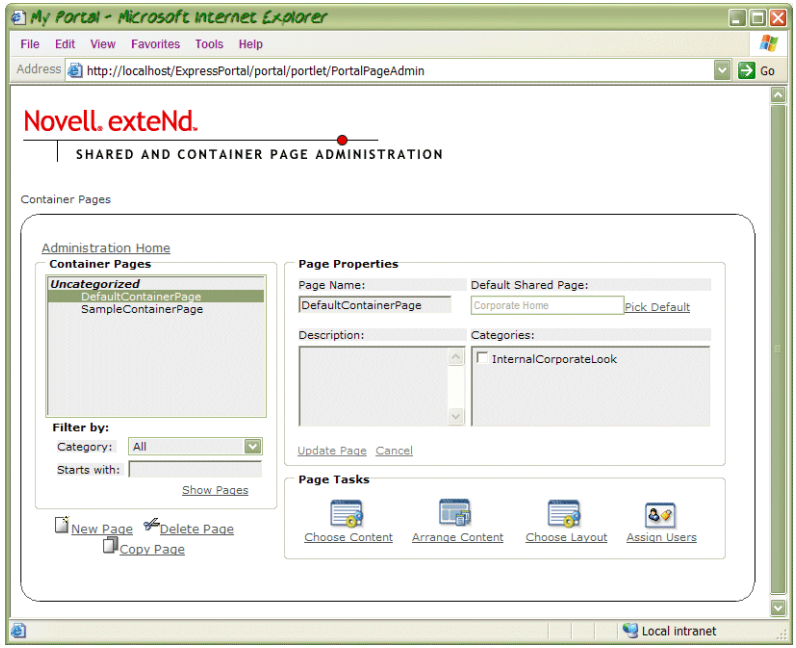
This procedure shows you how to configure a Page Navigation portlet or Page Header portlet to filter navigation links by categories on a container page.

Your Portal Administration tool should be running in your browser, displaying its home page.

➢ **To display navigation links by category:**

1 In the Portal Administration home page select **Maintain Container Pages**.

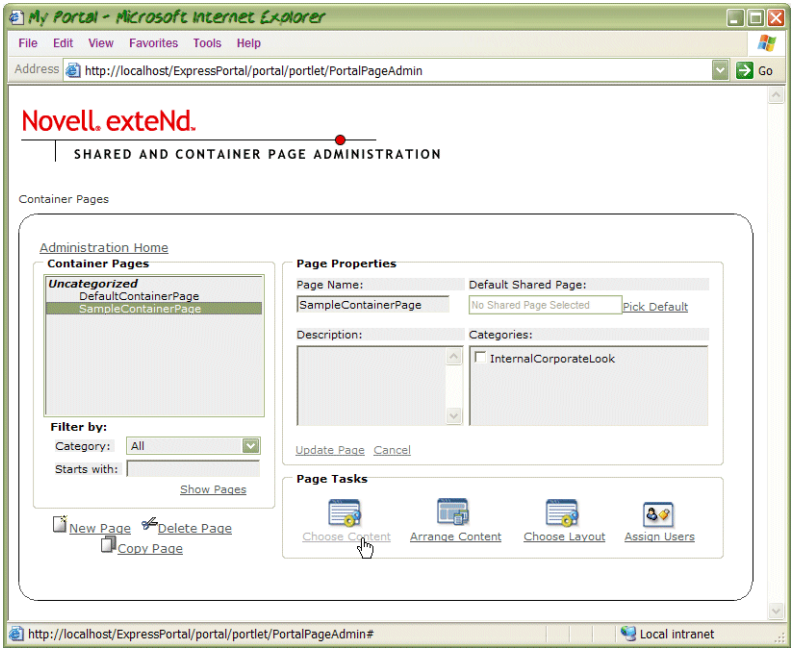
The container page administration page opens in your browser, as in this example:



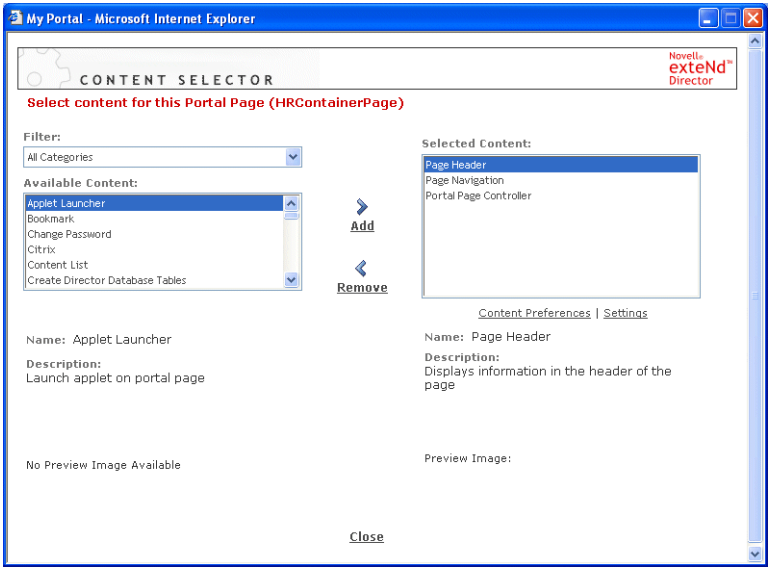
2 If you need to search for the page of interest, enter search criteria and click **Show Pages**.

TIP: You can streamline your search by selecting a category of pages or by entering the initial letters of the page name in the **Starts with** field.

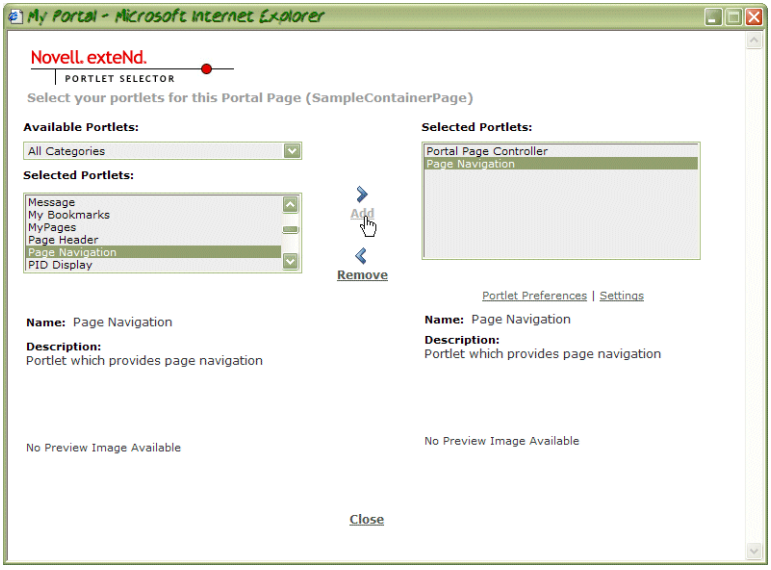
3 Select the desired container page from the list of pages returned and click **Choose Content** in the Page Tasks list:



The Portlet Selector portlet appears in your browser.

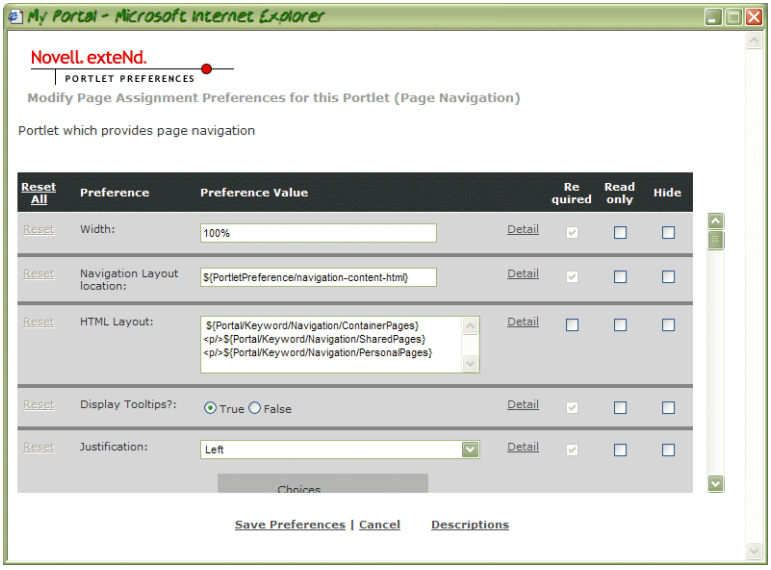


4 If the container page does not contain a Page Navigation or Page Header portlet, add one of these portlets by selecting it from the Available Portlets list and clicking **Add**.



5 Select the Page Navigation or Page Header portlet you just added in the Selected Portlets list, then click **Portlet Preferences**.

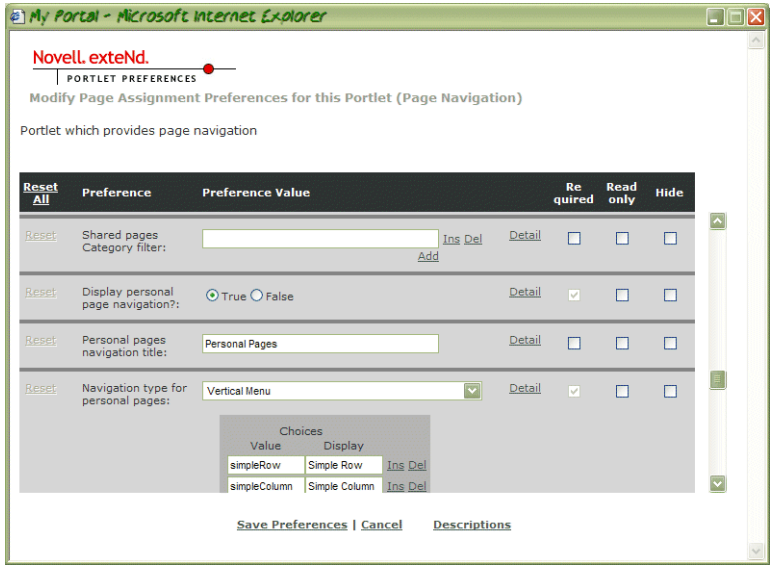
The Portlet Preferences portlet opens in a new browser window.



6 Scroll to the appropriate category filter preferences:

If you want to filter	Scroll to
Container pages	Container pages category filter
Shared pages	Shared pages category filter
Personal pages	Personal pages category filter

For example, here is the Shared pages category filter preference:



7 Enter the names of the categories to use for filtering, one per text field.

The categories you enter determine which pages can be accessed from the navigation links of the Page Navigation or Page Header portlet on this container page.

Follow these guidelines:

Pages to display	What to specify
All pages of the specified type	Leave the field blank. Enter no categories.
Uncategorized pages of the specified type	Enter **uncategorized** in one of the text fields
Pages of the specified type that belong to one or more categories	Enter the name of each category in a separate text field.

TIP: To add text fields, follow these steps:

To	Do this
Append a text field to the end of the list	Click **Add**.
Insert a text field at a specific location inside the list	Click **Ins** next to the field where you want to insert a field. The new field will be inserted above the selected field.

8 Click **Save Preferences**.

Assigning container pages to users and groups

See the section on assigning pages to users and groups in the *Portal Guide*.

9 Working with PID Pages

This chapter describes basic techniques for writing PID pages. It includes these topics:

About PID pages

A portal application can contain *Portal Page ID (PID) pages*. PID pages are pages that include s3-component tags. PID pages can contain HTML or XML.

Where to put your PID pages PID pages must be packaged in a resource set WAR, along with other resources required for the application. The PID pages must be located in the **portal-page** directory within the resource set.

PID page descriptors Each PID page must have at least one XML descriptor. Each descriptor specifies the name of the HTML or XML file that provides content for the page, as well as additional parameters that control how the page will look and behave at runtime. The XML descriptor for a PID page must also be located in the **portal-page** directory within the resource set.

Here is an example of a descriptor for a portal page that provides HTML content. It specifies the name of the HTML file that provides content, as well as a display name and a description:

```
<portal-page>
 <display-name>Hello World</display-name>
 <description>Hello World Page</description>
 <file-name>HelloWorld.html</file-name>
</portal-page>
```

Here is an example of a descriptor for a portal page that provides XML content. It specifies the name of the XML file as well as the name of a portal style that defines the XSL file that will be used for translation. This descriptor also specifies security role mappings for the page:

```
<portal-page>
 <display-name>Hello World</display-name>
 <description>Hello World Page</description>
 <category>UserPages</description>
 <mime-type>text/xml</mime-type>
 <style-name>HelloWorld</style-name>
 <file-name>HelloWorld.xml</file-name>
 <run-role-map>
  <role-name>manager</role-name>
  <role-name>administrator</role-name>
 </run-role-map>
</portal-page>
```

The descriptor file for a portal page must have an XML extension. To indicate which portal page a descriptor applies to, you may want to include the file name of the portal page in the name of the descriptor. For example, if the portal page is stored in a file called **HelloWorld.html**, you might name the descriptor file **HelloWorld.html.xml**.

URLs for PID pages To display PID pages, the portal's base servlet breaks the request URL down into several components:

- The **portal pages path** specifies the path to portal pages in the WAR. This portion of the URL is controlled by the PortalPathPagesKey context parameter in the web.xml for the WAR:

  ```
  <context-param>
   <param-name>PortalPathPagesKey</param-name>
   <param-value>pages</param-value>
   <description />
  </context-param>
  ```

- The **PID page name** identifies the requested item in the WAR. The PID page name specifies the name of the XML descriptor (without the XML extension). By convention, the name of the XML descriptor is often the same as the name of the portal page file. However, it need not be.

For example, suppose you want to access a portal page that has a descriptor file called HelloWorld.html.xml. The page is deployed to the Express Portal. In this case, you could use the following URL to display the page:

```
http://localhost/ExpressPortal/portal/pages/HelloWorld.html
```

Building PID pages that contain HTML

You can develop portal pages as typical HTML pages with graphics, tables, forms, and style sheets. You can use the text editor provided with development environment, or any other HTML editor.

To write the HTML, you need to know how to include portlets and portal components on the page.

Portlet tag

When you want to use a portlet in a PID page, you specify a special tag recognized by exteNd Director. You put the tag in the HTML where you want the portlet to insert its content.

The format for a portlet tag is:

```
<s3-component id="portletID/componentID" instance="instanceID" />
```

where arguments are as follows:

Argument	Description
portletID/componentID	The ID given to the portlet or component. For portlets, the ID is the portlet registration ID. For components, the ID is the name given to the XML file (without the extension) that describes the component. For example, if the name of the XML file for a component is MyComponent.xml, the component name is MyComponent.
instanceID	A name used to identify the portlet or component on the page. This name uniquely identifies this instance of the portlet or component. A good practice is to prefix the instance ID with the name of the page that the portlet or component is assigned to.

For example, to put a portlet in a table cell on your page, you would write HTML like this:

```
<td height="200">
   <s3-component
     id="PhoneList"
     instance="CorpHome_MyPhoneList" />
</td>
```

You can also specify *custom parameters* that alter the behavior of the portlet or component. A custom parameter is an arbitrary pairing of a name and a value:

```
<td height="200">
   <s3-component
     id="MyPortlet"
     instance="CorpHome_myportlet" MYATTRIBUTE="myvalue" />
</td>
```

The code that implements the component must check for the value of the parameter (in this case MYATTRIBUTE) and modify its behavior accordingly.

Building PID pages that contain XML

XML is a practical way to render text-based pages with a simple layout. It is particularly useful when the page is made up of portlets or components.

You determine the XML elements you want to represent on your page. You also define an XSL style sheet for rendering the elements into HTML. The base servlet combines the two to display the portal page in the client browser.

You can use the same XSL for some or all of your PID pages to give your portal a consistent appearance. It's easy to tweak the appearance by changing the XSL, which affects all pages that use it, or by providing another XSL specification for any or all of the pages.

10 Developing a Wireless Application

This chapter describes how to build exteNd Director portal applications that support wireless devices. It contains the following sections:

About wireless applications

exteNd Director allows you to easily extend your applications to the constantly expanding variety of devices used by the mobile community, such as handheld computers and cellular telephones. Any portlet that produces XML output can use exteNd Director's wireless capabilities to produce output suitable for wireless devices. You do not need to modify portlet output according to the size or type of display when implementing a portlet class.

The process of enabling applications for wireless devices consists of two steps: device-specific **rendering** and device-specific **pagination**. Although the steps are intended to work together, either one can be used separately. For example, a search engine can use device-specific pagination to present a specific number of hits per page for Web browsers on personal computers as well as wireless devices.

Device-specific rendering

You can designate any XML portlet in your exteNd Director application as wireless-enabled, which allows the Portal Aggregator to automatically identify wireless client devices and render the portlet's XML output in device-specific formats such as WML, cHTML, or Web Clipping.

The Portal uses a list of profiles for various devices that is stored in the portal's resource set. Updates to the list will be made available for download from the Novell Web site as new devices and formats appear on the market.

Application content that is already small enough to fit on a single page may require nothing more than rendering in the appropriate device-specific format. In other words, reducing the width of a page by increasing the length may or may not be acceptable. You must decide how much vertical scrolling (if any) you consider acceptable for a single page.

Device-specific pagination

Some portlet content is too large to fit on a single page. Thus, you can designate any XML portlet in your exteNd Director application for *device-specific pagination*, which slices the portlet's XML output into smaller blocks and adds navigational aids such as tables of contents and previous-next buttons.

Using a graphical editor integrated with development environment, you can drag and drop pieces of a sample portlet content file into a *data definition* that models both layout and navigation. As you work, you can instantaneously preview the output produced by your data definition.

At runtime, the *device transcoding* engine applies the data definition to the portlet's output before sending it to the Portal Aggregator.

Sample portlet

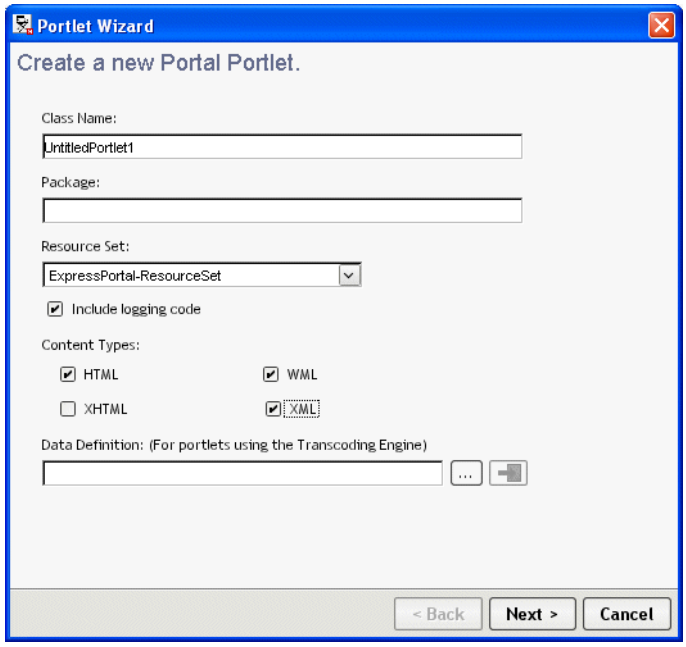
Your exteNd Director software includes a sample wireless-enabled portlet named **StockQuotePortlet** that uses device-specific rendering. You can use this portlet to begin learning about how to provide wireless support in your applications.

Creating a wireless-enabled portlet

You can create a wireless-enabled portlet using the Portlet Wizard, which does much of the work for you.

➢ **To create a new wireless portlet:**

1. Invoke the Portlet Wizard, as described in the section on using the Portlet Wizard.



2. Under Content Types, select **XML** and **WML**.
3. If you already have a device transcoding **data definition** for the portlet, specify it here. Otherwise, click **Finish** and turn to "Creating a device transcoding data definition" on page 138.

Wireless-enabling an existing portlet

To wireless-enable an existing portlet, you will need to modify and/or create several files as described in the following procedures.

➢ **To modify the portlet source file:**

1. Edit the portlet source file.
2. In the doView() method, set the MIME type to **MIME_TYPE_XML**, as shown below:

```
response.setContentType(com.novell.afw.portlet.api.
EbiPortletConstants.MIME_TYPE_XML);
```

NOTE: If you want to develop a wireless portlet that supports transcoding, you would need to set the MIME type to MIME_TYPE_DEVICE_PROFILING.

➢ **To modify the portlet descriptor:**

1. Open the portlet descriptor.
2. In the style section, enter the name of the **Data Definition** file, if you plan to support transcoding. The data definition file does not have to exist at this point.

```
<style>
 <name>MyPortletDefault</name>
 <display-name>MyPortlet Default Style</display-name>
 <user-agent>
  <device-name>Generic_HTML</device-name>
  <file-name>
   $RESOURCE_SET$/portal-style/MyPortlet_HTML.xsl
  </file-name>
 </user-agent>
 <user-agent>
  <device-name>Generic_WML</device-name>
  <file-name>
   $RESOURCE_SET$/portal-style/MyPortlet_WML.xsl
  </file-name>
 </user-agent>
 <data-definition>
  $RESOURCE_SET$/portal-data-definition/PhoneList
 </data-definition>
    </style>
```

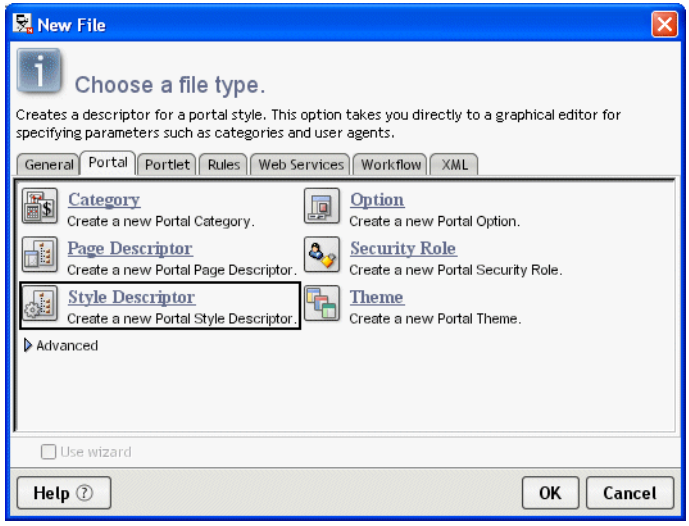
3. In the style section, examine the selected styles. If the portlet already has a portal style, proceed to "To modify the portal style descriptor:" on page 133. Otherwise, choose one of the following:
   - Select **GenericStyle** and proceed to "Creating a device transcoding data definition" on page 138.
   - Create a new portal style descriptor.

4 In the auto-register section, add the category **Wireless**.

```
<auto-register enabled="true">
  <category>Wireless</category>
</auto-register>
```

You can also set the category in the Portlet Management section of the DAC.

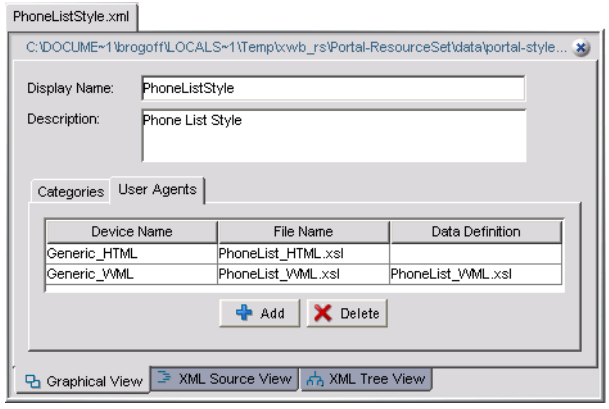
5 If you choose to create a new portal style descriptor at this point, invoke the Portal Style Descriptor Wizard.



6 Name the descriptor and proceed to "To modify the portal style descriptor:" on page 133.

➢ **To modify the portal style descriptor:**

1 Open the descriptor and click the **User Agents** tab.



2 If there is not already a user agent named **Generic_WML**, click the **Add** button.

3 A user agent specifies how the portal style is rendered for a particular user environment by mapping a device profile to an XSL style sheet.

Set the **Device Name** (`<device-name>`) to **Generic_WML**. The device name is the name of a device profile that defines the user environment.

Set the **Data Definition** (`<dp-file-name>`) to an XSL style sheet.

NOTE: The Data Definition column in the portal style descriptor GUI should not be confused with a device transcoding data definition.

If you do not have a style sheet for the portlet, use one of the generic style sheets provided in the `portal-style` directory such as `Generic_WML.xsl`.

```
<user-agent>
    <device-name>Generic_WML</device-name>
    <dp-file-name>Generic_WML.xsl</dp-file-name>
</user-agent>
```

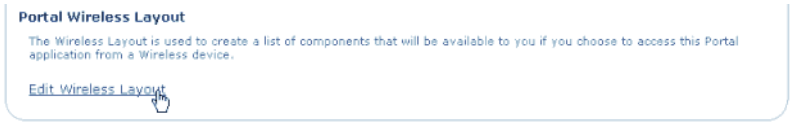
For more information about device profiles, see "Using the device profile editor" on page 135.

Using the Wireless Layout Manager

The **Wireless Layout Manager** allows you to define a **wireless profile**. Using a Web browser, you can select from the available wireless-enabled portlets and set the order in which they are displayed. The list contains portlets that have the category **Wireless**.

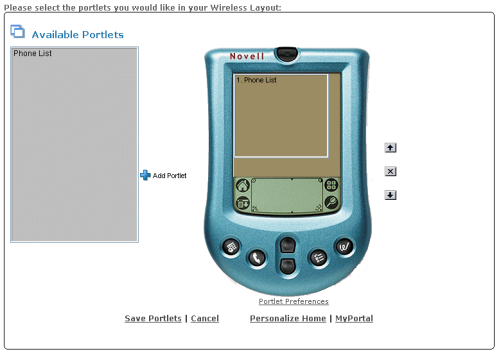
➢ **To define a wireless profile:**

1 Open the Portal Personalizer, scroll to the **Portal Wireless Layout** section and click **Edit Wireless Layout**.



The Wireless Layout Manager opens in your browser.

2 In the Wireless Layout Manager, select any available portlet and click **Add Portlet**.



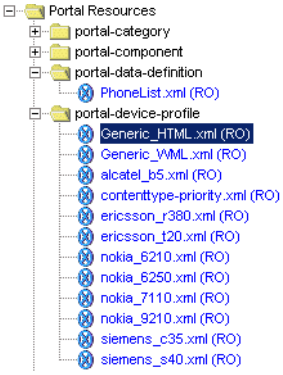
3 To see the wireless portlets in action, click **Save Portlets**, go to **MyPortal**, and click **MyWirelessProfile**.

Using the device profile editor

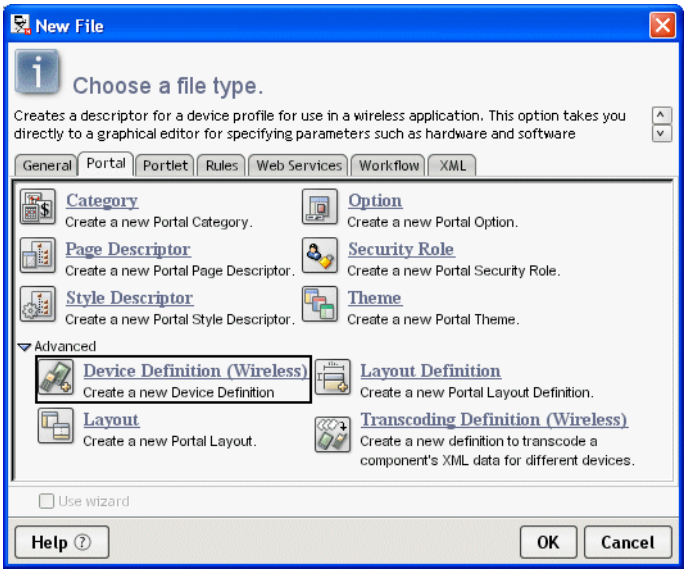
exteNd Director provides a graphical editor in development environment for creating or editing device profiles.

➢ **To start the device profile editor:**

1 Double-click an existing device profile.

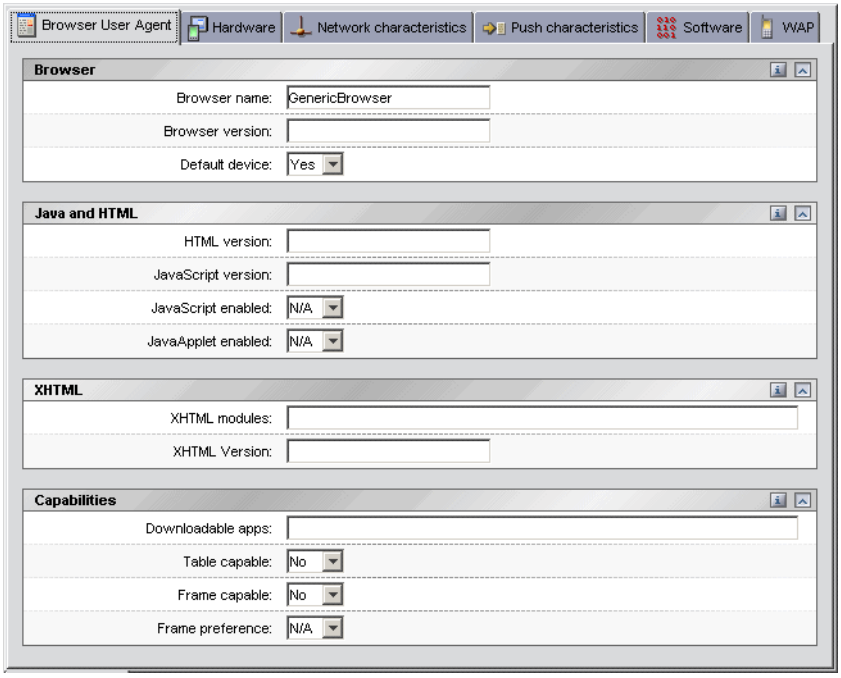


2 Select **File>New>Portal>Device Definition (Wireless)**.



➢ **To use the device profile editor:**

- The following specification describes the fields in the device profile editor: http://www1.wapforum.org/tech/terms.asp?doc=WAP-248-UAProf-20011020-a.pdf



There is a slight difference between the WAP specification and the exteNd Director editor—because Wapforum uses RDF to describe information and exteNd Director uses descriptor tags. This makes it possible to use a W3C schema for tag-completion in the XML source view.

Creating a device transcoding data definition

About transcoding data definitions

A transcoding data definition is an XML file that identifies certain tags in portlet output that can be used to specify how to slice the output data and add navigational aids. The first step is to capture a representative sample of portlet output, as described in "Creating a reference file" on page 138.

Once you have an output sample to work from, consider it to be a table with columns and rows. The next step is to identify **which tag is the row separator**. When you have defined the row separator, you can map the output data onto the following objects:

Object	Description
List block	A view that includes all rows
Content block	A view that includes only one row
Element	A column
Link	Connects a list block to a content block

Creating a reference file

To create a reference file, you need to capture a sample of your actual XML portlet output in a file. Any XML-based portlet can print its XML to the console by doing the following:

```
// Create the XML Document
Document newDoc = EboXmlHelper.getNewDocument();

// Add elements to the document
...

// Print the document to the console
com.sssw.fw.util.EboXmlHelper.printDOMTree(newDoc);
```

Save the text file in the **portal-data-definition** folder using a name that does not conflict with the actual data definition file. For example, some XML output from the PhoneList portlet is shown below. For convenience, you can copy this output and paste it into a text file named **PhoneListReference.xml**.

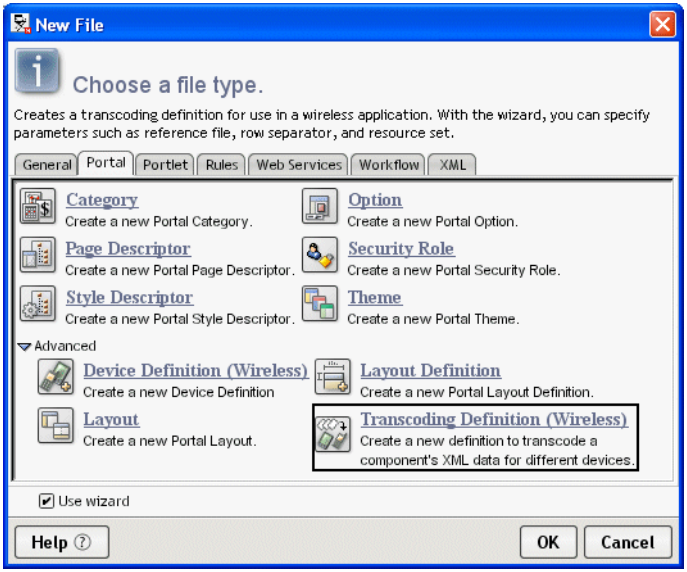
```
<?xml version="1.0"?>
<phonelist compInstance="a63974dcac7e4f28bd2c49ab7b1921d2"
compid="PhoneList" post-url="http://localhost/Director/Portal/main
/comp/PhoneList">
 <results querystring="c">
  <employee>
```

```
    <first-name>Samuel</first-name>
    <last-name>Craddock</last-name>
    <phone-number>(617) 343-6505</phone-number>
    <email>scraddock@silverdemo.com</email>
   </employee>
   <employee>
    <first-name>John</first-name>
    <last-name>Chester</last-name>
    <phone-number>(617) 343-6506</phone-number>
    <email>jchester@silverdemo.com</email>
   </employee>
   <employee>
    <first-name>Lynn</first-name>
    <last-name>Campbell</last-name>
    <phone-number>(617) 343-6507</phone-number>
    <email>lcampbell@silverdemo.com</email>
   </employee>
  </results>
</phonelist>
```

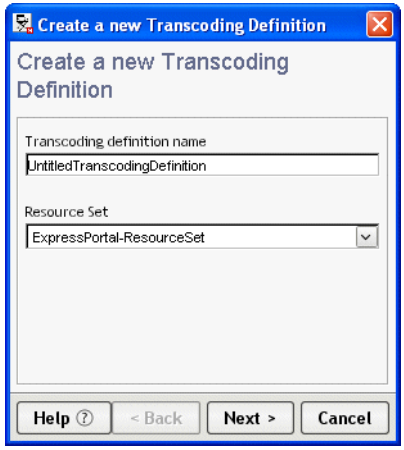
Creating a data definition file

exteNd Director includes a wizard for creating transcoding data definition files.

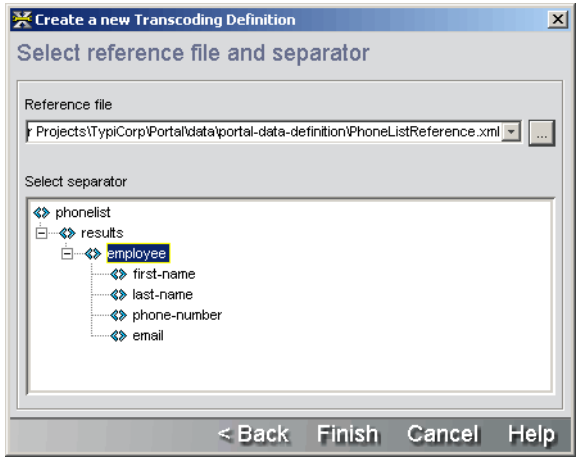
1. In development environment, select **File>New** from the menu and select the **Portal** tab.
2. Select **Transcoding Definition (Wireless)** and click **OK**.



3 Name your transcoding definition file, specify which resource set to add the files to (if necessary), and click **Next.**



4 Select your reference file. The wizard automatically opens the file and displays the XML tag structure.



5 Select the row separator. If you are using the sample portlet reference file (**PhoneListReference.xml**), select **Employee**.

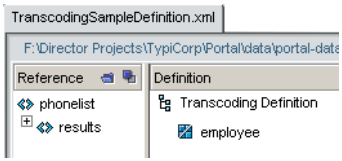
6 Click **Finish**.

Editing a data definition file

This section uses the sample portlet data definition file **PhoneList.xml** and reference file (**PhoneListReference.xml**) to illustrate the procedure.

➢ To open a reference file:

1. In the data definition editor, select the **Definition** tab. It has two panes: Reference and Definition.
2. Open a reference document (unless one is already open). Right-click the **Reference** pane and select **Open a reference file**.



NOTE: The sample portlet reference file is named **PhoneListReference.xml**.

The Reference Pane displays the structure of the data using a tree view. You can click plus (+) to expand and minus (-) to collapse branches of the view.

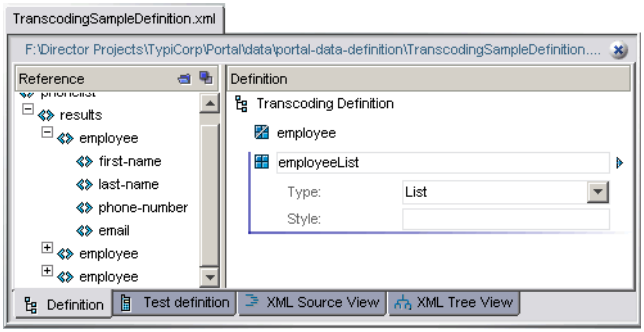
➢ To define a row separator:

If you have already defined a row separator (as described in "Creating a data definition file" on page 139), skip this procedure.

- Drag a tag from the Reference Pane to the empty block in the Definition Pane.

➢ To create blocks:

1. Select any item in the Definition Pane.
2. Right-click and select **Add Block**.



NOTE: If you are using the sample portlet reference file, create two blocks: a List block named **employeeList** and a Content block named **employeeDetails**.

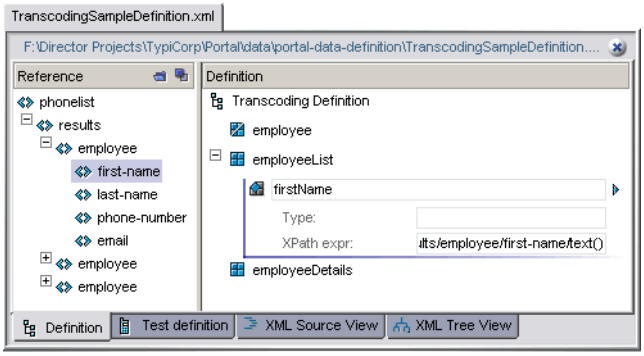
3. Name the new block and set the type to **List** or **Content**.

4 You can specify a **Style** to override the default style for this block only.

NOTE: Click the small triangle ( ) to the right of the block name to keep the block open when you move the cursor to another object. The triangle changes color and rotates to point downward ( ) to indicate that the block is locked open. By default, the editor closes objects (displays only the name of the object) when you move the cursor elsewhere.

➢ **To create elements:**

1 Select an element in the Reference Pane and drag it onto a block in the Definition Pane.



NOTE: If you are using the sample portlet reference file, add the following elements:
To **employeeList**: firstName, lastName
To **employeeDetails**: firstName, phoneNumber, email

2 Name the new element.

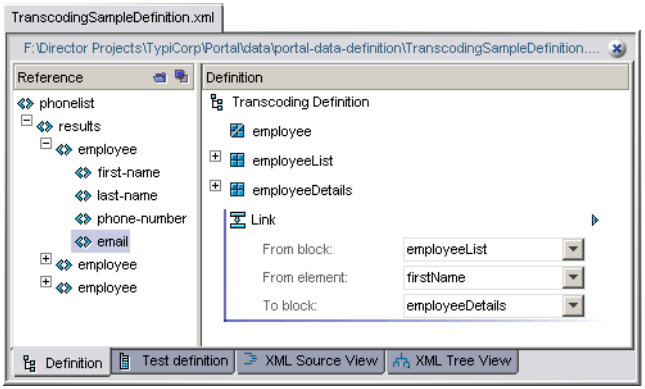
3 The **Type** field provides a way to pass user defined information to the style sheet.

4 The **XPath expr** field is an expression in XPath (XML Path Language) for addressing parts of an XML document. You can use this field to remap the element.

📖 For more information about XPath, see the language specification (http://www.w3.org/TR/xpath).

➢ **To create a link:**

1. Select any item in the Definition Pane.
2. Right-click and select **Add Link**.



NOTE: If you are using the sample portlet reference file, set:
From block to **employeeList**
From element to **firstName**
To block to **employeeDetails**

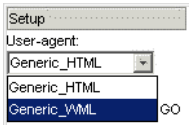
3. Set the **From block** attribute to a List block.
4. Set the **From element** attribute to an element in the List block.
5. Set the **To block** attribute to a Content block.

Testing a data definition

The Test View allows you to view the output produced by the transcoding definition you just created.

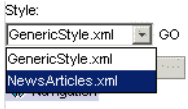
➢ **To set up the test view:**

1. In the data definition editor, select the **Test definition** tab. It has three panes: Setup, Navigation, and Result.
2. In the Setup Pane, select one of the available user agents from the dropdown list. The list contains all of the registered device types in the portal.



NOTE: If you are using the sample portlet data definition, set the User-agent to **Generic_WML**.

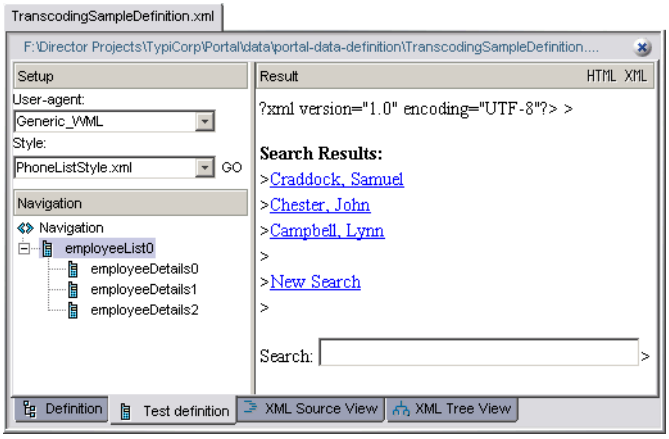
3 Select one of the available styles from the dropdown list. The list contains all of the styles that support the selected user agent.



NOTE: If you are using the sample portlet data definition, set the Style to **PhoneListStyle.xml**.

➢ **To execute the test:**

1 When you have made your selections, click **GO**.



2 Examine the Navigation Pane. Click plus (+) to expand and minus (-) to collapse branches of the view.

3 Click any item to see the transcoding engine output in the Result Pane.

TIP: For a graphical view of the rendered data, set up a Wireless Profile (as described in "Using the Wireless Layout Manager" on page 134) and add the PhoneList portlet to your profile.

II Portlet Concepts

Provides an overview of portlet concepts

11 About Portlets

This chapter introduces basic portlet concepts and describes how exteNd Director implements portlets. It covers these topics:

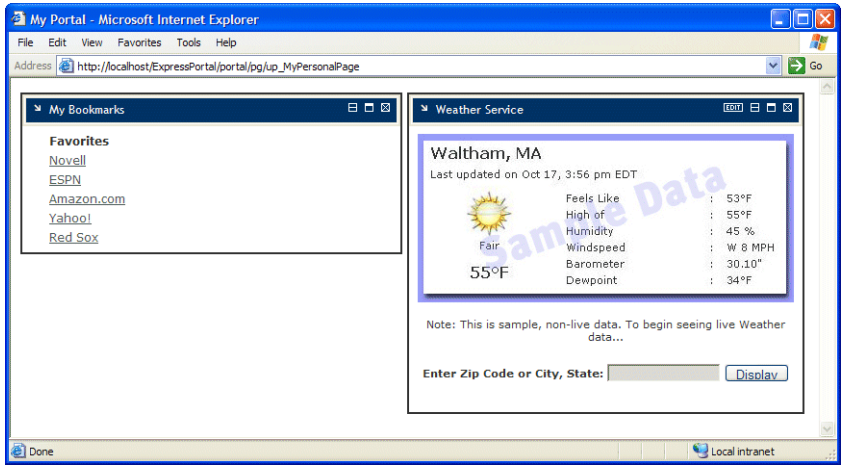
What is a portlet?

A *portlet* is a specialized Java class that processes requests from Web clients and generates dynamic content on a portal page. Portlets are defined in the Java Portlet 1.0 specification.

You can think of portlets as pluggable user interface elements that provide a presentation layer for portal applications. Users can personalize the content and appearance of portlets, based on preferences set by an administrator or other user.

In exteNd Director you run portlets in several ways:

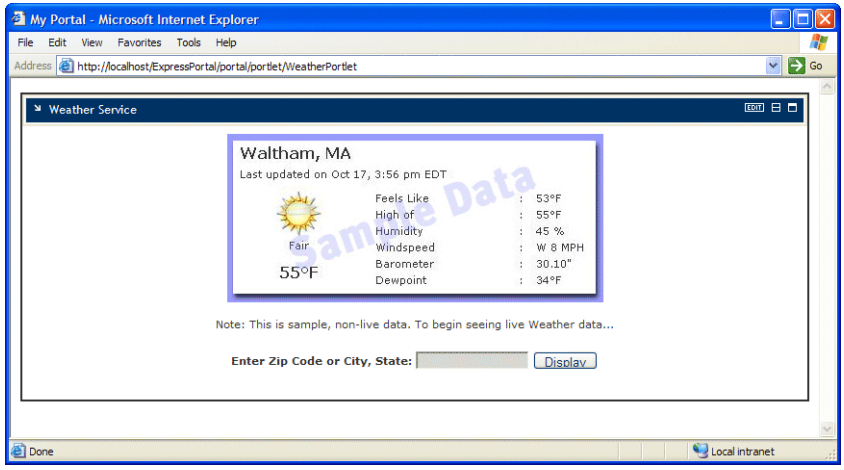
- Add one or more portlets to a portal page, then display the page in a Web browser, as in this example:



In this example, an exteNd Director portal page displays the content of two portlets: **My Bookmarks** and **Weather Service**.

For more information, see the chapter on working with portal pages.

- Run individual portlets directly in a Web browser. Here is an example of the Weather Service portlet running directly in a browser:



Based on servlet technology

To a large extent, portlets are based on servlet technology, as described in "The relationship between portlets and servlets" on page 152. For example, both portlets and servlets are managed by a container object and both use a request/response paradigm to communicate with Web clients.

To get a feel for how portlets operate under this paradigm, consider the case of a portlet application running in a Web browser. When a user interacts with content produced by one of the portlets—perhaps by following links or clicking buttons—the browser sends a request to the portal. The portal then forwards the request to the portlet container which repackages it as an *action request* or *render request* for execution by the appropriate portlet. You can learn more about portlet request handling in "Portlet requests and responses" on page 160.

exteNd Director provides support for developing and running Java Portlet 1.0-compliant portlets, but in addition provides extended functionality, as described in "exteNd Director extensions to Java Portlet 1.0" on page 151.

The Portlet specification

The Java Portlet 1.0 specification defines a standard set of APIs for developing portlets that can run in any Java Portlet 1.0-compliant portal.

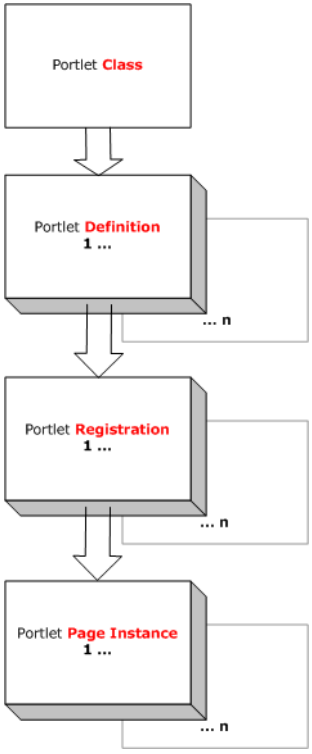
Although this chapter introduces you to basic portlet concepts, you can find detailed information about Java Portlet 1.0 description at the Java Community Process Web site:

```
http://jcp.org/aboutJava/communityprocess/final/jsr168/index.html
```

NOTE: This URL was valid at the time this chapter was published.

Portlet object model

exteNd Director implements a hierarchical portlet object model, illustrated in the following diagram:



Portlet class The Java class that defines the initialization parameters and default behavior of the portlet. Each portlet class in a portlet application must have a definition descriptor to make the portlet available to the portlet container. These descriptors must be created either as an entry in portlet.xml or as a separate portlet fragment deployment descriptor, as described in the section on portlet application deployment descriptors.

Portlet definition An instance of the portlet class. You can create multiple portlet definitions per portlet class—for example, if you want a definition that inherits all the characteristics of a particular class, but uses different initialization parameters.

Portlet registration A registered portlet definition. You can register one or more portlet definitions in a portlet application. A portlet registration inherits preferences and settings from its parent definition, but exteNd Director allows you to override any of these defaults in each registration. In this way, you can place multiple registrations of the same portlet definition on a single portal page, where each registration exhibits unique behavior and generates unique content.

You register portlet definitions in the Portlet Management section of the Director Administration Console (DAC), as described in the section on registering portlet definitions.

Portlet page instance An instance of a portlet registration that is associated with a specific portal page. A portlet page instance inherits preferences and settings from its parent portlet registration. You can override any of these defaults when you assign a portlet registration to a portal page using the Portal Page Administration tool and the Portal Personalizer tool. The following references describe how to assign portlet registrations to different types of portal pages:

To assign portlet registrations to	See
Container pages	Adding content to a container page.
Shared pages	Adding content to a shared page.
Personal pages	Adding content to a personal page

📖 For information about assigning preferences at each level of the portlet object model, see "Portlet preferences" on page 162.

exteNd Director extensions to Java Portlet 1.0

The exteNd Director implementation of Java Portlet 1.0 includes the following extensions, designed to offer more flexibility in developing and running portlets:

- Design tools for developing specialized portlets called *pageflows* without the need for Java programming

  📖 See the chapter about pageflows.

- An additional, optional portlet deployment descriptor—**novell-portlet.xml**—that allows you to specify a broader range of preferences and settings for portlets, such as title bars, automatic registration, and style sheets for portlets that generate XML.

  📖 See the section on portlet application deployment descriptors.

- Ability to set and modify portlet preferences at design time—and after deployment at registration time, page assignment time, and runtime

  📖 See "Portlet preferences" on page 162.

- Ability to style portlets that generate XML content

  📖 See the section on styling portlets that generate XML content.

- Default implementation for Edit mode preference sheet

  📖 See the section on default implementation for Edit mode.

- Ability to specify asynchronous versus synchronous processing at design time for individual portlets or all portlets in an application

  📖 See the section on synchronous versus asynchronous processing.

- Dynamic loading of portlets using portlet fragment deployment descriptors

  📖 See the section on portlet application deployment descriptors.

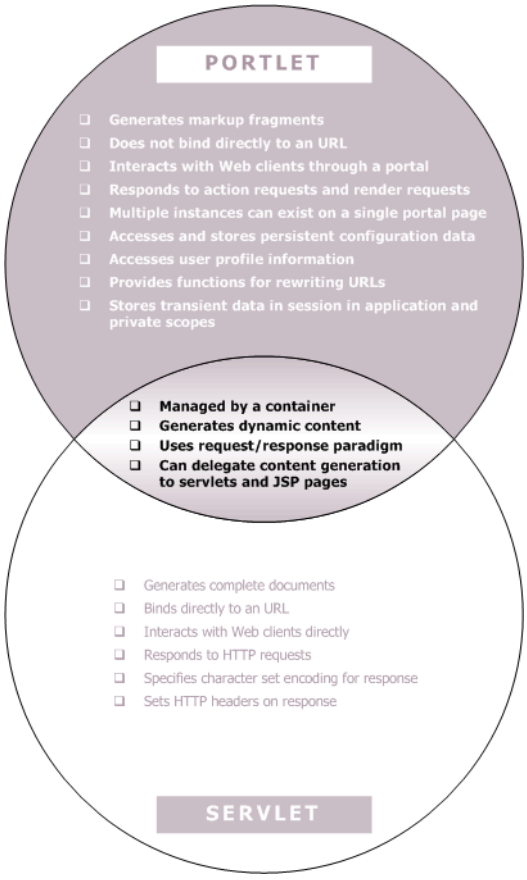
The relationship between portlets and servlets

Although portlets and servlets share many functional characteristics, they are distinct Java classes with individual—though analogous—missions: portlets encapsulate application logic to run in a portal; servlets encapsulate application logic to run on a Web server.

javax.Portlet mimics javax.Servlet and leverages the following functionality from servlets:

- Content generation
- Deployment
- Class loading
- Management of life cycle and session
- Interaction with Web clients
- Request handling and dispatching

The following diagram compares portlets and servlets, highlighting their features in common and those that are specific to each object:



Portlet container

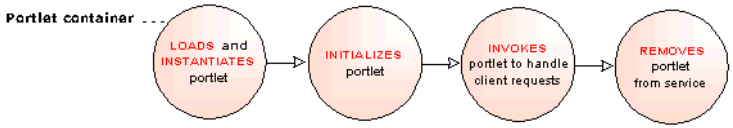
Like servlets, portlets are managed by a container object. Called a *portlet container*, the object acts as a liaison between the portal and its associated portlets.

As specified in Java Portlet 1.0, the portlet container is responsible for:

- Managing the portlet life cycle, as described in "Portlet life cycle" on page 154.
- Establishing the portlet context, as described in "Portlet context" on page 159.
- Providing persistent storage for portlet preferences, as described in "Portlet preferences" on page 162.
- Responding to requests from the portal to execute on specific portlets, as described in "Portlet requests and responses" on page 160.

Portlet life cycle

Portlets move through several stages from initial activation to final deactivation. These stages constitute the portlet life cycle, as shown in the following diagram:



Portlet containers in exteNd Director automatically manage the life cycles of portlets in your deployed portlet applications.

Loading and instantiation

The portlet container loads portlet classes, then instantiates them for use. The portlet container invokes the same class loader to load portlet resources as the servlet container uses to load Web application resources per Web application.

In exteNd Director, portlet containers begin loading portlets and other portlet resources when you deploy a portlet application or start a server that contains deployed portlet applications.

Initialization

During initialization, the portlet may execute performance-intensive, one-time activities, such as connecting to databases or to Enterprise JavaBeans (EJBs).

The portlet container initializes the portlet object based on the following configuration data:

- Parameters set in the portlet deployment descriptor
- Resource bundles defined in the portlet deployment descriptor
- Context object, which describes the portlet's runtime environment

exteNd Director provides several portlet deployment descriptors:

- The standard **portlet.xml**
- An optional extension file **novell-portlet.xml**
- Descriptors for individual portlets in the portlet application resource set, required for dynamic loading of portlets, as described

 For more information, see the section on portlet application deployment descriptors.

Invocation

After the portlet is initialized, the portlet container can invoke it to handle client requests that are triggered by portlet URLs, as described in "Portlet URLs" on page 158. These URLs specify that a particular portlet respond to an action request or render request from a client.

Portlets respond to action requests by changing state and respond to render requests by generating content, as described in "Portlet requests and responses" on page 160. Portlets display content in a portlet window constructed by the portlet container on the portal page, as described in "How portlets are rendered" on page 155.

Removal from service

The portlet container determines when to remove a portlet from service—for example to conserve resources or as part of housekeeping prior to shutting down.

When the portlet container disables a portlet, the portlet releases its resources and saves persistent state information.

How portlets are rendered

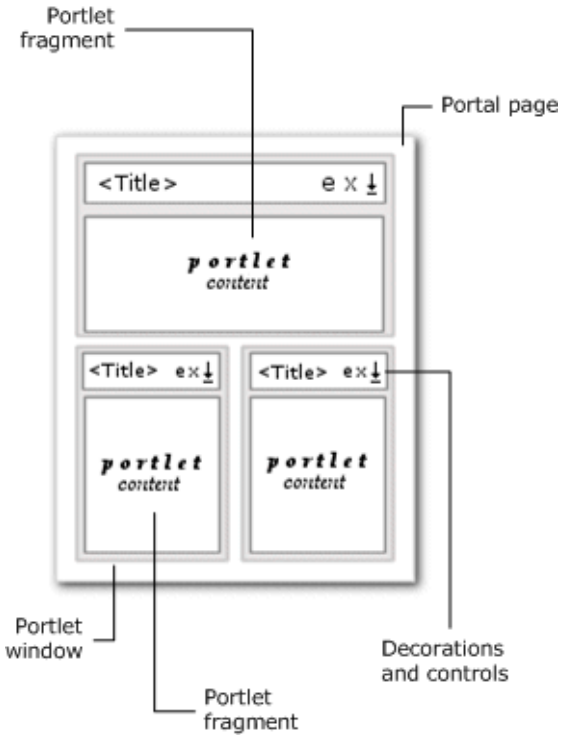
When a portal page is requested at runtime, the exteNd Director Portal Aggregator aggregates the content of all portlets assigned to the page. To render portlets, the Portal Aggregator sends a request to the portlet containers to allocate real estate on the page for displaying the content generated by each portlet. This area is called a portlet window, described in the next section.

Portlet windows

A *portlet window* is an area on a portal page that consists of the content generated by the portlet, along with decorations and controls added by the portal. Each portlet window is constructed from a portlet class that is associated with a preferences object. The preferences object provides access to the values of preferences and settings specified in the portlet deployment descriptors.

For more information, see "Portlet preferences" on page 162 and the section on portlet application deployment descriptors.

The portal aggregates portlet windows into a complete document, called the portal page, as in the following example:



A single portal page may contain multiple portlet windows that reference the same portlet definition, but display different registered instances of the portlet—each with its own preferences and settings specified at page assignment time. For example, you might want to assign multiple registrations of a weather portlet to the same page to display forecasts for different cities.

Window states

A *window state* indicates the amount of portal page real estate to be allocated for the content generated by a portlet. The portlet container sends the current window state to each portlet it invokes; the portlet then determines how much information to render. Portlets can programmatically change their window states when processing action requests.

Supported window states

The exteNd Director portal supports the standard window states defined in Java Portlet 1.0:

Window state	Description
Normal	The portlet may share the portal page with other portlets or components. The portlet assumes it has limited display space and restricts the size of its rendered output.
Maximized	The portlet occupies most or all of the page real estate so it generates richer content than it would in the normal state.
Minimized	The portlet renders minimal output or no output at all. In exteNd Director, the portlet renders the title bar with options, if enabled, but no content.

Portlet content

The content generated by a portlet is called a *fragment*. Each fragment is a piece of *markup*—that is, a sequence of characters and other symbols that describe a document's logical structure or specify how a file should look when it is printed or displayed.

exteNd Director portlets support the following types of markup:

- HTML
- XML
- XHTML
- WML

exteNd Director allows you to style portlets that generate XML, as described in the section on styling portlets that generate XML content.

Portlets generate content in response to render requests, as described in "Portlet requests and responses" on page 160. The window state and mode of a portlet determines how it renders content, as described in "How portlets are rendered" on page 155.

The portal aggregates the content of multiple portlets to form a portal page.

Portlet modes

The mode of a portlet indicates what activities the portlet should perform and what content it should generate.

Standard portlet modes

exteNd Director supports the standard portlet modes specified by Java Portlet 1.0:

Mode	Portlet activity	How to implement
View	Generates markup that reflects its current window state **NOTE:** This is the default mode. All portlets must support View mode.	Override the **doView()** method of the GenericPortlet class
Edit	Provides content and logic to allow users to customize portlet behavior at runtime.	Override the **doEdit()** method of the GenericPortlet class
Help	Provides help information about the portlet	Override the **doHelp()** method of the GenericPortlet class

Portlet URLs

A *portlet URL* is an URL that triggers a request for action by the portlet it references. Portlets generate these URLs as part of their content. When users select a portlet URL—for example by clicking a link that references the URL— the client sends a request to the portal for action by the portlet.

Types of portlet URLs

There are two types of portlet URLs, each triggering different sequences of portlet requests:

Type of portlet URL	Triggers:
Action URL	One action request, followed by one render request per portlet on the portal page
Render URL	One render request per portlet on the portal page

📖 For more information about action and render requests. see "Portlet requests and responses" on page 160.

Information associated with portlet URLs

The portlet API allows you to include the following information in portlet URLs:

- Request parameters
- One portlet mode
- One portlet window state

Portlet context

The portlet context provides information about the container in which each portlet is running. Through the context object, portlets can:

- Record events in the portlet log
- Access URL references to portlet application resources
- Set and store attributes that can be shared by other portlets and servlets in the portlet application

Scope of portlet context

The scope of the portlet context depends on the type of portlet application:

Type of portlet application	Scope of portlet context
Local	One context instance per portlet application per Java Virtual Machine (JVM)
Distributed	One context instance per JVM

Based on servlet context

The portlet context leverages functionality from the servlet context of the portlet application, including:

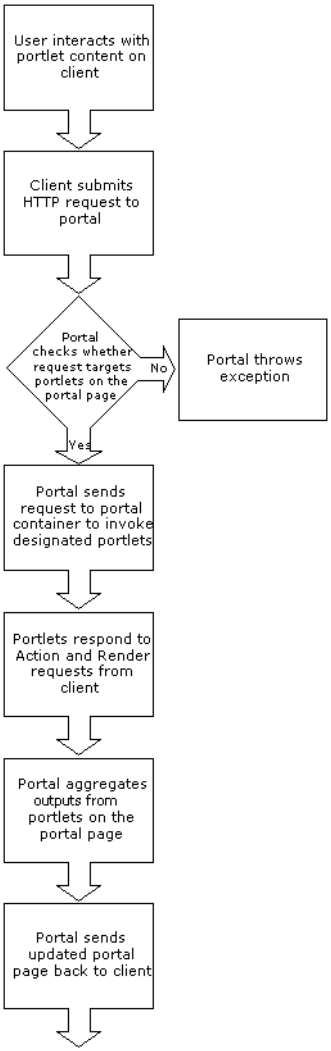
- Initialization parameters
- Context attributes
- Resources exposed by the servlet context

As a result, portlets can share data with servlets and JSP pages in the same Web application. Data stored in the servlet context by servlets and JSP pages can be accessed by portlets through the portlet context. In turn, servlets and JSP pages can access portlet data through the servlet context.

Portlet requests and responses

Web clients communicate with portlets using a request/response paradigm implemented by the portal. The paradigm is based on the Hypertext Transfer Protocol (HTTP) request/response model.

Portlet requests are generated when users interact with content produced by portlets on a portal page—for example by navigating links or submitting forms in a Web browser. Here is a typical request/response scenario:



There are two types of portlet requests—*action requests* and *render requests*. These requests are triggered in different sequences by portlet URLs, as described in "Portlet URLs" on page 158.

The following sections describe each type of portal request and response.

Action requests and responses

An *action request* requires a portlet to update its state, based on a set of input parameters, defined by the developer.

Portlets respond to action requests by performing portlet-specific logic, such as:

- Changing mode (see "Portlet modes" on page 158)
- Changing window state (see "How portlets are rendered" on page 155)
- Redirecting the user to a new URL
- Setting render parameters
- Sending an e-mail
- Updating a database record
- Modifying a preference

Action requests take precedence over render requests in an action URL to ensure that the portlet enters the desired state before rendering content. This order of operations is important because portlet window state and mode play a role in the amount and type of content to render.

For example, suppose a portlet causes an action to occur via a link. When the user clicks the link—for example, to update a record—the portal is guaranteed to process the action before rendering any of the portlets. Thus, the rendered content can reflect the updated data from the action.

When processing an action URL, portlet containers must wait for the portlet to complete the action request before initiating the render requests for the portal page.

Render requests and responses

A *render request* requires a portlet to generate content based on its current render state.

Often, render URLs trigger multiple render requests for a given portal page. Render requests may be executed sequentially or in parallel in random order.


For more information, see the section on synchronous versus asynchronous processing.

Portlets respond to render requests by producing content directly or delegating content generation to a servlet or JSP page.

This flexibility allows you to make informed decisions about how your portlet should provide content to a portal page. For example, if you already have a servlet that generates specialized content for your Web application, you don't need to reinvent the wheel; your portlet can delegate the task of content creation to that servlet.

Content rendering is governed by the portlet's mode and window state. For example, if the portlet's window state is MINIMIZED, no content can be rendered.

📖 For more information, see "How portlets are rendered" on page 155 and "Portlet modes" on page 158.

Portlet sessions

Each portlet application includes one portlet session object per user session. The portlet session object provides a mechanism for all resources in the portlet application to share information.

Recall that portlet applications may contain servlets and JSP pages in addition to portlets. To facilitate data sharing, the portlet session stores all attributes in the HTTP session of the portlet application. As a result, portlets use PortletSession methods to access data stored by servlets or JSP pages in the HttpSession object. Similarly, servlets and JSP pages use HttpSession methods to access data stored by portlets in the PortletSession object.

PortletSession methods are based on the HttpSession methods of the same names.

Portlet preferences

At runtime, portlets are associated with a preferences object whose attributes determine how a portlet behaves and what content it produces. Portlet preferences are defined in the portlet class and, therefore, are unique for every portlet.

exteNd Director provides a flexible paradigm that allows you to apply portlet preferences at four levels of priority, based on the portlet object model:

Priority	Type	Description	Where specified	How specified
1 (highest)	User-level Preferences	Writable preferences defined on a portlet page instance by a user at runtime, Overrides **definition-level**, **registration-level**, and **page-assignment-level** preferences for a given portlet instance on a specific page. At this level, you can set different preferences for the same portlet on different pages or for each instance of a single portlet on the same page.	User preference data store	Edit portlet preferences at runtime—for example in the Edit mode of a portlet.
2	Page-Assignment-level preferences	Writable preferences defined for a portlet registration when it is assigned to a portal page, usually by an administrator after deployment and registration. Overrides **definition-level** and **registration-level** preferences for a given portlet registration on a specific page. At this level, you can set different preferences for the same portlet on different pages or for each registration of a single portlet on the same page.	Portal Administration tool	Assign portlets to shared and container pages using the Portal Administration tool. 📖 See the chapter on administering the portal.

Priority	Type	Description	Where specified	How specified
3	Registration-level preferences	Writable preferences defined for a portlet instance when it is registered, usually by an administrator after deployment. Overrides **Definition-level** preferences for a given portlet registration. **NOTE:** Registered portlet instances are called *portlet registrations*.	Director Administration Console (DAC)	Register portlet instances using the Director Administration Console (DAC). 📖 See the chapter on using the Portlet Management section of the DAC.
4 (lowest)	Definition-level preferences	Default, read-only preferences defined for a portlet definition. These preferences cannot be modified after deployment.	Portlet deployment descriptor	Edit portlet deployment descriptors. 📖 See the section on portlet deployment descriptors.

The portlet container determines what preferences and values are presented to the user. At runtime, the portlet container searches from highest to lowest level for preference values, stopping at the level where it finds the first occurrence. For example, if the portlet container finds the target in page assignment preferences, it does not continue searching in registration or descriptor preferences.

Portlets are associated with a preferences object during these activities:

- When a portlet handles a request
- When a portlet is placed in a portal page (to create a portlet window)

The portlet may read, modify and add preference attributes.

📖 To learn how to get and store preferences at each level programmatically, see the section on getting and storing portlet preferences.

Portlet settings

Portlet settings govern how portlets should interact with the portal. Settings are defined by the portal and can be applied to portlet registrations and portlet page instances in the application.

📖 For more information about portlet registrations and portlet page instances, see "Portlet object model" on page 150.

Standard settings exteNd Director defines the following portlet settings:

- Title
- Maximum timeout
- Requires authentication
- Display title bar
- Hidden from user

These settings are defined by a portal administrator in the exteNd Director portlet deployment descriptor, **novell-portlet.xml**. They can also be modified by the administrator at the registration and page assignment levels.

Portal options exteNd Director defines the following optional portal settings, also known as portal options:

- Restart pageflow
- Help
- Edit
- Minimize
- Restore
- Maximize
- Remove

Each portlet specifies which of these options it supports (if any) in the <supported-options> element, defined in novell-portlet.xml. For more information see the chapter on working with portal options.

12 About Portlet Applications

This chapter provides an introduction to exteNd Director portlet applications. It contains the following sections:

- What is a portlet application?
- Support for dynamic loading of portlets
- How portlet applications work with the exteNd Director portal
- Configuring portlet applications
- Portlet application deployment descriptors


For more information about exteNd Director projects in general, see the chapters that cover working with projects in *Developing exteNd Director Applications* .

What is a portlet application?

Java Portlet 1.0 defines a portlet application as a Web application that consists at a minimum of one or more portlets, the Web application deployment descriptor (web.xml), and a portlet deployment descriptor (portlet.xml). exteNd Director provides additional portlet deployment descriptors that allow you to:

- Configure a broader range of preferences and settings
- Dynamically load portlets from a resource set


For more information, see “Portlet application deployment descriptors” on page 174.

How portlet applications interact with portal applications

A portlet application must run in conjunction with a portal application. The portal application includes a portal service that interacts with portlet applications in this way:

1. The portal service sends portlet requests to the portlet applications where the portlets reside.
2. The portlets respond by processing actions and generating dynamic content
3. The portal service renders the content generated by portlets on portal pages

NOTE: exteNd Director supports several types of portal pages: **personal**, **shared**, and **container** pages. For more information, see the chapter on working with portal pages.

Portlet applications can run externally or locally in relation to the portal application. exteNd Director supports each scenario, as described in "How portlet applications work with the exteNd Director portal" on page 169.

Packaging requirements

Portlet applications are packaged as Web application archives (WARs). When portlet applications need additional resources that cannot be packaged in the WAR file— such as Enterprise JavaBeans (EJBs)—the portlet application may be packaged together with these resources in an EAR file.

Directory structure

Portlet applications use the same directory structure as J2EE Web applications, but require an additional deployment descriptor in WEB-INF, as described in "Portlet application deployment descriptors" on page 174.

Portlet classes and resources must reside in WEB-INF/classes, within a JAR in WEB-INF/lib, or as defined by the resource set class loader (specified in the **<libPath>** element in resourceset.xml of the portlet application).

In exteNd Director, you can add portlet classes and deployment descriptors locally to the resource set of an exteNd Director portal application or portlet application to enable dynamic loading of updates during development and test cycles.

 For more information, see "Support for dynamic loading of portlets" on page 169.

Support for dynamic loading of portlets

exteNd Director supports dynamic loading of portlets if they are stored in the resource set of an exteNd Director portal application. When you enable this feature, exteNd Director dynamically loads portlet changes from disk rather than from the deployed WAR, and reflects the changes at runtime. As a result, dynamic loading speeds development, because you can test modifications without having to redeploy the entire project.

Dynamic loading is enabled by default in the Express Portal project, and when you create a new portal application EAR or WAR using the exteNd Director Project Wizard.

For portlets, dynamic loading is implemented seamlessly when you create pageflows (a type of portlet) with exteNd Director pageflow design tools and when you develop portlets with the exteNd Director Portlet Wizard. These tools automatically create portlet fragment deployment descriptors and store them along with the associated portlet classes at the appropriate locations in the resource set of a portal application.

 For more information about resource sets, see the chapter on using the resource set in an exteNd Director application. For more information about dynamically loading portlet descriptors, see "exteNd Director portlet fragment deployment descriptor" on page 182.

How portlet applications work with the exteNd Director portal

Portlet application WARs can run externally or locally in relation to the exteNd Director portal on an application server.

Running portlet applications externally with the exteNd Director portal

A portlet application is considered to run externally in relation to the portal application when:

- The portlet application is packaged as a standalone application WAR and deployed to the server where the portal application WAR is running in a shared library environment.

  IMPORTANT: In a shared library environment, only one exteNd Director portal application can be deployed to the server at a time, and no other portal applications can be deployed to the server at the same time.

- The portlet application WAR is packaged with a portal application WAR inside an EAR

 For more information see the section on changing a project's shared library configuration.

Running portlet applications locally with the exteNd Director portal

A portlet application is considered to run locally to the portal when its portlets and portlet deployment descriptors reside inside the portal WAR—whether the portal WAR is packaged in an EAR or as its own self-contained application.

In exteNd Director this scenario yields a slight performance advantage because the portal communicates directly with the portlet container within the same context.

Configuring portlet applications

When configuring portlet applications, consider these key decision points:

Decision	Implication
Do I want to create custom portlets and run them locally in an exteNd Director portal application?	You use exteNd Director design tools to create portlets and package them locally in the exteNd Director portal application: ◆ Pageflow design tools allow you to create a specialized type of portlet called a pageflow without Java coding, as described in the *Pageflow and Form Guide* ◆ Portlet Wizard allows you to create Java Portlet 1.0-compliant portlets that take advantage of exteNd Director extensions to Java Portlet 1.0. When you use these tools, you will be able to dynamically load portlet classes and portlet deployment descriptors during the development cycle. Dynamic loading allows you to test modifications to portlets without having to redeploy an entire application. 📖 For details, see "Configuring portlet applications to run locally in an exteNd Director portal application" on page 173.

Decision	Implication
Do I want to run existing Java Portlet 1.0-compliant portlet WARs from other sources with an exteNd Director portal application?	You can run existing Java Portlet 1.0-compliant portlet applications externally in relation to an exteNd Director portal application. You use the exteNd Director Portlet Application Wizard to configure the existing portlet WAR to run externally to the exteNd Director portal application. 📖 For details, see "Configuring portlet applications to run externally with the exteNd Director portal" on page 171.

Configuring portlet applications to run externally with the exteNd Director portal

➢ **To configure portlet applications to run externally when deployed as standalone WARs:**

1. Configure your application server to use a full shared library configuration, as described in procedures for changing the project configuration.
2. Create an exteNd Director portal application EAR or WAR project, as described in creating an exteNd Director project.
3. Configure the exteNd Director portal application project to use shared libraries, as described in procedures for changing the project configuration.
4. Deploy the exteNd Director portal application project to the application server.
5. Configure one or more existing portlet applications to work with the exteNd Director portal, as described configuring a portlet application project.
6. If you want to package the portlet applications inside an EAR, follow the instructions for adding projects.

   NOTE: Java Portlet 1.0-compatible portlet applications can also be deployed as standalone WARs.
7. Deploy the portlet application WARs or EAR to the application server.

➢ **To configure portlet applications to run externally when deployed within a portal EAR:**

1. Create an exteNd Director portal application EAR project, as described in creating an exteNd Director project.
2. With the portal application EAR project open in the exteNd Director development environment, select **File>New>Project**.

   The New Project dialog displays.

3 Select **Portlet Application** and click **OK**.

The Add to Current Project dialog displays, asking if you want to add the portlet application to the currently open EAR project.

4 Select **Yes** and click **Next**.

The Select Existing Web App WAR dialog displays.

5 Enter the path to your portlet application WAR, as in this example:

6 Click **Finish**.

The portlet application WAR is configured to work with the exteNd Director portal and is added to your EAR project.

7 Deploy the portal application EAR to your application server.

Configuring portlet applications to run locally in an exteNd Director portal application

➢ **To run portlet applications locally in an exteNd Director portal application:**

1 Create an exteNd Director WAR or EAR project using the exteNd Director Project Wizard, as described in creating an exteNd Director project.

NOTE: The wizard creates a project that includes a portal, portlet runtime artifacts, and resource set. The portlet runtime artifacts include the portlet deployment descriptors **portlet.xml** and **novell-portlet.xml**, as described in "Portlet application deployment descriptors" on page 174.

2 With the exteNd Director portal application WAR or EAR open, create Java Portlet 1.0-compliant portlets using either of the following exteNd Director design tools:

- Pageflow Modeler
- Portlet Wizard.

NOTE: Each of these tools automatically adds the new portlets to the resource set of the portal application, along with a portlet fragment deployment descriptor for each portlet. Each fragment descriptor stores both Java Portlet 1.0 and exteNd Director configuration information for its associated portlet; there is no need to update portlet.xml and novell-portlet.xml.

3 Rebuild and redeploy the project.

Portlet application deployment descriptors

exteNd Director provides support for the following deployment descriptors for portlet applications:

Deployment descriptor	Name of descriptor file	Comments
Web application deployment descriptor	web.xml	Required for all WARs See "Web application deployment descriptor" on page 176.
Standard portlet deployment descriptor, defined by Java Portlet 1.0	portlet.xml	Required for all portlet application WARs See "Standard portlet deployment descriptor" on page 177.
exteNd Director extension to standard portlet deployment descriptor	novell-portlet.xml	Optional • Allows you to add exteNd Director value-added extensions to standard preferences • Allows you to define an extended set of preferences and settings • Used by portlet applications that run with the exteNd Director portal • Extends portlet.xml See "exteNd Director portlet deployment descriptor" on page 177.

Deployment descriptor	Name of descriptor file	Comments
exteNd Director portlet fragment deployment descriptor	*portlet name*.xml	Optional ◆ Provides the union of all specifications in portlet.xml and novell-portlet.xml ◆ Allows you to dynamically load portlets during development ◆ Can be used only in portlet applications that run with the exteNd Director portal ◆ Requires that the portlet application have a resource set **NOTE:** The Portlet Application Wizard automatically adds a resource set to an existing portlet application ◆ Cannot be used in portlet applications that run with third-party portals See "exteNd Director portlet fragment deployment descriptor" on page 182.

NOTE: When you use the exteNd Director Project Wizard to create an exteNd Director WAR or EAR project, portlet.xml and novell-portlet.xml are automatically included in your project. See the chapter on developing portlets.

Determining which deployment descriptors to use

The following chart presents guidelines for determining which portlet deployment descriptors to include in your portlet applications.

Deployment descriptor	Include in all portlet applications	Include to dynamically load portlets	Include to specify value-added exteNd Director preferences	Include when creating portlets with Portlet Wizard	Works with exteNd Director portal only	Requires an exteNd Director resource set
web.xml	●					
portlet.xml	●					
novell-portlet.xml			●		●	
portlet name.xml		●	●	●	●	●

Web application deployment descriptor

Recall that the Web application deployment descriptor **web.xml** contains the standard settings for a WAR and resides in the **/WEB-INF** directory of your application WAR. This descriptor specifies all standard Web resources including servlets, JSP pages, HTML pages, Java classes, and static documents.

Because web.xml is not extensible, portlet resources must be specified in a separate file, described in "Standard portlet deployment descriptor" on page 177. However, the following portlet application properties can be set in the web.xml deployment descriptor:

Portlet application property to specify in web.xml	Tag
Portlet application description	<description>
Portlet application name	<display>
Portlet application security role mapping	<security-role>

Standard portlet deployment descriptor

The standard portlet deployment descriptor **portlet.xml** must be included in the **/WEB-INF** directory of every portlet application. It specifies configuration and deployment information for all portlets in the application.

📖 For more information about the standard portlet deployment descriptor, see the Java Portlet 1.0 specification. You can also view the schema in:

```
extend5 install directory\Common\Resources\SchemaCatalog\portlet-app_1_0.xsd
```

exteNd Director portlet deployment descriptor

When you run portlet applications with the exteNd Director portal, you have the option of including the exteNd Director portlet deployment descriptor **novell-portlet.xml** in your portlet applications. This optional descriptor extends portlet.xml by providing settings and additional preferences that are interpreted by the exteNd Director portal, as described in "A look inside novell-portlet.xml" on page 178.

novell-portlet.xml contains the following types of information:

- Additional descriptor elements for standard preferences defined in portlet.xml
- exteNd Director value-added preferences
- exteNd Director settings

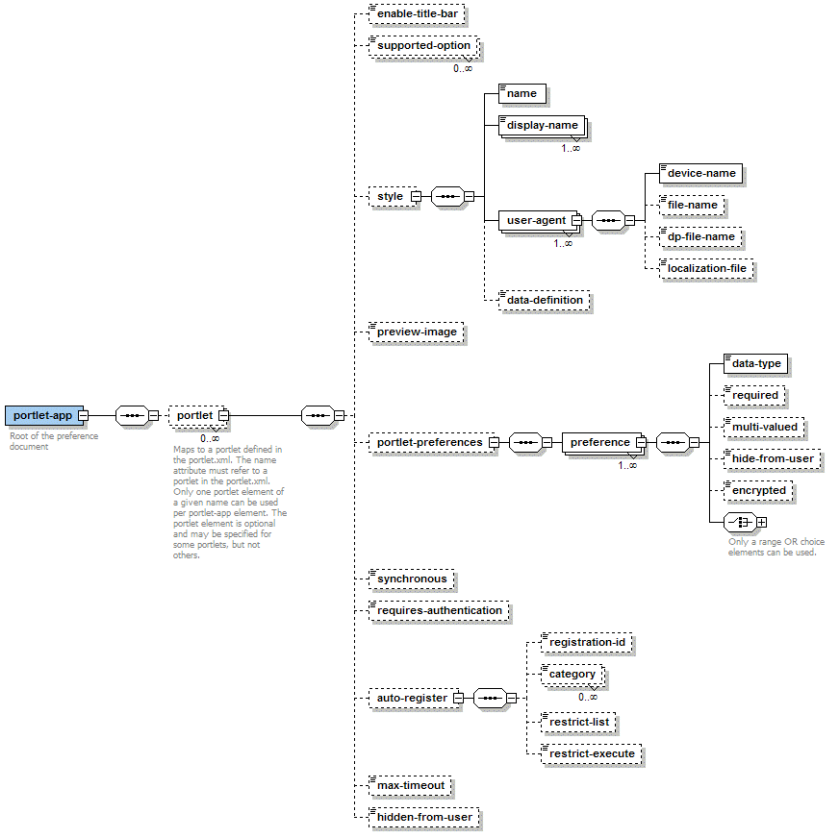
  NOTE: Although settings can be specified in novell-portlet.xml at design time, they can also be modified by an administrator after deployment when portlet definitions are registered and when portlet registrations are assigned to pages, as described in the section on portlet settings.

You include novell-portlet.xml in the **/WEB-INF** directory of your portlet application, along with portlet.xml. and web.xml.

Whereas portlet.xml requires that you include descriptors for all portlet definitions in your application, novell-portlet.xml has no such restriction. You include only the portlet definitions that require additional settings, but you cannot include portlets that haven't been defined in portlet.xml.

A look inside novell-portlet.xml

Here is a high-level view of the novell-portlet.xml schema:



You can view the novell-portlet.xml schema in:

extend5 install directory\Common\Resources\SchemaCatalog\novell-portlet.xsd

The following table describes the elements you can define in novell-portlet.xml for any portlets in your application

Element	Description	Example

enable-title-bar	A setting that specifies whether to display the portlet with or without a title bar:
	◆ 1 = enable
	◆ 0 = disable
	NOTE: This setting must be enabled if you want the options defined in **supported-options** to be displayed in the portlet window
		`<portlet name="TextMessagePortlet">`
		`...`
		`<enable-title-bar>1</enable-title-bar>`
		`...`

supported-option	A setting that specifies which of the options defined in the portal are supported by the portlet.
	All options defined by the portal are described in the portal-option directory of the portal application resource set.
	NOTE: If you want options to be displayed in the portlet window, you must set **enable-title-bar**
	For more information, see the chapter on working with portal options.
		`<portlet name="TextMessagePortlet">`
		`...`
		`<supported-option>edit</supported-option>`
		`<supported-option>help</supported-option>`
		`...`

style	A configuration option that specifies a style for the portlet. Only one style element can be specified per portlet, but each style can contain multiple user agents that point to device-specific XSL style sheets.	`<portlet name="BookmarkPortlet">`
		`...`
		`<style>`
		`<name>BookmarkPortletDefault</name>`
		`<display-name>Default Bookmark Portlet Style</display-name>`
		`<user-agent>`
		`<devicename>Generic_HTML</device-name>`
		`<file-name>$RESOURCE_SET$/portal-style/BookmarkPortlet.xsl</file-name>`
		`</user-agent>`
		`</style>`
		`...`

Element	Description	Example
preview-image	A configuration option that provides an image for previewing the portlet.	`<preview-image>$RESOURCE_URL$/images/PortletPreview.gif</preview-image>`

portlet-preferences	A preference that allows you to extend the configuration choices for standard preferences defined in portlet.xml	`<portlet name="TextMessagePortlet">`
		`...`
		`<portlet-preferences>`
		`<preference name="min-timeout">`
		`<data-type>Integer</data-type>`
		`<required>true</required>`
		`<multi-valued>false`
		`</multi-valued>`
		`</preference>`
		`<preference name="max-timeout">`
		`<data-type>Integer</data-type>`
		`<required>true</required>`
		`<multi-valued>false</multi-valued>`
		`</preference>`
		`<preference name="scrollbar">`
		`<data-type>Boolean</data-type>`
		`</preference>`
		`</portlet-preferences>`
		`...`

synchronous	A preference that specifies whether the portlet executes synchronously or asynchronously
	◆ 0 = asynchronous
	◆ 1 = synchronous
	Asynchronous (multithreaded) execution is the default.
	📖 For more information see the section on synchronous versus asynchronous processing
		`<portlet name="TextMessagePortlet">`
		`...`
		`<synchronous>1</synchronous>`
		`...`

requires-authentication	A setting that specifies whether or not an authenticated user is required to run the portlet
	◆ 0 = does not require authentication
	◆ 1 = requires authentication
		`<portlet name="TextMessagePortlet">`
		`...`
		`<requires-authentication>0`
		`</requires-authentication>`
		`...`

Element	Description	Example

auto-register	A preference that indicates whether or not the portlet should be registered automatically at deployment
	◆ enabled = “false”: do not register automatically
	◆ enabled = “true”: register automatically
	You may want to enable auto-register for:
	◆ System portlets and other portlet registrations that are permanent parts of your application
	◆ Portlets that need to be available to users immediately, without administrator intervention
	You may want to disable auto-register when you don’t know how a portlet will be used at runtime—for example, if the portlet will communicate with a back-end mail server that hasn’t been configured yet
	NOTE: If you remove a portlet that has been automatically registered, the portal will reregister the portlet at server restart
		`<portlet name=”TextMessagePortlet”>`
		`...`
		`<auto-register enabled="true">`
		`<registration-id>`
		`AdminTextMessagePortlet`
		`</registration-id>`
		`<category>General Portlets`
		`</category>`
		`<restrict-list>true`
		`</restrict-list>`
		`<restrict-execute>true`
		`</restrict-execute>`
		`</auto-register>`
		`...`
		NOTE: In this example, an instance of TextMessagePortlet called AdminTextMessagePortlet will be registered automatically at deployment time and added to the General Portlets category. List and execute permissions will be restricted to portal administrators.

max-timeout	A setting that specifies the maximum number of milliseconds the portal should wait for this portlet to return its content. The portlet container can use this value or ignore it.
	◆ >= 0: no timeout set; the request timeout takes precedence
	◆ >0: number of milliseconds to wait before timing out
		`<portlet name=”TextMessagePortlet”>`
		`...`
		`<max-timeout>-1</max-timeout>`
		`...`

Element	Description	Example

hidden-from-user	A setting that indicates whether or not the portlet should be displayed in a selection list to authorized users who are not portal administrators
	◆ true = hide from all users except portal administrators
	◆ false = show portlet to all authorized users
		`<portlet name="TextMessagePortlet">`
		`...`
		`<hidden-from-user>true`
		`</hidden-from-user>`
		`...`

exteNd Director portlet fragment deployment descriptor

The exteNd Director portlet fragment deployment descriptor enables dynamic loading of portlets during the development cycle. Dynamic loading allows you to update portlets and test your changes without redeploying your entire application, as described in "Support for dynamic loading of portlets" on page 169.

IMPORTANT: You can use the portlet fragment deployment descriptor only if your portlet application contains a resource set and runs with the exteNd Director portal.

The portlet fragment deployment descriptor is an XML descriptor that is stored in the **portal-portlet** directory of the resource set in your portlet application. Each descriptor describes the preferences and settings for a single portlet.You must include one portlet fragment deployment descriptor for each portlet you want to dynamically load. The name of the portlet fragment descriptor file should match the name of the portlet class it describes, following this convention:

name of portlet class.xml

For example, the portlet fragment deployment descriptor for the portlet class **TextMessagePortlet** should be called**TextMessagePortlet.xml**.

Contents of portlet fragment deployment descriptor

A portlet fragment deployment descriptor represents the union of portlet.xml and novell-portlet.xml for a particular portlet. It consolidates in one location the following configuration information about its associated portlet:

- Java Portlet 1.0 settings and preferences (normally stored in portlet.xml)
- Value-added exteNd Director settings and preferences (normally stored in novell-portlet.xml)

For a detailed look at the portlet fragment deployment descriptor schema, see:

extend5 install directory\Common\Resources\SchemaCatalog\portlet-fragment.xsd

Automatic generation of portlet fragment deployment descriptors

The exteNd Director Pageflow Modeler and Portlet Wizard automatically generate a portlet fragment deployment descriptor for each pageflow or portlet you create. The descriptor file is automatically stored it in the appropriate location in the resource set to enable dynamic loading.

 For more information, see the chapter on developing custom portlets.

13 Strategies for Developing Portlets

This chapter describes strategies for developing portlets, with an emphasis on using the exteNd Director portlet tools and API for implementing value-added features. The following topics are covered:

- The portlet development cycle
- Portlet development tools
- Anatomy of a portlet class
- The Portlet interface
- The GenericPortlet class
- Working with context objects
- Setting content type
- Synchronous versus asynchronous processing
- Getting information about portlets
- Styling portlets that generate XML content
- Default implementation for Edit mode
- Specifying a secure port for portlet URLs
- Getting and setting cookies on a portlet

 For detailed information about the Java Portlet 1.0 API, see the Java Community Process Web site:

```
http://jcp.org/aboutJava/communityprocess/final/jsr168/index.html
```

The portlet development cycle

When developing a portlet to run with the exteNd Director portal, the following steps are recommended:

1. Create the portlet class using exteNd Director development tools, as described in "Portlet development tools" on page 186.
2. Create at least one instance of the portlet class—called a portlet definition, as described in the section on creating a portlet definition.
3. Register at least one of your portlet definitions to create a portlet registration, as described in the section on registering a portlet definition.
4. Test the portlet by running the portlet registration directly in a browser, as described in the section on testing a portlet.
5. Add the portlet registration to the appropriate portal pages, as described in the section on adding portlets to portal pages.

 For more information about portlet classes and portlet definitions, see the section on the portlet object model.

Portlet development tools

exteNd Director provides several tools for developing portlets:

- **Pageflow design tools** allow you to create a specialized type of portlet called a *pageflow* without the need for Java coding. See "Pageflow design tools" on page 186.
- **Portlet Wizard** simplifies the process of creating custom portlet classes using traditional Java coding techniques

Pageflow design tools

exteNd Director provides design tools for developing pageflows. As the name suggests, *pageflows* are portlets that implement a flow of control between activities within a portlet session. These activities can model a sequence of user interactions or background processing tasks. The design tools in exteNd Director facilitate the process of defining and linking activities in a pageflow without requiring you to write Java code.

Types of pageflow design tools

There are several pageflow design tools to choose from:

Tool	What it does
Pageflow Modeler	Allows you to create generic pageflow processes
Database Pageflow Wizard	Allows you to create *Database pageflows*, which are specialized flows that allow users to find, display, and modify records in a database during their portlet session
Web Service Pageflow Wizard	Allows you to create pageflows that execute Web Services
Composer Pageflow Wizard	Allows you to create pageflows that execute exteNd Composer services

 For an in-depth discussion about pageflows and how to use these design tools, see the *Pageflow and Form Guide*.

When to use pageflow design tools

Consider using pageflow design tools for developing your custom logic when:

- You can model your application requirements as a flow of control between activities within a single portlet session
- Your application does not need to meet cross-platform requirements
- You want to implement your logic without Java coding

Portlet Wizard

Although you can implement most of your portlet application logic using pageflows, there are situations in which you may want to write custom portlet classes. The Portlet Wizard in exteNd Director simplifies the process by

- Providing an interface for specifying portlet settings at design time
- Automatically generating a barebones portlet class that includes the imports, method signatures, and required logic to get started

 To learn how to create custom portlet classes using the Portlet Wizard, see the section on using the Portlet Wizard.

When to use the Portlet Wizard

Consider using the Portlet Wizard for developing your custom logic when:

- You cannot model your application requirements as a flow of control between activities within a single portlet session
- Your applications must meet cross-platform requirements
- Your portlets need to be lightweight, as in real-time applications
- Your are experienced in Java coding
- You want more direct control of code-level logic

Anatomy of a portlet class

This section discusses the logic requirements for a portlet class and describes what the Portlet Wizard generates automatically.

Minimum code requirements

At a minimum, a portlet class must include the following logic:

Required logic	What the Portlet Wizard generates
Include Java imports	Imports: ◆ `java.io.PrintStream` ◆ `java.io.PrintWriter` ◆ `javax.portlet.*`
Implement javax.portlet.Portlet or extend javax.portlet.GenericPortlet	Extends javax.portlet.GenericPortlet
Get initialization parameters from the portlet.xml deployment descriptor	Adds method signature for the init() method
Implement mandatory View (when extending GenericPortlet)	Adds method signature for the doView() method, along with code that: ◆ Sets the content type for the portlet ◆ Gets the Writer of the RenderResponse object ◆ Creates a string buffer to build portlet content
Process portlet requests	Adds method signature and basic code for the processAction() method

Required imports for working with the exteNd Director portal

The Portlet Wizard automatically generates code for importing packages that are required for working with the exteNd Director portal:

- Framework API package `com.sssw.fw.api.*`
- Portal API package `com.sssw.portal.api.*`
- Portlet API package `com.novell.afw.portlet.api.EbiPortletConstants`

Code example

Here is an example of a portlet class generated by the Portlet Wizard, based on its default settings:

```
/**
 * Generated by Novell XSLT Code Generator, version 1.0.
 * This generated source file may be freely modified.
 */


package com.novell.portlets;

// Java imports
import java.io.PrintStream;
import java.io.PrintWriter;
import javax.portlet.*;

// Portal/Framework imports
import com.sssw.fw.api.*;
import com.sssw.portal.api.*;

// Portlet API imports
import com.novell.afw.portlet.api.EbiPortletConstants;


/**
 * MyPortlet
 */
public class MyPortlet extends GenericPortlet {

    // an Instance of a log for error/trace reporting
    private static EbiLog m_log =
com.sssw.fw.log.EboLogFactory.getLog(com.sssw.fw.log.EboLogFactory.PORTLET);

    /**
     * Get the initialization parameters from the portlet.xml file
     */
    public void init() throws PortletException {
    }

    /**
```

```
     * Helper method to serve up the mandatory view mode
     * @param request           an portlet request object
     * @param response          an render response object
     */
    public void doView( RenderRequest request, RenderResponse response ) throws
PortletException, java.io.IOException {

        try {

            // Uncomment the code below when access to portal subsystems is needed
            /*
            // EbiContext stores information about the user's environment
            EbiContext ebiContext = com.sssw.fw.factory.EboFactory.createEbiContext(
request, response, getPortletContext() );

            // Get a reference to the Portal Context object
           EbiPortalContext ebiPortalContext = ( EbiPortalContext ) request.getAttribute(
EbiPortletConstants.EBI_PORTAL_CONTEXT );
            */

            PortletURL renderUrl = response.createRenderURL();
            response.setContentType( EbiPortletConstants.MIME_TYPE_HTML );
            renderUrl.setPortletMode( PortletMode.VIEW );

PrintWriter writer = response.getWriter();

            // Build the screen of HTML, set it as the content
            StringBuffer sb = new StringBuffer();

            // Output code goes here
            sb.append( "View Mode" );
            sb.append( "<br></br>" );

            writer.print( sb.toString() );
        }
        catch ( Throwable e ) {
            // Log any errors generated
            m_log.error(e);
            new PortletException( e );
        }
    }

    /**
     * Process any requests that the portlet may have.
     * @param request           an action request object
     * @param actionResponse    an action response object
     */
    public void processAction (ActionRequest request, ActionResponse response) throws
PortletException, java.io.IOException {

        try {
            PortletContext portletContext = getPortletContext();

            // only log items if the log level indicates we should
```

```
            if ( m_log.isTrace() ) {
                m_log.trace( "MyPortlet in processAction method" );
            }
            // Do "save" processing here if necessary, and then
            // set the portlet mode to be "View" after completion.
            response.setPortletMode( PortletMode.VIEW );
        }
        catch ( Throwable e ) {
            // Log any errors generated
            m_log.error( e );
            new PortletException( e );
        }
    }
}
```

The Portlet interface

All portlets implement the Portlet interface, as defined by Java Portlet 1.0. The Portlet interface provides methods that the portlet container uses to manage the life cycle of each portlet:

Method	Description
init()	Places a portlet into service
processAction()	Notifies the portlet to respond to an action request sent by a client and triggered by an action URL. This method can be called only if the portlet has created an action URL with RenderResponse.createActionURL
render()	Notifies the portlet to respond to a render request sent by a client by generating content. The portlet renders content based on its current state and the request/response pair passed in as parameters.
destroy()	Takes a portlet out of service

Portlets can implement the Portlet interface directly or extend the **GenericPortlet** class, which implements the Portlet interface and also provides helper methods, as described in "The GenericPortlet class" on page 192.

The GenericPortlet class

When you develop portlets, you can extend the GenericPortlet class, rather than implement the Portlet interface directly.

The **GenericPortlet** class provides a default implementation for the **Portlet** interface. The advantage of extending GenericPortlet is that you gain access to its helper methods which simplify coding for the following tasks:

- Implementing standard portlet modes
- Handling life cycle phases
- Getting portlet configuration parameters

Implementing standard portlet modes

The following methods defined in the GenericPortlet interface implement the standard portlet modes, as described in "Portlet modes" on page 158. When you write portlets, you can override these methods with custom processing logic as needed for each mode you implement:

Method	Description
doView()	Implements portlet behavior in View mode. This is the default mode in which the portlet generates markup that reflects its current window state.
doHelp()	Implements portlet behavior in Help mode. This is an optional mode that provides help information about the portlet.
doEdit()	Implements portlet behavior in Edit mode. This is an optional mode that provides content and logic to allow users to customize portlet behavior.

These methods perform the render operation in each mode.

NOTE: exteNd Director provides a default implementation for Edit mode, as described in "Default implementation for Edit mode" on page 203.

Portlet life cycle methods

The following methods defined in the GenericPortlet interface implement portlet life cycle phases, as described in "Portlet life cycle" on page 154:

Method	Description
init()	Called by the portlet container to place the portlet into service
processAction()	Notifies the portlet that an action request was issued
render()	Notifies the portlet that a render request was issued
destroy()	Called by the portal container to take the portlet out of service

Getting portlet configuration parameters

The following **get** methods defined in the GenericPortlet interface provide access to portlet configuration parameter:

Method	Gets
getInitParameter()	Value of a named portlet initialization parameter
getInitParameterNames()	Names of all portlet initialization parameters
getPortletConfig()	Configuration object for the portlet
getPortletContext()	Portlet application context, as described in "Portlet context" on page 159
getPortletName()	Name of the portlet
getResourceBundle()	Resource bundle for a given locale

Working with context objects

In addition to supporting the standard context objects defined in Java Portlet 1.0, exteNd Director provides proprietary context objects that allow portlets to interact with the exteNd Director portal and communicate with other exteNd Director subsystems:

- EbiContext
- EbiPortalContext

EbiContext

EbiContext, defined in the exteNd Director Framework system, is the interface through which all exteNd Director subsystems communicate. EbiContext stores information about the user environment, including the user's ID, session, and the response and request objects appropriate to the current user agent or browser.

You can also access information about other exteNd Director subsystems through the EbiContext, as described in "Using EbiContext to access information about other subsystems" on page 195.

The EbiContext object exists for the duration of a request. With each new request, the exteNd Director portal instantiates a new context object. Information that persists between requests is stored in the EbiSession object, available by calling a getEbiSession() method on EbiContext.

By contrast, there is one standard javax.portlet.PortletContext per portlet application. The standard PortletContext object does not provide access to exteNd Director-specific information in the portlet application.

Getting a reference to EbiContext

Portlets can reference EbiContext objects by calling createEbiContext() methods on the EboFactory class for the Framework subsystem. To get an EbiContext context object, use:

```
EbiContext context =
com.sssw.fw.factory.EboFactory.createEbiContext(req, res,
getPortletContext());
```

Getting EbiContext for specific types of portlet requests and responses

You can get an EbiContext associated with a particular type of portlet request and response, as follows:

- To get the action context object, use:

  ```
  com.sssw.fw.factory.EboFactory.createEbiContext(javax.portlet.A
  ctionRequest request, javax.portlet.ActionResponse response,
  javax.portlet.PortletContext ctxt)
  ```

  This method returns an EbiActionContext object when you call it from the processAction() method of your portlet. EbiActionContext is an extension of EbiContext. The difference between an EbiContext object and an EbiActionContext object is that EbiActionContext is guaranteed to have an ActionRequest and an ActionResponse as the underlying wrapped objects.

- To get the render context object, use:

  ```
  com.sssw.fw.factory.EboFactory.createEbiContext(javax.portlet.R
  enderRequest request, javax.portlet.RenderResponse response,
  javax.portlet.PortletContext ctxt)
  ```

This method returns an EbiRenderContext object when you call it from your portlet's render(), doView(), doEdit(), or doHelp() method. EbiRenderContext is an extension of EbiContext. The difference between an EbiContext object and an EbiRenderContext object is that EbiRenderContext is guaranteed to have a RenderRequest and a RenderResponse as the underlying wrapped objects.

Using EbiContext to access information about other subsystems

There are several ways for a portlet to access information about other exteNd Director subsystems through EbiContext:

- Use EbiContext to access the context objects of other subsystems.

  For example, the following code accesses the EbiContext of the Rule subsystem from the EbiContext of the Framework subsystem:

```
...
//Get the EbiContext
EbiContext context =
  com.sssw.fw.factory.EboFactory.createEbiContext(req, res,
  getPortletContext());
...
// get the re context out of fw context
com.sssw.re.api.EbiContext reContext =
  com.sssw.re.factory.EboFactory.createEbiContext(context);
```

- Pass EbiContext to methods implemented by exteNd Director manager objects to access subsystem-specific data.

  For example, the following code passes the EbiContext of the Framework subsystem to the content manager to get information from the Content Management subsystem:

```
...
//Get the EbiContext
EbiContext context =
  com.sssw.fw.factory.EboFactory.createEbiContext(req, res,
  getPortletContext());
...
//Find a document from the content management repository,
String html_path = req.getParameter("html_path");
...
EbiContentMgmtDelegate cmgr =
  com.sssw.cm.client.EboFactory.getDefaultContentMgmtDelegate()
;
EbiDocument doc =
(EbiDocument)cmgr.lookupDirectoryEntry(context,
html_path,EbiDocument.EL_DOCUMENT);
```

EbiPortalContext

Like the standard javax.portlet.PortalContext, the EbiPortalContext object gives portlets access to portal-specific information. The difference is that EbiPortalContext provides methods that return information specific to the exteNd Director portal, such as data relating to personal, shared, and container pages.

Getting a reference to EbiPortalContext

Portlets can reference EbiPortalContext objects by calling createEbiContext() methods on the EboFactory class for the Framework subsystem. To get an EbiPortalContext context object, use:

```
// Get a reference to the Portal Context object
EbiPortalContext ebiPortalContext = ( EbiPortalContext )
  request.getAttribute( EbiPortletConstants.EBI_PORTAL_CONTEXT );
```

Setting content type

You must set the content type on the render response for each portlet by calling the setContentType() method on the EbiResponse interface in the Framework subsystem.

IMPORTANT: Modes are based on content types. Therefore, It is strongly recommended that you set the content type **before** you set the mode in the doView(), doEdit(), and doHelp() methods—or in the render() method—of your portlet. This order of operations allows exteNd Director to validate modes against content type.

Synchronous versus asynchronous processing

You can set portlets to run synchronously or asynchronously. In exteNd Director, portlets run asynchronously by default. Each portlet runs in its own thread so content rendering can occur in parallel. The exteNd Director portal waits for each portlet on a page to complete the render operation or to time out (whichever comes first), then aggregates the rendered output onto the page.

Here are guidelines for determining whether portlets should run synchronously or asynchronously:

- Determine your application server's level of support for running portlet applications asynchronously, as described in the section on predeployment tasks.
- Portlets that do not use a lot of CPU time and do not call external sources should run in synchronous mode.

If you determine that one or more portlets should not run asynchronously in your environment, you can disable asynchronous portlet rendering, as described in the section on turning off asynchronous portlet rendering.

Asynchronous rendering of portlets

Asynchronous rendering of portlets is controlled by property settings in the following files:

- Framework service.xml file
- Framework config.xml file
- Portal config.xml file
- novell-portlet.xml file

This section describes the various settings that control asynchronous rendering.

NOTE: The service.xml and config.xml files for the Framework are located in WEB-INF\lib\ConfigService\FrameworkService-conf. The config.xml file for the Portal is located in WEB-INF\lib\ConfigService\PortalService-conf\config.xml.

Portlet thread pool settings

The **service.xml file for the Framework** specifies which class should be loaded as the thread pool implementation. The thread pool implementation class implements the EbiThreadPoolManager interface:

```
<service>
    <interface>com.sssw.fw.api.EbiThreadPoolManager</interface>
    <impl-class>com.sssw.fw.core.EboThreadPoolManagerImpl</impl-
class>
    <description>The thread pool manager for dealing with
multithreading</description>
    <max-instances>1</max-instances>
    <startup>M</startup>
</service>
```

The max-instances setting is ignored. This class is always handled as a singleton.

The **config.xml file for the Framework** specifies several parameters that control the behavior of the thread pool:

```
<property>
        <key>com.sssw.fw.api.threadpool.buffersize</key>
        <value>10</value>
    </property>
    <property>
        <key>com.sssw.fw.api.threadpool.maxthreads</key>
        <value>100</value>
    </property>
    <property>
```

```
        <key>com.sssw.fw.api.threadpool.minthreads</key>
        <value>4</value>
    </property>
    <property>
        <key>com.sssw.fw.api.threadpool.initialthreads</key>
        <value>10</value>
    </property>
    <property>
        <key>com.sssw.fw.api.threadpool.keepalifetime</key>
        <value>300000</value>
    </property>
    <property>
        <key>com.sssw.fw.api.threadpool.enabled_at_startup</key>
        <value>true</value>
    </property>
```

Each of these settings is described below:

Property	Description
com.sssw.fw.api.threadpool.buffersize	Defines how many portlet render requests should be buffered if the pool is busy. Once this limit is reached, the pool will no longer accept new render requests until it frees up again. Incoming requests will be diverted to the calling thread. In the case of the exteNd Application Server, this is the server's client thread.
com.sssw.fw.api.threadpool.maxthreads	The maximum number of threads to which the thread pool can grow.
com.sssw.fw.api.threadpool.minthreads	The minimum number of threads that will always be kept in the pool.
com.sssw.fw.api.threadpool.initialthreads	The number of threads immediately available in the pool after the pool has been started.
com.sssw.fw.api.threadpool.keepalifetime	The time (in milliseconds) a thread will be kept alive when it is idle.
com.sssw.fw.api.threadpool.enabled_at_startup	Determines whether the thread pool is enabled. This property must set to true if you want the thread pool to be enabled. If the thread pool is disabled, all parallel processing defaults to synchronous processing

Portal settings

The config.xml file for the Portal includes several properties that control asynchronous processing for the Portal as a whole:

```
<property>
 <key>com.novell.afw.portal.aggregation.default_request_timeout
 </key>
 <value>2000</value>
</property>
<property>
 <key>com.novell.afw.portal.aggregation.max_request_timeout</key>
 <value>2500</value>
</property>
<property>
 <key>com.novell.afw.portal.aggregation.render_synch_in_main_thread
 </key>
 <value>true</value>
</property>
```

Each of these settings is described below:

Property	Description
com.novell.afw.portal.aggregation.default_request_timeout	This is the default time (in milliseconds) that a request will wait before it times out. If none of the asynchronous portlets defines a timeout, or none of the portlets defines a timeout that is bigger then this value, this default value will be used. If one or more of the portlets to render defines a timeout that is bigger than this default value, the bigger one will be used instead of the default. This setting can be used to protect the application from getting too many messages indicating that portlets have timed out (which might happen if the portlet's descriptor, novell-portlet.xml, defines values that are too small). **NOTE:** In the event that all portlets can be rendered before this default timeout occurs, the request will immediately return to the client.

Property	Description
com.novell.afw.portal.aggregation.max_request_timeout	This is the maximum time (in milliseconds) that a request will be held back from finishing. This means that after this amount of time, every request will return to the client, regardless of whether any portlet defines a bigger timeout value. This setting can be used to make sure that the Portal responds in a timely fashion even if one or more of the portlets define a large timeout value.
com.novell.afw.portal.aggregation.render_synch_in_main_thread	Determines whether synchronous portlets are executed within the application server's client thread. This is set to true by default, meaning that all synchronous portlets are executed in the application server's client thread. Setting this property to true can present some problems. For example, if one of the synchronous portlets uses an IO operation and blocks on a Socket read indefinitely, the application server thread will be blocked indefinitely as well. To prevent the application server's client thread pool from such a situation, you can set this property to false. When this property is set to false, all synchronous portlets are still be processed sequentially by the same thread, but get a new thread from the portlet thread pool, so they don't execute in the application server's client thread.

Portlet max-timeout setting

The **novell-portlet.xml** file includes a property called **max-timeout** that determines the maximum timeout interval for each portlet:

```
<portlet name="StockQuotePortlet">
 <style>
  <name>StockQuotePortletDefault</name>
  <display-name>Default StockQuote Portlet Style</display-name>
  <user-agent>
   <device-name>Generic_HTML</device-name>
   <file-name>$RESOURCE_SET$/portlet-style/StockQuote.xsl
   </file-name>
```

```
  </user-agent>
 </style>
  <requires-authentication>0</requires-authentication>
  <auto-register enabled="true">
   <category>General Portlets</category>
  </auto-register>
  <max-timeout>1000</max-timeout>
</portlet>
```

The max-timeout property sets the maximum time (in milliseconds) that the Portal should wait for a portlet's render request to finish.

NOTE: A value of 0 is interpreted as -1, which indicates that this portlet does not have a timeout value and the default request timeout will be applied.

How the request timeout is determined

The timeout of a request is determined as follows:

1. The timeout of each asynchronous portlet is read and the greatest value is determined.
2. If the value from step 1 is between the default_request_timeout and the max_request_timeout, it will be used as this request's maximum timeout.
3. If the value from step 1 is below the default_request_timeout, the timeout for this request will be set to the default_request_timeout.
4. If the value from step 1 is above the max_request_timeout value, the max_request_timeout value will be used as the timeout value for the current request.

The max_request_timeout setting applies to synchronous portlets, as well as asynchronous portlets. Since synchronous portlets are processed sequentially, the timeout is checked after each portlet returns. If the max_request_timeout value is reached, no new render process is started and the default timeout message is used as the portlet's content for all remaining, unprocessed synchronous portlets.

NOTE: You should not confuse the request timeout settings with the expiration cache setting in the portlet.xml file. The expiration cache setting is used for content caching.

Getting information about portlets

This section describes how to get information about portlet preferences and settings.

Getting and storing portlet preferences

exteNd Director provides a flexible paradigm that allows you to apply portlet preferences at four levels of priority, listed here from lowest to highest priority:

- Definition-level preferences are defined on the portlet definition and cannot be modified after deployment
- Registration-level preferences are defined on a portlet instance at registration (also known as a *portlet registration*)
- Page Assignment-level preferences are defined on a portlet registration when it is assigned to a page (also known as a *portlet page instance*).
- User-level preferences defined for a portlet page instance by a user at runtime

 For more information, see the section on portlet preferences.

To get and set preferences, you typically call methods on the standard javax.portlet.PortletPreferences interface. These methods get and set values for any given preference at the highest level of priority that is available.

For more advanced customization—for example to get and set preferences at specific levels of priority—see the methods provided by the EbiPortletInfoManager.

Getting and setting portlet settings

exteNd Director defines a group of settings that determine how portlets interact with the portal, as described in the section on portlet settings. You can get and set these values by calling methods on the EbiPortletSettings interface.

Styling portlets that generate XML content

In exteNd Director, you can style a portlet that generates XML content by specifying the **<style>** element in the novell-portlet.xml deployment descriptor or in the associated portlet fragment deployment descriptor.

You can specify one <style> element per portlet, but each style can contain multiple user agents that point to device-specific XSL style sheets.

For more information about how to specify the <style> element, see the section that presents a look inside novell-portlet.xml or look at the novell-portlet.xml schema in:

```
extend5 install directory\Common\Resources\SchemaCatalog\novell-portlet.xsd
```

Default implementation for Edit mode

exteNd Director provides a default implementation for Edit mode.

If you enable the Edit option for a portlet, but don't support the Edit mode explicitly, exteNd Director displays a preference sheet of all the preferences that have been defined for the portlet and that can be edited by the end-user.

If you want to use this default implementation, follow these guidelines:

- Make sure you define all the preferences of interest in the portlet fragment deployment descriptor of your portlet.
- Enable the Edit option in the portlet fragment deployment descriptor, as follows:

  ```
  <supported-option>edit</supported-option>
  ```
- Do **not** enable Edit mode in the **<supports>** element of the portlet fragment deployment descriptor.

IMPORTANT: If you enable the Edit option and **do** support the Edit mode by implementing a doEdit() method, be sure to enable the Edit mode **and** Edit option in the portlet fragment deployment descriptor as follows:

- Add a **<portlet-mode>** entry under the **<supports>** element:

  ```
  <supports>
    <portlet-mode>edit</portlet-mode>
  </supports>
  ```
- Add a **<supported-option>** element:

  ```
  <supported-option>edit</supported-option>
  ```

Specifying a secure port for portlet URLs

When you create a render or action URL by calling the appropriate method on `javax.portlet.RenderResponse`, you can set whether or not the URL is a secure URL by calling `PortletURL.setSecure(true)`.

By default, the secure port is 443. If you have configured your server to use a different secure port, you can instruct the exteNd Director portal to use your port when creating portlet URLs. add a property to the PortalService config.xml as follows:

➢ **To specify a secure port for portlet URLs:**

- Add a property to the Portal subsystem configuration file, as follows:

```
<property>
<key>portlet-url-secure-port</key>
<value>444</value>
</property>
```

Getting and setting cookies on a portlet

The exteNd Director API allows you to get and set cookies on a portlet by using methods in the EboCookieUtil class of the Framework subsystem.

To get cookies, you can call methods that propagate headers from the HTTP request down to the portlet request properties:

Method	Description
getCookieValue()	Gets the value of a specified cookie
getCookie()	Gets a specified cookie from the HTTP request
getCookies()	Returns an array of cookies from the HTTP request

To set cookies, you can call the addCookieToResponse() method, which propagates the set-cookie header from the PortletResponse properties to the HTTP response back to the client.

You can also get and set cookies by using the Request and Response scoped paths, as described in the chapter on working with scoped paths and XPaths.

III Tools

Explains how to use Director tools to personalize and administer your portal

14 Personalizing Your Portal

This chapter explains how to personalize your portal by creating personal pages, configuring content and layout, and applying themes. It covers the following topics:

About the Portal Personalizer

The *Portal Personalizer* is a facility provided with the portal that allows you to create personal pages and customize your portal environment to show personalized content. All portal users can access the Portal Personalizer to perform the following tasks:

- Create personal pages
- Apply a theme to your portal
- Set a default personal page
- Set default container and shared pages
- Delete personal pages
- Create a portal wireless layout

Starting the Portal Personalizer

You can start the Portal Personalizer portlet directly in your browser or through the Portal Home Page.

➢ **To start the Portal Personalizer:**

1. Make sure your server is running.
2. Choose one of two starting points:
   - Start directly
   - Start from the home page

- **To start directly**:

  2a Enter this URL in your browser:

  ```
  http://server/project context/portal/portlet/Personalize
  ```

  The exteNd Director Login portlet opens in your browser and prompts you to log in to the Portal:

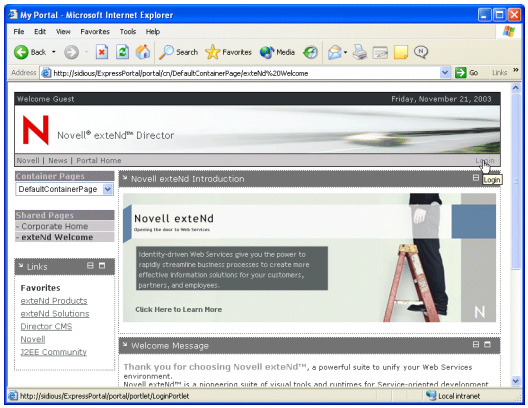
  2b Enter your user name and password, then click **Login**.

- **To start from the home page**:

  2a Enter this URL to go to the Portal Home Page:

  ```
  http://server/project context/portal
  ```

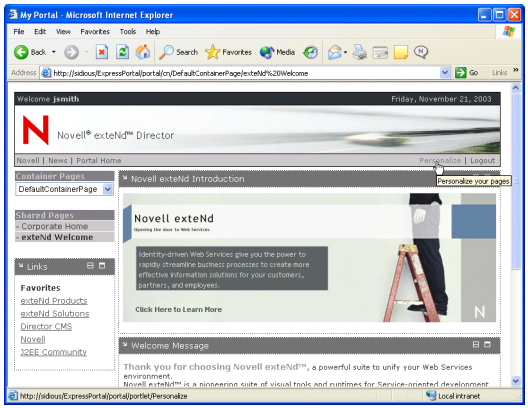
2b Click **Login** on the Portal Home Page:



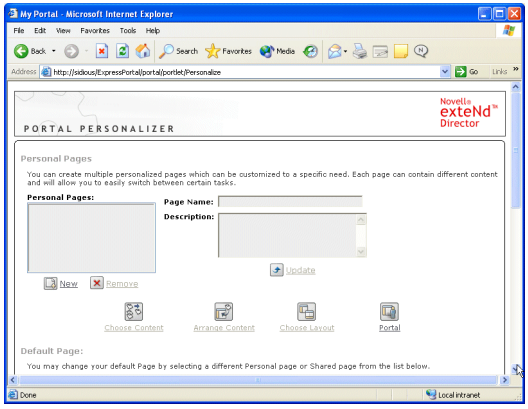
The exteNd Director Login portlet opens in your browser and prompts you to log in to the portal:

Enter your user name and password, then click **Login**.

2c Click **Personalize** on the Portal Home Page:



The Portal Personalizer opens in your browser:



Now you are ready to create personal pages, assign default pages, apply themes, and create a wireless layout.

Creating personal pages

Any authenticated portal user can define a personal page. The process involves the following steps:

1. Create a new page using the **Portal Personalizer**, as described in "Creating a personal page" on page 211.
2. Add content—in the form of portlets—to the page using the **Content Selector** portlet, as described in "Adding content to a personal page" on page 212.
3. Choose a portal layout, as described in "Modifying the page layout" on page 214.

4 Arrange the order and position of content on the selected layout using the **Layout Selector** portlet, as described in "Arranging content on the personal page" on page 215.

5 Display the new page right away, as described in "Displaying a personal page" on page 218.

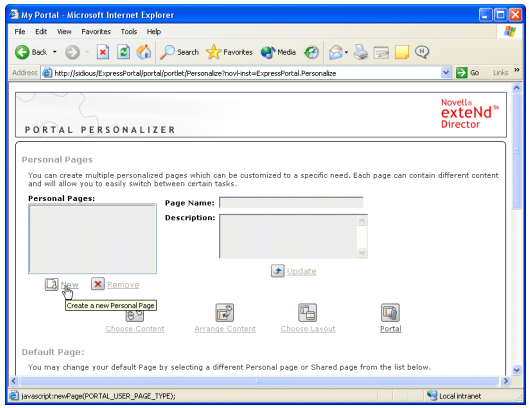
Pages created with the Portal Personalizer are not tightly bound to portal layouts. That means users can switch layouts for their pages without disrupting the page contents. When the user applies a new layout to a page, any portlets that have been added to the page are automatically displayed using the new layout.

Creating a personal page

➢ **To create a personal page:**

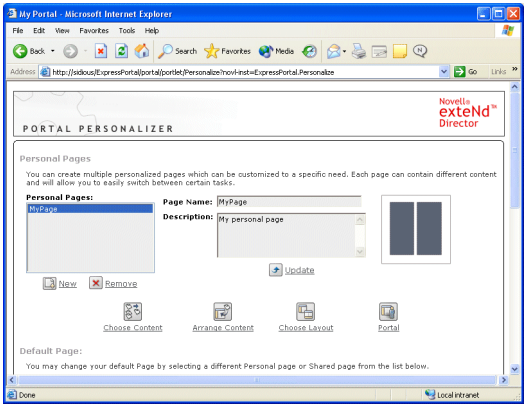
1 Start the Portal Personalizer, as described in "Starting the Portal Personalizer" on page 208.

2 Click New under the personal pages list box.



An untitled personal page is created.

3 Enter a name for the page in the **Page Name** field and optionally enter a description in the **Description** field, as in this example:



4 Click **Update**.

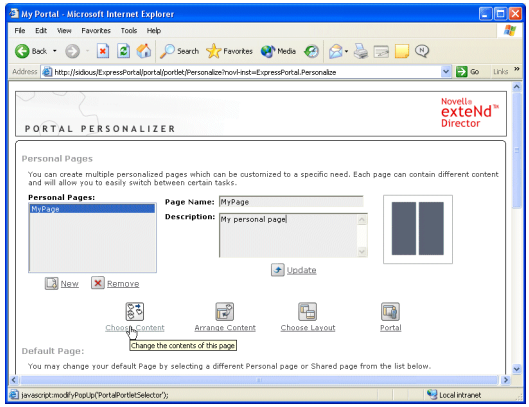
Now you are ready to add content to the personal page. See "Adding content to a personal page" on page 212.

Adding content to a personal page

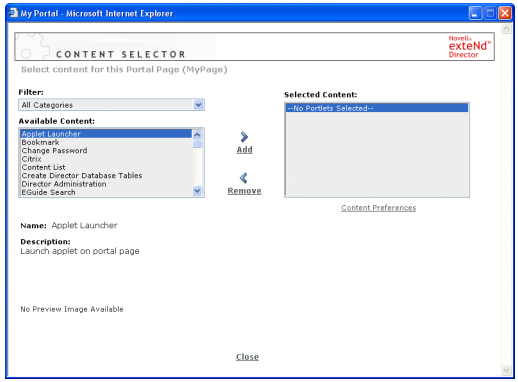
After creating a personal page, the next step is to add content by selecting portlets to place on the page. To add content, you access the **Content Selector** from the Portal Personalizer. You select from a list that includes prebuilt portlets supplied with exteNd Director along with any custom portlets you have created and registered.

➤ **To add content to a personal page:**

1. Start the Portal Personalizer, as described in "Starting the Portal Personalizer" on page 208.
2. Select a page from the personal pages list and click **Choose Content**.



The **Content Selector** opens in a new browser window.



3. If you want to display a specific category of available portlets, choose a category from the **Filter** dropdown menu.
4. Select one or more portlets from the list of **Available Portlets**.

   TIP: Hold down the **Control** key to select multiple non-contiguous portlets from the list; use the **Shift** key to make multiple contiguous selections.
5. Click **Add** to move your choices to the list of **Selected Portlets**.
6. Click **Close**.

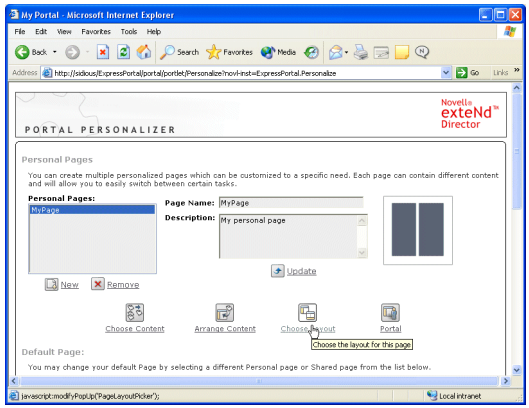
Now that you have chosen the content for your personal page, you can select a new layout as described in "Modifying the page layout" on page 214, or arrange the content on the current layout as described in "Arranging content on the personal page" on page 215.

Modifying the page layout

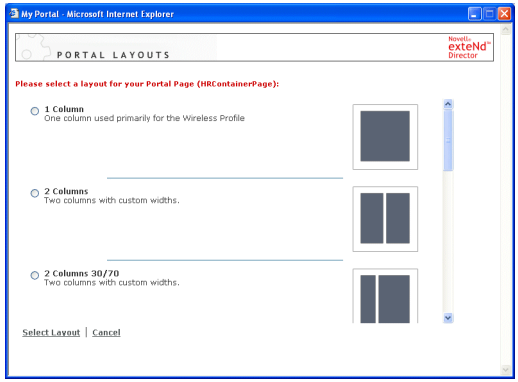
When you modify the layout of an existing page, the Portal shifts existing contents to accommodate the new layout. Although the Portal makes its best guess on content placement, you may need to fine tune the end result. For example, if you were to change the layout from 3 Columns to 2 Columns, you would probably want to make some adjustments to get the desired presentation.

➢ **To modify the layout of a personal page:**

1. Start the Portal Personalizer, as described in "Starting the Portal Personalizer" on page 208.
2. Select **Choose Layout**.



A layout selection page opens in a new browser window.



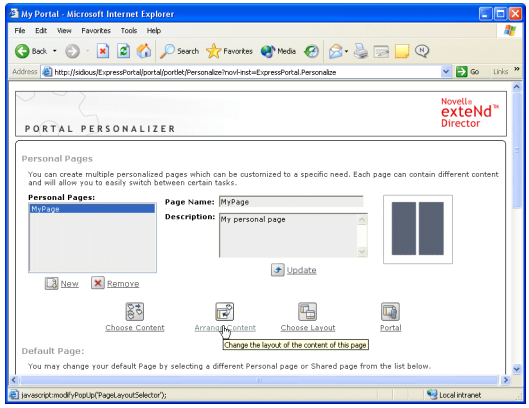
3. Scroll through the choices and select the layout of interest.
4. Click **Select Layout**.

Arranging content on the personal page

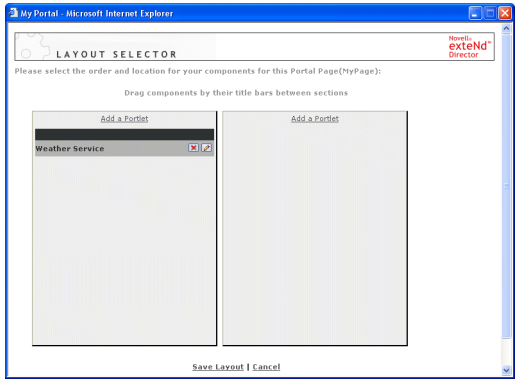
After you have designated the content and layout for your personal page, you can position the content in the selected layout.

➢ **To arrange content on a personal page:**

1. Start the Portal Personalizer, as described in "Starting the Portal Personalizer" on page 208.
2. Select **Arrange Content**.



The Layout Selector opens in a new browser window:



The portlets that you have already added appear in the page layout. By default, portlets are added in top-to-bottom order within the first section.

3. If you want to add a portlet to the page, follow these steps:

   3a Click **Add a Portlet** in the desired layout frame.

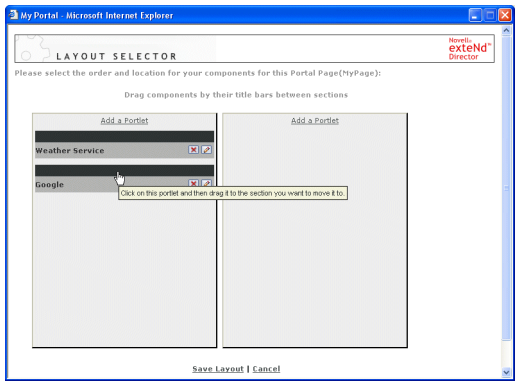
   The Content Selector opens in a new browser window.

   3b If you want to display a specific category of available portlets, choose a category from the **Filter** dropdown menu.

3c Double click the portlet of interest from the list of **Available Portlets**.

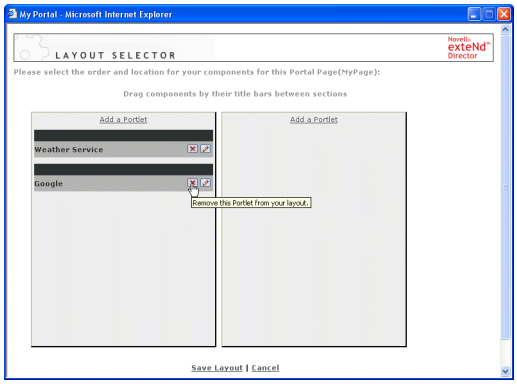
The Content Selector window closes and the portlet you selected appears in the target layout frame in the Layout Selector window.

4 If you want to move a portlet to a different location in the layout, follow these browser-specific steps

For	Do this
Internet Explorer	**1** Move your cursor over the title bar of the portlet until the cursor changes to a hand shape.  **2** Hold down the left mouse button and drag the portlet to the desired location in the layout.
Netscape/Mozilla	**1** Click on the portlet you want to move. **2** Click inside the destination layout frame. The portlet moves to the destination.

5 If you want to remove a portlet from the layout, follow these steps:

5a Click the **X** in the upper right hand corner of the portlet:



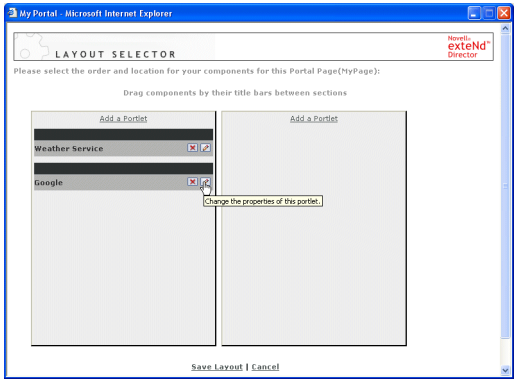
A message box appears asking you to confirm the deletion.

5b Click **OK**.

The portlet is removed from the layout.

6 If you want to edit the preferences of a portlet, follow these steps:

6a Click the pencil icon in the upper right-hand corner of the portlet:



The **Portlet Preferences** portlet opens in your browser.

6b Specify your preferences. The preference values you specify take effect for the instance of the portlet that appears on your page.

7 Click **Save Layout** to save your changes and close the Layout Selector.

Displaying a personal page

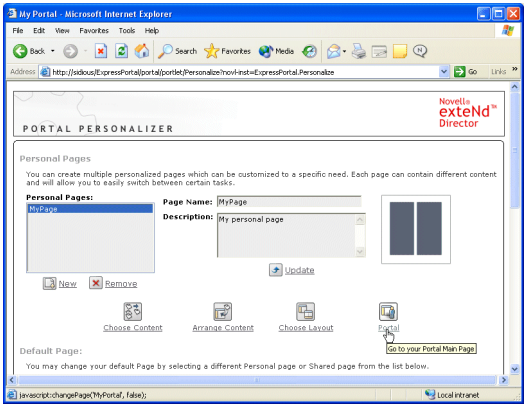
There are two ways to display a personal page:

Method	Displays	Prerequisites
Use the Portal link provided on the Portal Personalizer	Content of the personal page aggregated with a container page	Set a default container page that includes the Page Navigation portlet.
Enter the personal page URL in your browser	Content of the personal page	None

This section describes each procedure.

➢ **To display a personal page using the Portal:**

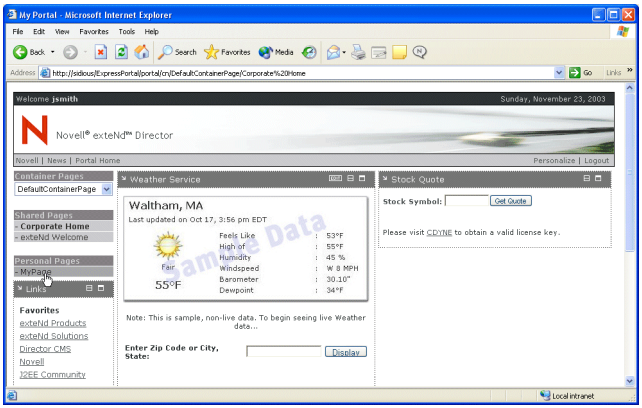
1 Select **Portal** from the Portal Personalizer, as shown below.



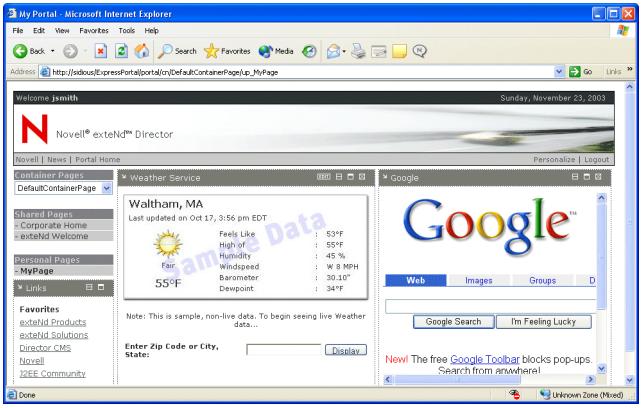
NOTE: To start the Portal Personalizer, see "Starting the Portal Personalizer" on page 208.

The personal page is displayed in the context of the container page. The Page Navigation portlet on the container page highlights the personal page or shared page that appears in the content area.

2 If the personal page is not displayed in the content area of the container page, click the link to the personal page in the Page Navigation portlet, as in this example:



As you can see, this container page displays the content of the shared page **Corporate Home**, shown highlighted in the page navigation links. Selecting the navigation link to MyPage will direct the Portal Aggregator to display the contents of that personal page instead. In the following example, the link to MyPage now appears highlighted in the Page Navigation portlet, and the contents of the personal page—the Weather Service and Google portlets—now appears in the content area of the container page:



➢ **To display a page directly using the personal page URL:**

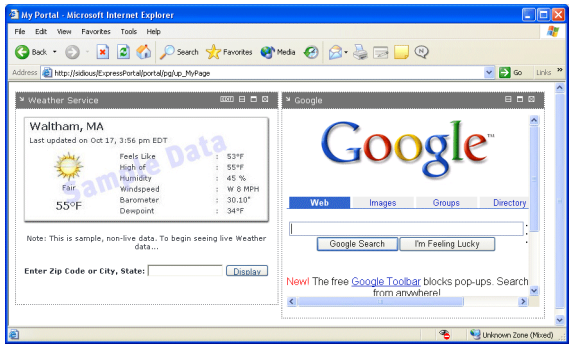
- Enter the following URL in your browser:

  `http://server/project context/portal/pg/up_personal page name`

  For example, if your server is **localhost** and your project context is **MyWAR**, you can access a personal page called **MyPage** by entering this URL in your browser:

  `http://localhost/MyWAR/portal/pg/up_MyPage`

  When you navigate to the personal page URL, the content of the personal page appears by itself in your browser. It does not include the container page, as shown in this example:

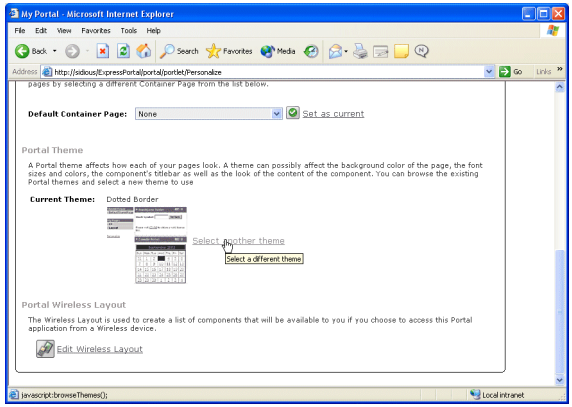


Setting the theme for your portal

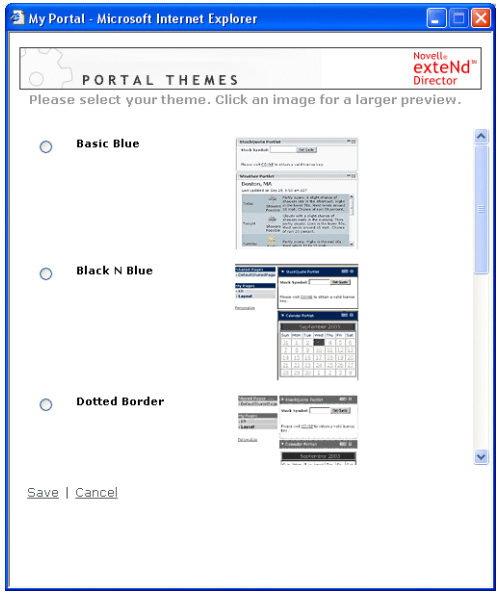
You can change the theme for your portal using the Portal Personalizer. The theme applies to all pages—user, shared, and container pages—in your portal.

➢ **To set the theme for your portal:**

1. Start the Portal Personalizer, as described in "Starting the Portal Personalizer" on page 208.
2. Scroll down to the Portal Theme section and click **Select another theme**.



exteNd Director displays the Portal Themes page in a new browser window.



3. Click the radio button for the theme you want to use.
4. Click **Save**.

   The new theme takes effect immediately.

Setting the default personal or shared page

In the Portal Personalizer, you can designate a specific personal or shared page to be the default page for your portal. The page you designate is then displayed by the default Container page.

➢ **To set a default page:**

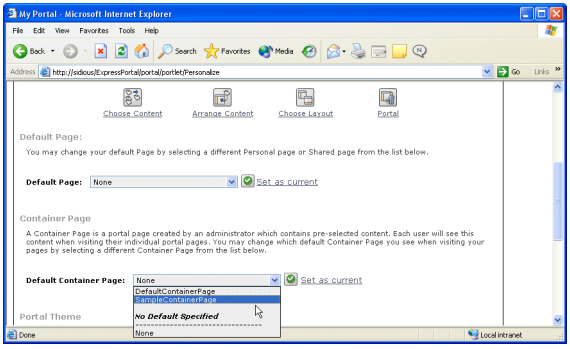
1. Start the Portal Personalizer, as described in "Starting the Portal Personalizer" on page 208.
2. Select the page in the **Default Page** list.
3. Click **Set as default**.

Setting the default container page

In the Portal Personalizer, you can specify a default container page to wrap all of your personal pages and a default page to display in the content area of your container page if no other page takes precedence. The Portal Aggregator uses these settings to determine what content to display for each user request, as described in "How content is determined for the current user" on page 49.

➢ **To set a default container page:**

1. Start the Portal Personalizer, as described in "Starting the Portal Personalizer" on page 208.
2. Scroll to the **Container Page** section and select a container page from the dropdown menu in the **Default Container Page** field.



NOTE: The dropdown menu shows only the container pages you are authorized to view.

Deleting a personal page

➢ **To delete a personal page:**

1. Start the Portal Personalizer, as described in "Starting the Portal Personalizer" on page 208.
2. Select the page in the **Personal Pages** list.
3. Click **Remove**.

   A message box opens, asking you to confirm the deletion.
4. Click **OK**.

Creating a wireless layout page

From the Portal Personalizer you can access the Wireless Layout Manager, which allows you to a define a layout page containing wireless-enabled portlets that you can use when accessing your portal application from a wireless device.

To learn how to create a wireless layout, see "Using the Wireless Layout Manager" on page 134.

15 Administering the Portal

This chapter explains how portal administrators can use the Portal Administration tool (PortalPageAdmin portlet) to manage the portal environment. It covers the following topics:

- About the portal administrator ACL
- Portal administrator tasks
- About the Portal Administration tool
- Creating and maintaining container pages
- Creating and maintaining shared pages
- Assigning pages to users and groups
- Choosing a default shared page for a container page

NOTE: This chapter does not cover the Portal Management and Portlet Management sections of the Director Administration Console (DAC).

 For details on the Portal Management section of the DAC, see Chapter 20, "Using the Portal Management Section of the DAC". For details on the Portlet Management section of the DAC, see Chapter 21, "Using the Portlet Management Section of the DAC".

About the portal administrator ACL

A user who acts as the portal administrator is responsible for configuring, managing, and maintaining the portal environment for an organization. In exteNd Director, the user designated to be the portal administrator must be assigned to the **PortalAdmin** ACL.

An administrator assigns users to administrative groups and permissions in the Director Administration Console (DAC), as described in Using the Security Management Section of the PAC.

Portal administrator tasks

In exteNd Director, portal administrators can perform the following tasks:

Task	Tool	For more information, see
Create and maintain container pages to establish a corporate look and feel for the portal	Portal Administration (PortalPageAdmin portlet)	"Creating and maintaining container pages" on page 230
Create and maintain shared pages to manage the distribution of common information to users and groups logged in to the portal	Portal Administration (PortalPageAdmin portlet)	"Creating and maintaining shared pages" on page 241
Assign pages to users and groups	Portal Administration (PortalPageAdmin portlet)	"Assigning pages to users and groups" on page 252
Assign owners to shared pages	Portal Administration (PortalPageAdmin portlet)	"Assigning pages to users and groups" on page 252
Assign a default shared page to a container page	Portal Administration (PortalPageAdmin portlet)	"Choosing a default shared page for a container page" on page 255
Assign categories to pages to restrict the flow of information within ACL groups	Portal Administrator portlet	Chapter 8, "Working with Page Categories"

Task	Tool	For more information, see
Managing portlets	Portlet Management section of the DAC	Chapter 21, “Using the Portlet Management Section of the DAC”
Managing the portal	Portal Management section of the DAC	Chapter 20, “Using the Portal Management Section of the DAC”

About the Portal Administration tool

As its name suggests, the *Portal Administration* tool is designed for portal administrators. It provides utilities for:

- Creating container pages, and assigning them to users and groups
- Creating shared pages, and assigning them to users and groups
- Assigning ownership of shared pages

Who can use the Portal Administration tool

You can use the Portal Administration tool if you meet **EITHER** of these requirements:

Requirement	Description	What you can do in the Portal Administration tool
You are a portal administrator	You belong to the PortalAdmin administrative group with PROTECT permission	Use all functions
You own one or more shared pages	A portal administrator assigned you OWNERSHIP permission for one or more shared pages	Modify content and layout of the shared pages that you own

Starting the Portal Administrator

You can start the Portal Administration tool directly in your browser or from the default portal page. This section describes each method.

➢ To start the Portal Administration tool from your browser:

1. Make sure your server is running.
2. Enter this URL in your browser:

   `http://server/project context/portal/portlet/PortalPageAdmin`

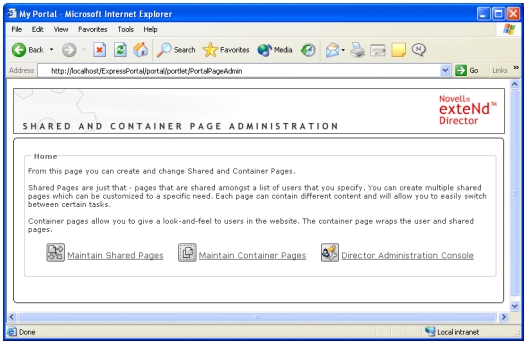
   For example, if your server is localhost and your project is MyWAR, you would enter this URL:

   `http://localhost/MyWAR/portal/portlet/PortalPageAdmin`

   The exteNd Director Login portlet opens in your browser and prompts you to log in to the portal:

3. Enter your user name and password, then click **Login**.

   The Portal Administration tool opens in your browser:



➢ **To start the Portal Administration tool from the default portal page:**

1. Enter this URL in your browser:

   `http://`*`server/project context`*`/portal/cn/DefaultContainerPage`

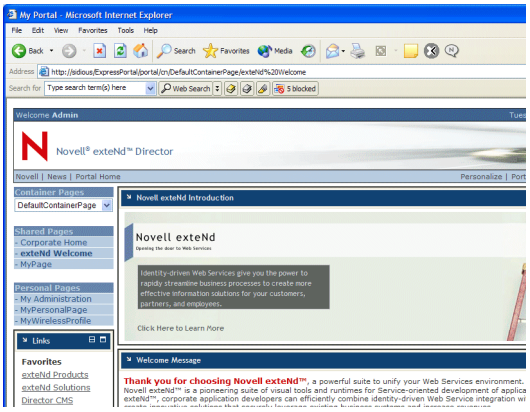
   For example, if your server is localhost and your project is MyWAR, you would enter this URL:

   `http://localhost/MyWAR/portal/cn/DefaultContainerPage`

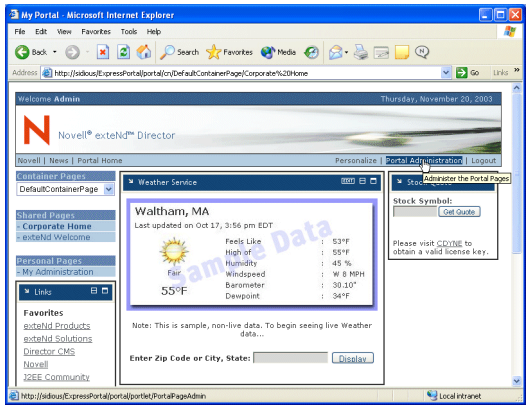
   The exteNd Director Login portlet opens in your browser and prompts you to log in to the portal:

2. Enter your user name and password, then click **Login** or the **Enter** key.

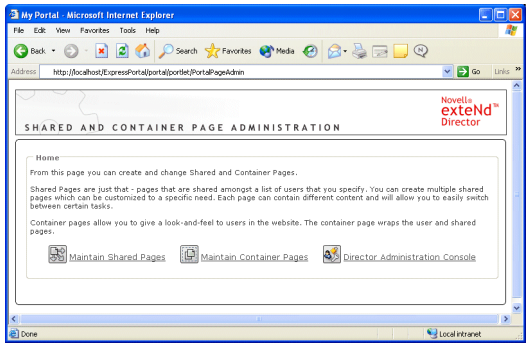
   The default portal page opens in your browser:



3 Click **Portal Administration**, a link on the upper right side of the page:



The Portal Administration tool opens in a new browser window:



Creating and maintaining container pages

Only portal administrators can create and maintain container pages. The process involves the following steps:

1 Create a new container page or select an existing container page using the **Portal Administration** tool, as described in "Creating container pages" on page 231.

2 Add content—in the form of portlets—to the page using the **Portlet Selector** portlet, as described in "Adding content to a container page" on page 233.

3 Choose a portal layout, as described in "Modifying the layout of a container page" on page 235.

4 Arrange the order and position of content on the selected layout using the **Layout Selector** portlet, as described in "Arranging content on the container page" on page 237.

5 Display the new page right away by entering the container page URL in your browser, as described in"Displaying a container page" on page 240.

Container pages are not tightly bound to portal layouts. That means portal administrators can switch layouts for their container pages without losing any page contents. When the portal administrator applies a new layout to a container page, any portlets that have been added to the page are automatically displayed using the new layout. You may need to fine-tune the content placement in the new layout.

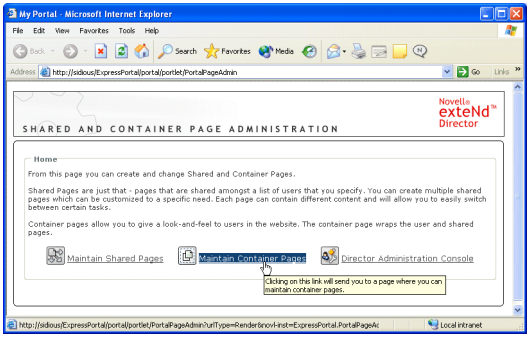
Creating container pages

Portal administrators can create container pages from scratch or by copying existing pages. This section describes both procedures.

➢ **To create a container page from scratch:**

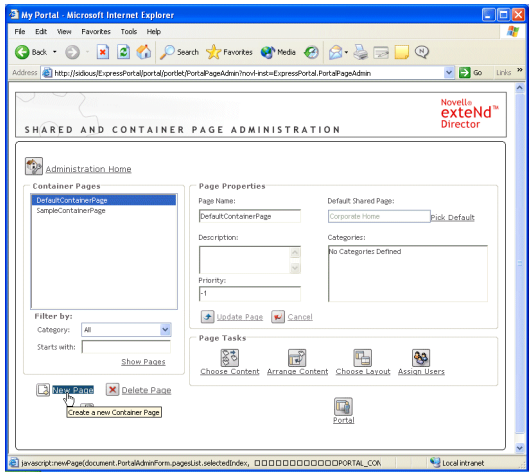
1 Start the Portal Administration tool, as described in "Starting the Portal Administrator" on page 228.

2 Select **Maintain Container Pages**.



The container page section of the Portal Administration tool opens in your browser.

3 Select **New Page** at the bottom of the page:



An untitled, uncategorized container page is created.

4 Enter a name for the new container page in the **Page Name** field.

5 Enter other optional page properties as needed:

Property	What to specify
Description	Enter text that describes the page.
Default Shared Page	See "Choosing a default shared page for a container page" on page 255.
Categories	See Chapter 8, "Working with Page Categories".

6 Select **Update Page**.

➢ **To create a container page by copying an existing page:**

1 Search for and select the page you want to copy in the list of container pages.

2 Select **Copy Page** at the bottom of the page.

A new container page is created with the name **Copy of *Original Page Name***.

3 Enter a name for the new container page in the **Page Name** field.

4 Enter other optional page properties as needed:

Property	What to specify
Description	Enter text that describes the page.
Default Shared Page	See "Choosing a default shared page for a container page" on page 255.
Categories	See Chapter 8, "Working with Page Categories".

5 Select **Update Page**.

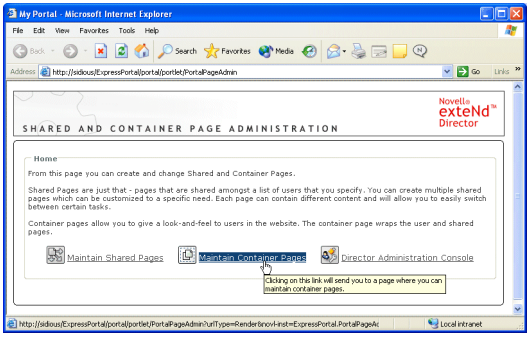
Adding content to a container page

After creating a container page, the next step is to add content by selecting portlets to place on the page. To add content to a new or existing container page, you access the **Portlet Selector** portlet from the Portal Administration tool. You select from a list that includes prebuilt portlets supplied with exteNd Director along with any custom portlets you have created and registered.

➢ **To add content to a container page:**

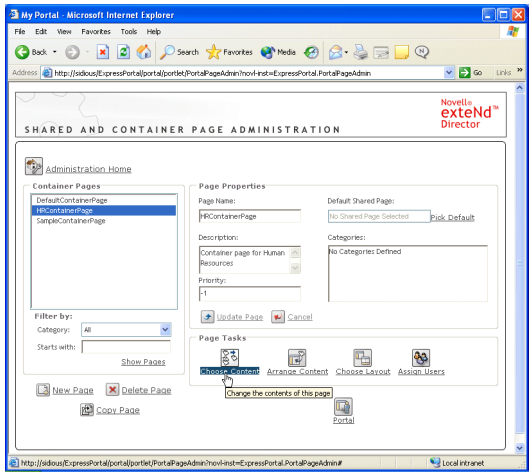
1 Start the Portal Administration tool, as described in "Starting the Portal Administrator" on page 228.

2 Select **Maintain Container Pages**.

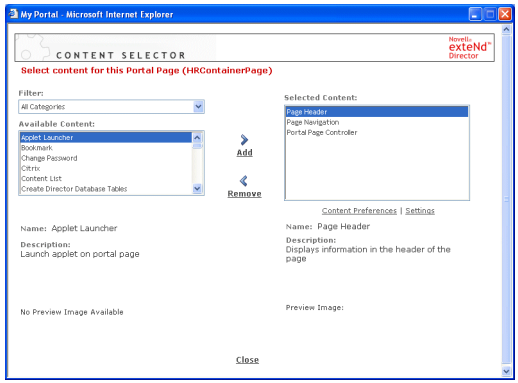


The container page section of the Portal Administration tool opens in your browser.

3 Select a page from the Container Pages list and click **Choose Content**.



The Portlet Selector opens in a new browser window.



4 If you want to display a specific category of available portlets, choose a category from the **Filter** dropdown menu.

5 Select one or more portlets from the list of **Available Portlets**.

TIP: Hold down the **Control** key to select multiple non-contiguous portlets from the list; use the **Shift** key to make multiple contiguous selections.

6 Click **Add** to move your choices to the list of **Selected Portlets**.

NOTE: You can edit the preferences of one or more portlets that you have selected to be added to your container page. The preference values you specify take effect for the instance of the portlet that appears on your page.

7 Click **Close**.

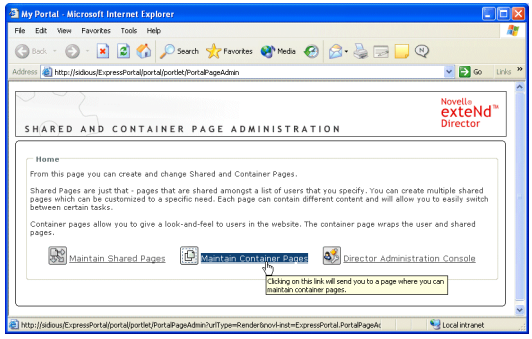
Now that you have chosen the content for your container page, you can select a new layout as described in "Modifying the layout of a container page" on page 235, or arrange the content on the current layout as described in "Arranging content on the container page" on page 237.

Modifying the layout of a container page

When you modify the layout of a container page, the Portal Aggregator shifts existing content to accommodate the new layout. Although the Portal Aggregator makes its best guess on content placement, you may need to fine tune the end result.

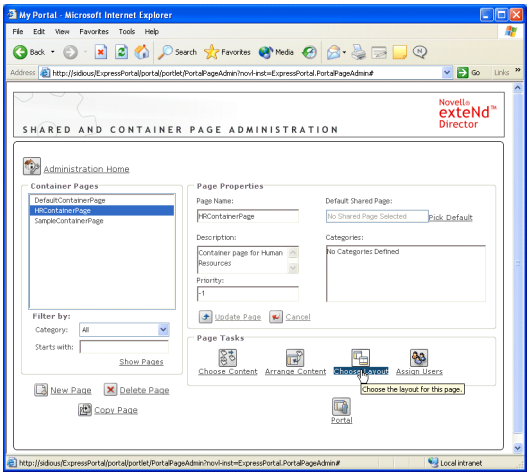
➢ **To modify the layout of a container page:**

1. Start the Portal Administration tool, as described in "Starting the Portal Administrator" on page 228.
2. Select **Maintain Container Pages**.

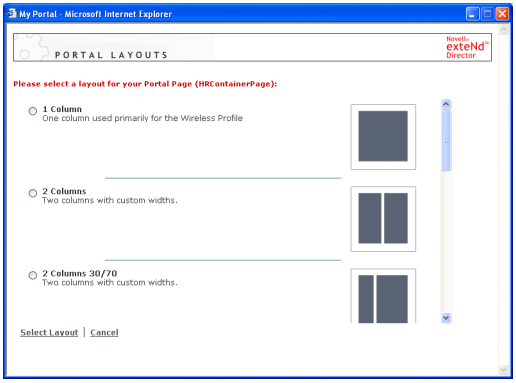


The container page section of the Portal Administration tool opens in your browser.

3 Select a page from the Container Pages list and click **Choose Layout**.



A layout selection page opens in a new browser window.



4 Scroll through the choices and select the layout of interest.

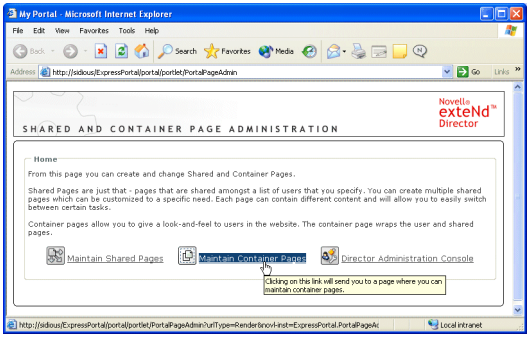
5 Click **Select Layout**.

Arranging content on the container page

After you have designated the content and layout for your container page, you can position the content in the selected layout, add other portlets in specific locations, or delete portlets.

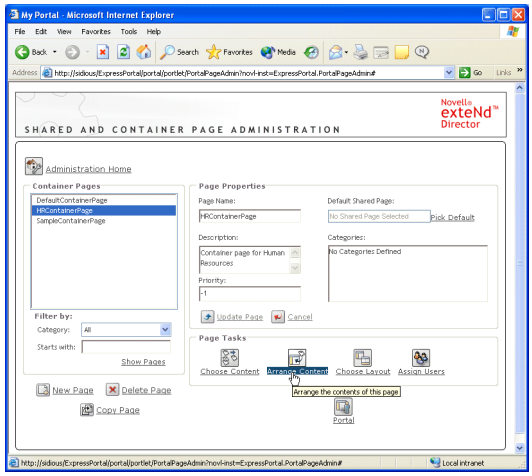
➢ **To arrange content on a container page:**

1. Start the Portal Administration tool, as described in "Starting the Portal Administrator" on page 228.
2. Select **Maintain Container Pages**.

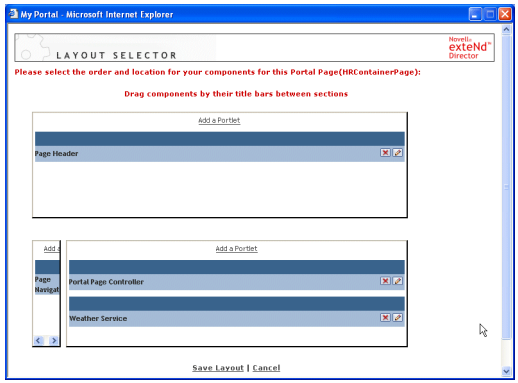


The container page section of the Portal Administration tool opens in your browser.

3 Select a page from the Container Pages list and click **Arrange Content**.



The Layout Selector portlet appears in a new browser window.



The portlets that you have already added appear in the page layout. By default, portlets are added in left-to-right, top-to-bottom order.

4 If you want to add a portlet to the page, follow these steps:

4a Click **Add a Portlet** in the desired layout frame.

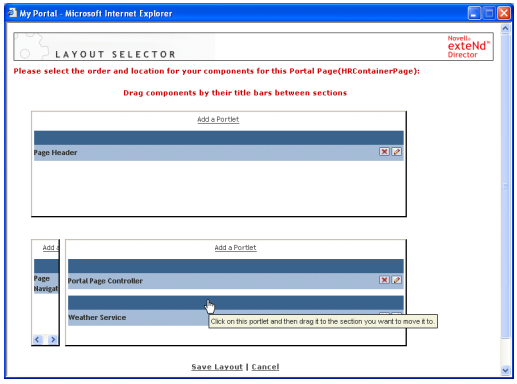
The Portlet Selector opens in a new browser window.

4b If you want to display a specific category of available portlets, choose a category from the **Filter** dropdown menu.

4c Double click the portlet of interest from the list of **Available Portlets**.

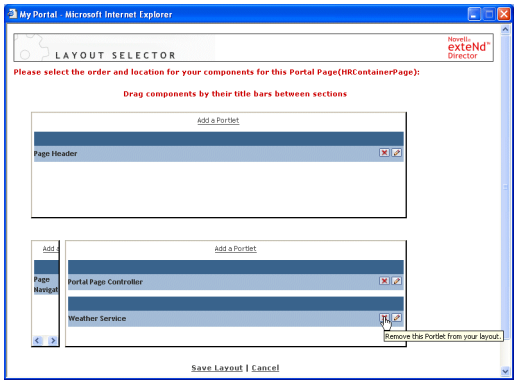
The Portlet Selector window closes and the portlet you selected appears in the target layout frame in the Layout Selector window.

5 If you want to move a portlet to a different location in the layout, follow these browser-specific steps:

For	Do this
Internet Explorer	**1** Move your cursor over the title bar of the portlet until the cursor changes to a hand shape.  **2** Hold down the left mouse button and drag the portlet to the desired location in the layout.
Netscape	**1** Click on the portlet you want to move. **2** Click inside the destination layout frame. The portlet moves to the destination.

6 If you want to remove a portlet from the layout, follow these steps:

6a Click the X in the upper right hand corner of the portlet:



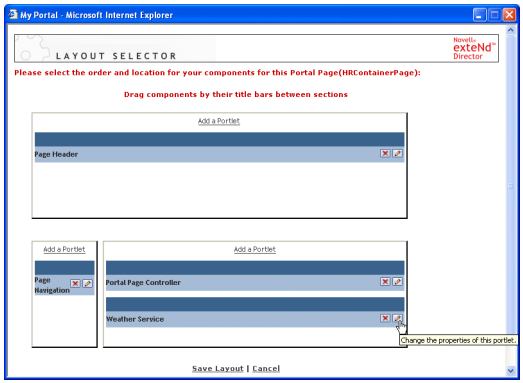
A message box appears, asking you to confirm the deletion.

6b Click OK.

The portlet is removed from the layout.

7 If you want to edit the preferences of a portlet, follow these steps:

7a Click the pencil icon in the upper right-hand corner of the portlet:



The **Portlet Preferences** tool opens in your browser.

7b Specify your preferences. The preference values you specify take effect for the instance of the portlet that appears on your page.

8 Click **Save Layout** to record your changes and close the Layout Selector.

Displaying a container page

You can display your page by entering the container page URL in your browser.

➢ **To display a container page:**

- Enter the following URL in your browser:

  `http://server/project context/portal/cn/container page name`

  For example, if your server is **localhost** and your project context is **MyWAR**, you can access a container page called **DefaultContainerPage** by entering this URL in your browser:

  `http://localhost/MyWAR/portal/cn/DefaultContainerPage`

Creating and maintaining shared pages

Two types of users can work with shared pages:

- Portal administrators can create and maintain all shared pages, and assign ownership of specific shared pages to other users.
- Shared page owners—whether portal administrators or not—can modify the content and layout of shared pages that they own.

The process of creating and maintaining shared pages involves the following steps:

1. Create a new shared page or select an existing shared page using the **Portal Administration** tool, as described in "Creating shared pages" on page 242.
2. Add content—in the form of portlets—to the page using the **Portlet Selector** portlet, as described in "Adding content to a shared page" on page 243.
3. Choose a portal layout, as described in "Modifying the layout of a shared page" on page 246.
4. Arrange the order and position of content on the selected layout using the **Layout Selector** portlet, as described in "Arranging content on the shared page" on page 248.
5. Display the new page right away by entering the shared page URL in your browser, as described in "Displaying a shared page" on page 251.

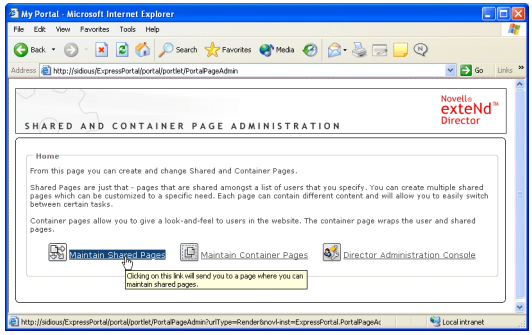
Shared pages are not tightly bound to portal layouts. That means portal administrators and shared page owners can switch layouts for their shared pages without losing any page contents. When a new layout is applied, any portlets that have been added to the page are automatically displayed using the new layout. You may need to fine-tune the content placement in the new layout.

Creating shared pages

Portal administrators can create shared pages from scratch or by copying existing pages. This section describes both procedures.

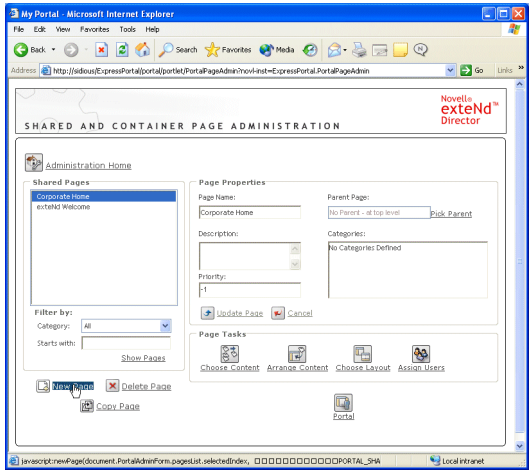
➢ To create a shared page:

1. Start the Portal Administration tool, as described in "Starting the Portal Administrator" on page 228.
2. Select **Maintain Shared Pages**.



The shared page section of the Portal Administration tool opens in your browser.

3. Select **New Page** at the bottom of the page:



An untitled, uncategorized shared page is created.

4. Enter a name for the new shared page in the **Page Name** field.

5 Enter other optional page properties as needed:

Property	What to specify
Description	Enter text that describes the page.
Parent Page	See "Shared page hierarchies" on page 59.
Categories	See Chapter 8, "Working with Page Categories".

6 Select **Update Page**.

➢ **To create a shared page by copying an existing page:**

1 Search for and select the page you want to copy in the list of shared pages.

2 Select **Copy Page** at the bottom of the page.

A new shared page is created with the name **Copy of *Original Page Name***.

3 Enter a name for the new shared page in the **Page Name** field.

4 Enter other optional page properties as needed:

Property	What to specify
Description	Enter text that describes the page.
Parent Page	See "Shared page hierarchies" on page 59.
Categories	See Chapter 8, "Working with Page Categories".

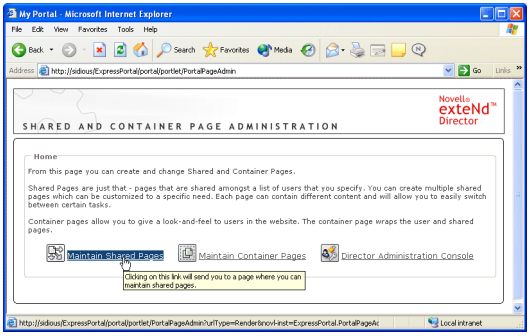
5 Select **Update Page**.

Adding content to a shared page

After creating a shared page, the next step is to add content by selecting portlets to place on the page. To add content to a new or existing shared page, you access the **Portlet Selector** portlet from the Portal Administration tool. You select from a list that includes prebuilt portlets supplied with exteNd Director along with any custom portlets you have created and registered.

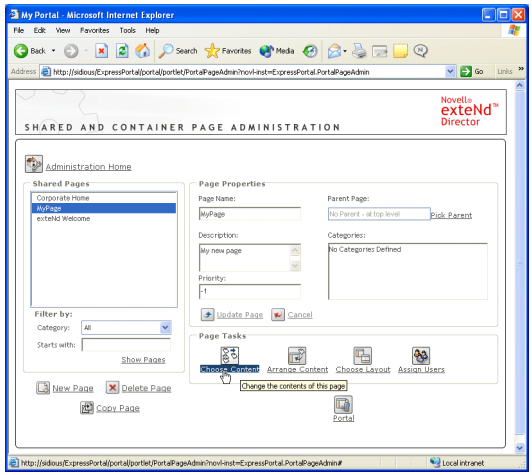
➢ **To add content to a shared page:**

1. Start the Portal Administration tool, as described in "Starting the Portal Administrator" on page 228.
2. Select **Maintain Shared Pages**.

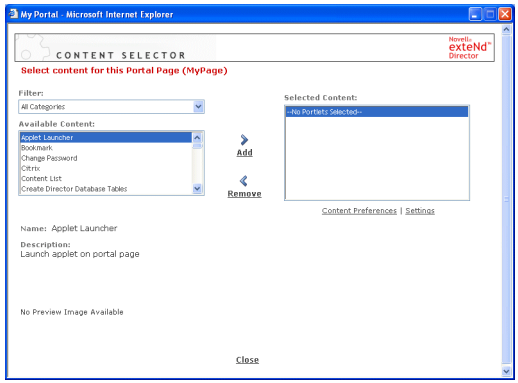


The container page section of the Portal Administration tool opens in your browser.

3 Select a page from the Shared Pages list and click **Choose Content**.



The Portlet Selector opens in a new browser window.



4 If you want to display a specific category of available portlets, choose a category from the **Filter** dropdown menu.

5 Select one or more portlets from the list of **Available Portlets**.

TIP: Hold down the **Control** key to select multiple non-contiguous portlets from the list; use the **Shift** key to make multiple contiguous selections.

6 Click **Add** to move your choices to the list of **Selected Portlets**.

NOTE: You can edit the preferences of one or more portlets that you have selected to be added to your shared page. The preference values you specify take effect for the instance of the portlet that appears on your page.

7 Click **Close**.

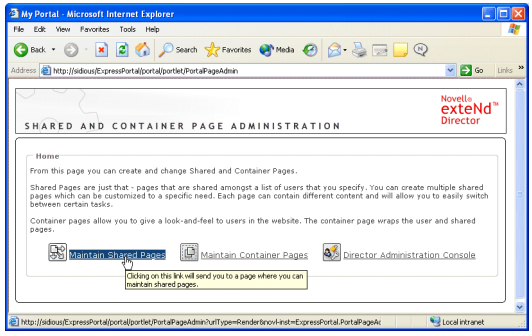
Now that you have chosen the content for your shared page, you can select a new layout as described in "Modifying the layout of a shared page" on page 246, or arrange the content on the current layout as described in "Arranging content on the shared page" on page 248.

Modifying the layout of a shared page

When you modify the layout of a shared page, the Portal Aggregator shifts existing content to accommodate the new layout. Although the Portal Aggregator makes its best guess on content placement, you may need to fine tune the end result.

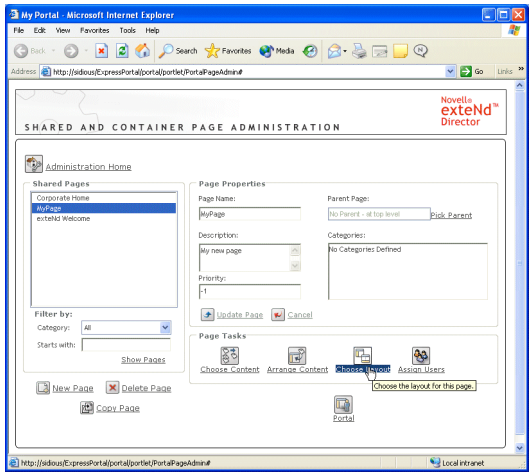
➢ **To modify the layout of a shared page:**

1. Start the Portal Administration tool, as described in "Starting the Portal Administrator" on page 228.
2. Select **Maintain Shared Pages**.

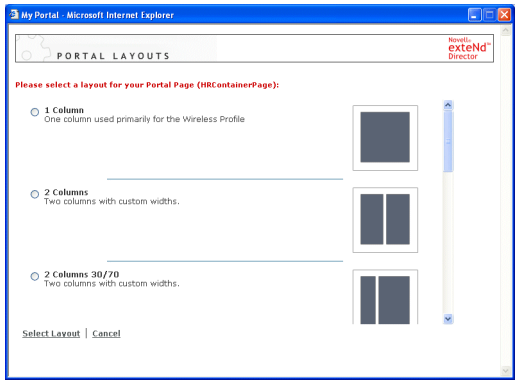


The shared page section of the Portal Administration tool opens in your browser.

3 Select a page from the Shared Pages list and click **Choose Layout**.



A layout selection page opens in a new browser window.



4 Scroll through the choices and select the layout of interest.

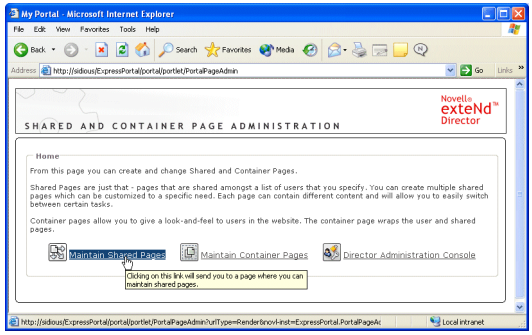
5 Click **Select Layout**.

Arranging content on the shared page

After you have designated the content and layout for your shared page, you can position the content in the selected layout, add other portlets in specific locations, or delete portlets.

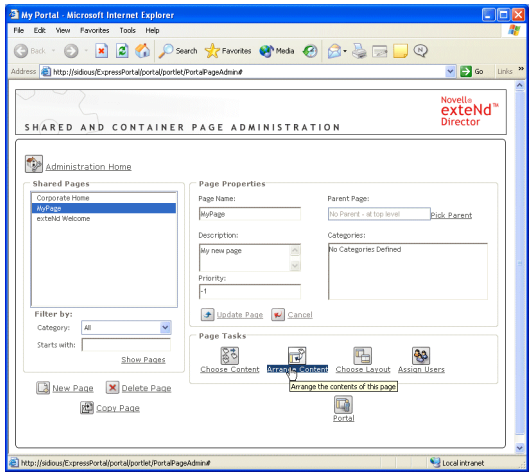
➢ **To arrange content on a shared page:**

1. Start the Portal Administration tool, as described in "Starting the Portal Administrator" on page 228.
2. Select **Maintain Shared Pages**.

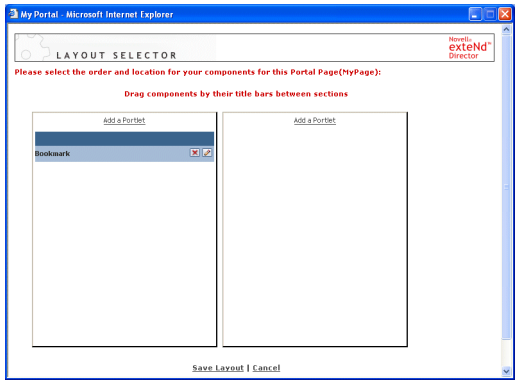


The shared page section of the Portal Administration tool opens in your browser.

3 Select a page from the Shared Pages list and click **Arrange Content**.



The Layout Selector portlet appears in a new browser window.



The portlets that you have already added appear in the page layout. By default, portlets are added in left-to-right, top-to-bottom order.

4 If you want to add a portlet to the page, follow these steps:

4a Click **Add a Portlet** in the desired layout frame.

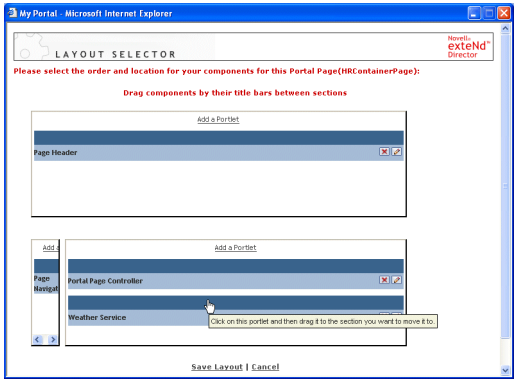
The Portlet Selector opens in a new browser window.

4b If you want to display a specific category of available portlets, choose a category from the **Filter** dropdown menu.

4c Double click the portlet of interest from the list of **Available Portlets**.

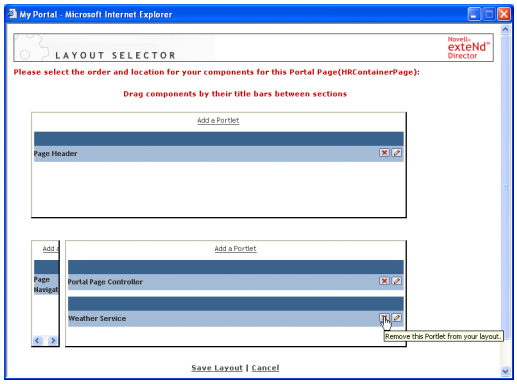
The Portlet Selector window closes and the portlet you selected appears in the target layout frame in the Layout Selector window.

5 If you want to move a portlet to a different location in the layout, follow these browser-specific steps:

For	Do this
Internet Explorer	**1** Move your cursor over the title bar of the portlet until the cursor changes to a hand shape.  **2** Hold down the left mouse button and drag the portlet to the desired location in the layout.
Netscape	**1** Click on the portlet you want to move. **2** Click inside the destination layout frame. The portlet moves to the destination.

6 If you want to remove a portlet from the layout, follow these steps:

6a Click the **X** in the upper right hand corner of the portlet:



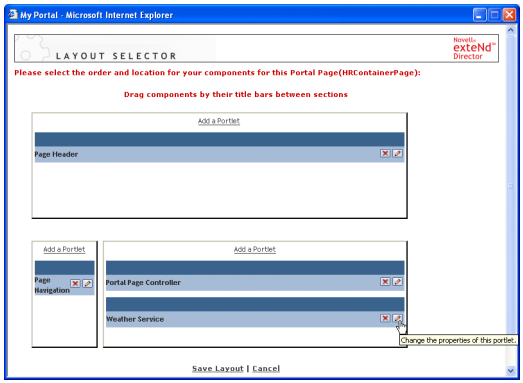
A message box appears, asking you to confirm the deletion.

6b Click **OK**.

The portlet is removed from the layout.

7 If you want to edit the preferences of a portlet, follow these steps:

7a Click the pencil icon in the upper right-hand corner of the portlet:



The **Portlet Preferences** tool opens in your browser.

7b Specify your preferences. The preference values you specify take effect for the instance of the portlet that appears on your page.

8 Click **Save Layout** to record your changes and close the Layout Selector.

Displaying a shared page

You can display your page by entering the shared page URL in your browser.

➢ **To display a shared page:**

- Enter the following URL in your browser:

  `http://server/project context/portal/pg/shared page name`

  For example, if your server is **localhost** and your project context is **MyWAR**, you can access a container page called **MySharedPage** by entering this URL in your browser:

  `http://localhost/MyWAR/portal/pg/MySharedPage`

Assigning pages to users and groups

By default, only portal administrators and locksmith users have permission to create, access, and modify container and shared pages. However, portal administrators can assign permission to other users and groups to work with specific container and shared pages. Two security levels of permission can be assigned:

Permission	Description	Can be assigned for
VIEW	Allows a user or group to access the page and see it in a list of available pages. Equivalent to READ plus LIST permissions.	**Container** and **shared** pages
OWNERSHIP	Allows a user or group to modify the content and layout of the page. Equivalent to WRITE plus PROTECT permissions.	**Shared** pages

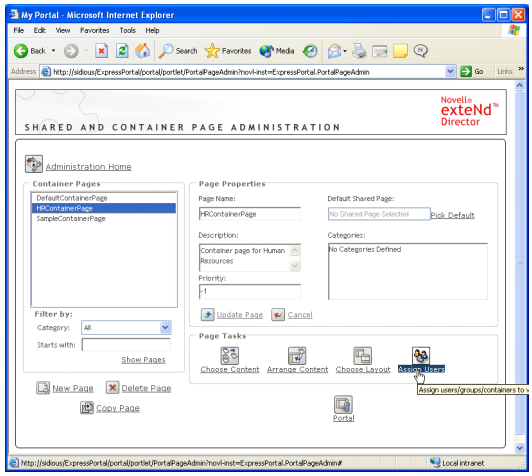
Assigning page VIEW permission

When you assign users VIEW permission for a container or shared page, they can access the page and see it in a list of available pages.

➢ **To assign VIEW permission for container or shared pages:**

1. Start the Portal Administration tool, as described in "Starting the Portal Administrator" on page 228.
2. Select **Maintain Container Pages** or **Maintain Shared Pages**.
3. Search for and select the container or shared page you want to assign.

4 Click **Assign Users**.



The **Page Permissions** page opens in a new browser window.

5 Select the **View** tab if it isn't already selected.

6 Enter the following information:

Field	What to specify
Search for	Select **Users** or **Groups** from the dropdown menu.
Starts with	Enter an optional text string to narrow the search results.

7 Click **Go**.

The results of your search appear in the **Results** panel.

8 Select the user or group you want to assign to the page and click the Add button:

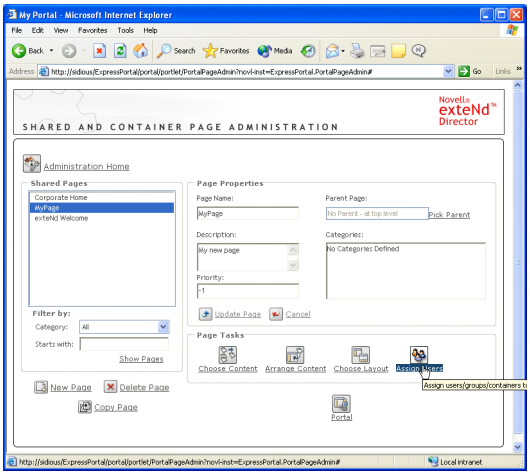
9 Click **Save**.

Assigning shared page owners

Users who own shared pages can modify the content of the pages they own and change the preferences of portlets on the page.

➢ **To assign OWNERSHIP permission for shared pages:**

1. Start the Portal Administration tool, as described in "Starting the Portal Administrator" on page 228.
2. Select **Maintain Shared Pages**.
3. Search for and select the shared page you want to assign.
4. Click **Assign Users**.



The **Page Permissions** page opens in a new browser window.

5. Select the **Ownership** tab if it isn't already selected.
6. Enter the following information:

Field	What to specify
Search for	Select **Users** or **Groups** from the dropdown menu.
Starts with	Enter an optional text string to narrow the search results.

7. Click **Go**.

The results of your search appear in the **Results** panel.

8 Select the user or group you want to assign as owner of the page and click the Add button:

9 Click **Save**.

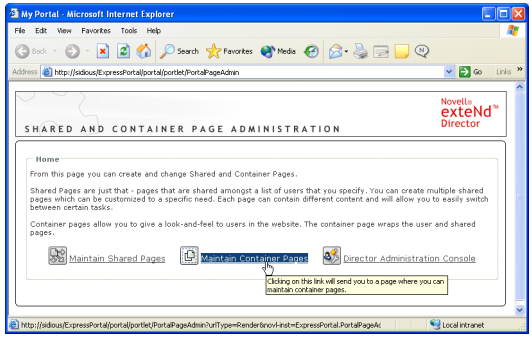
Choosing a default shared page for a container page

You can assign a default shared page to each container page you create. The Portal Aggregator considers this page assignment when aggregating content for a render request, as described in the section Determining the content page.

➢ **To choose a default shared page for a container page:**

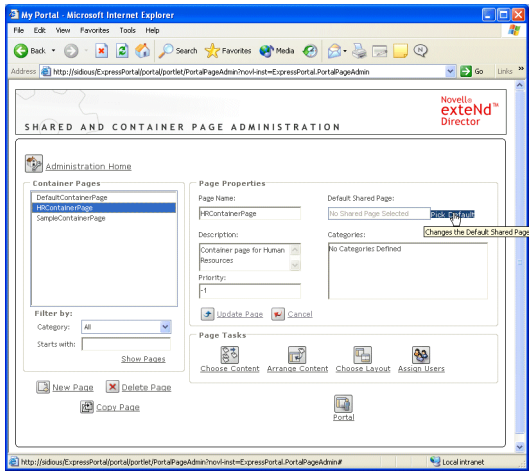
1 Start the Portal Administration tool, as described in "Starting the Portal Administrator" on page 228.

2 Select **Maintain Container Pages**.



The container page administrator opens in your browser.

3 Select a page in the list of container pages and click **Pick Default**.

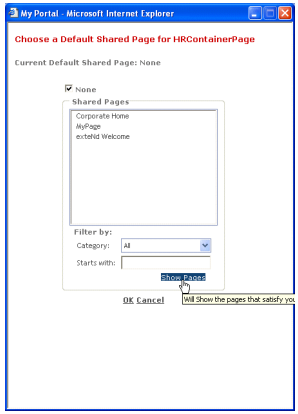


A selection page opens in another browser window.

4 Optionally enter search criteria in either or both **Filter By** fields:

Field	What to specify
Category	Select a category from the dropdown menu.
Starts with	Enter an optional text string to narrow your search results.

5 Select **Show Pages** to display a list of shared page you are authorized to view that meets your search criteria.



6 Select the shared page to use as the default for the container page and click **OK**.

The container selection page closes.

7 Back in the container page administrator, select **Update Page**.

16 Creating Custom Layouts

This chapter explains how to create custom layouts using the exteNd Director development environment. It contains the following sections:

About the Portal Layout and Portal Layout Definition Wizards

You create a new layout by creating a layout definition first and then a layout descriptor., as follows:

Use	To
Portal Layout Definition Wizard	Create a **layout definition file** for an XML layout that specifies: ◆ Sections within the layout ◆ Width of main page ◆ Width of sections ◆ How portlets flow within a particular section (left to right or top to bottom)

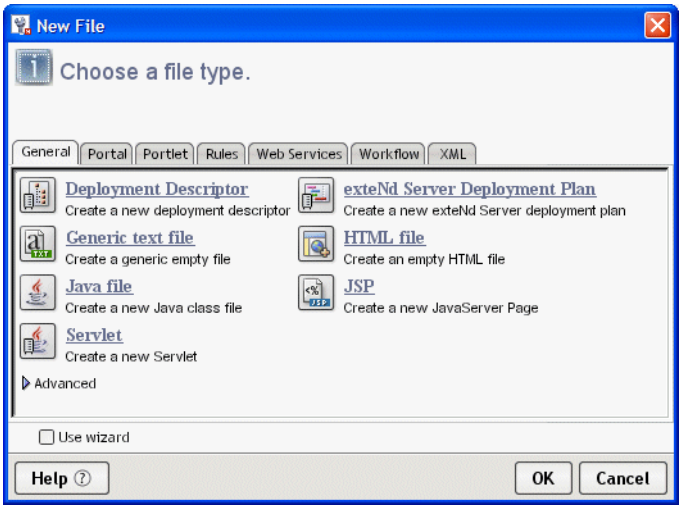
Use	To
Portal Layout Wizard	Create a **layout descriptor file** that provides: ◆ Display name ◆ Description ◆ Preview image file ◆ Reference to layout definition file

Creating an XML layout definition

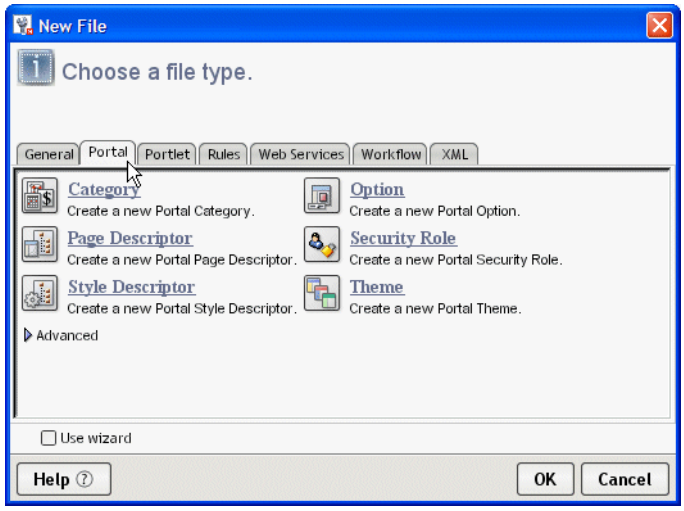
➢ **To create an XML layout definition:**

1. Start the exteNd Director development environment and open the project of interest.
2. Select **File>New>File**.

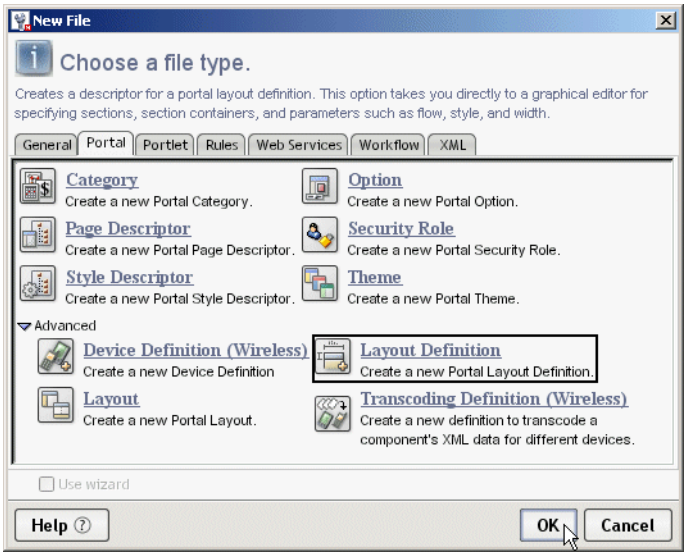
   The New File dialog appears:



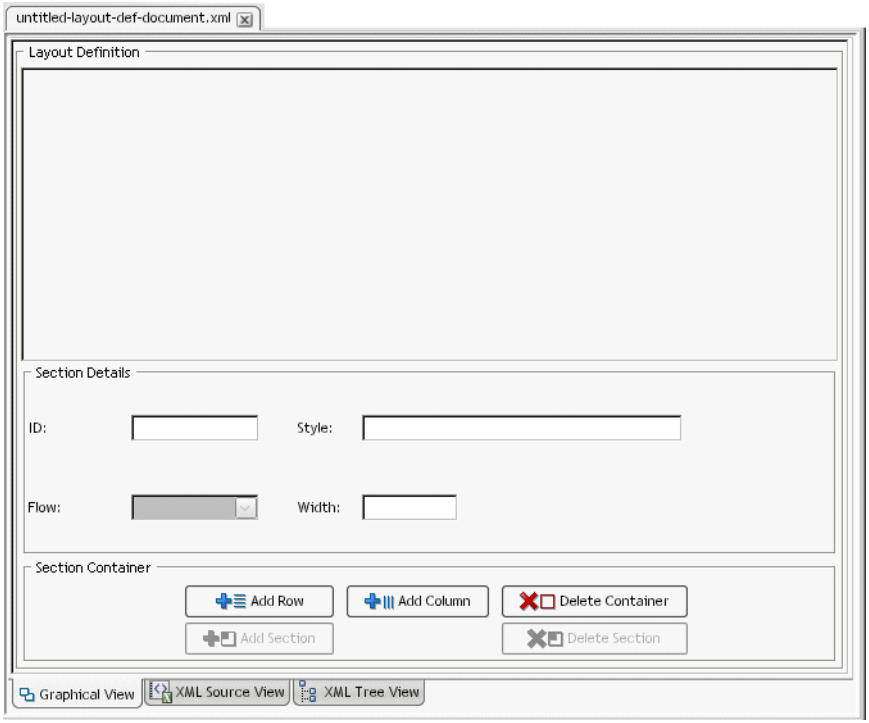
3 Click the **Portal** tab.



4 Click **Advanced**, then choose **Layout Definition** and click **OK**.

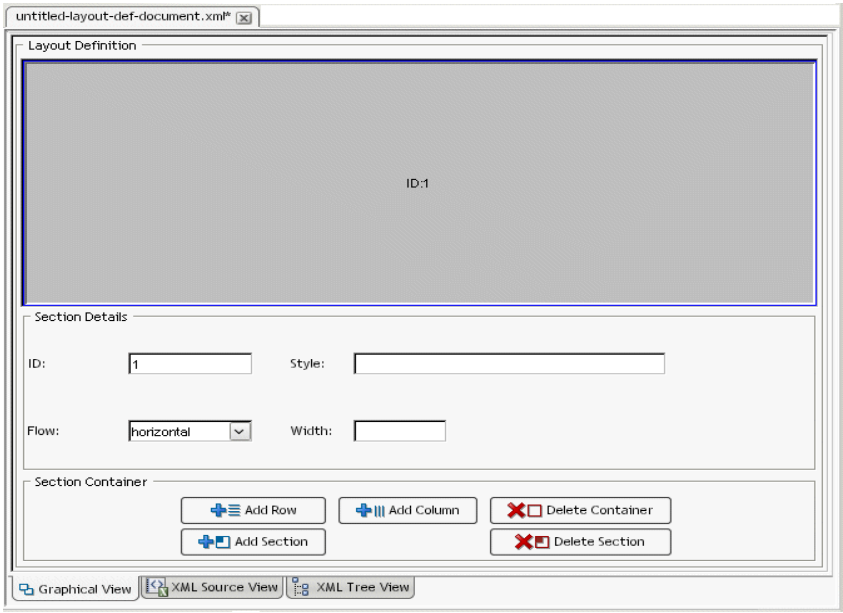


The layout definition editor displays.



5 To add a section container of type **row** and an initial section, click **Add Row**.

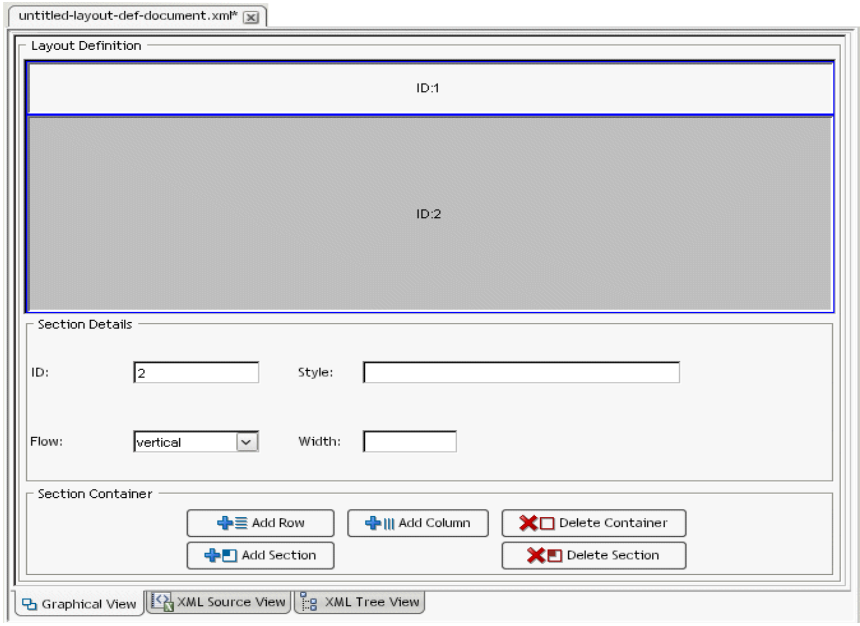
The layout definition editor creates the section container and adds a new section with an ID of 1.



TIP: Specify the flow, style, and width attributes for the new section container, as described in "Editing the attributes of a section or section container" on page 264.

6 To add a section container of type **column** and an initial section, click **Add Column**.

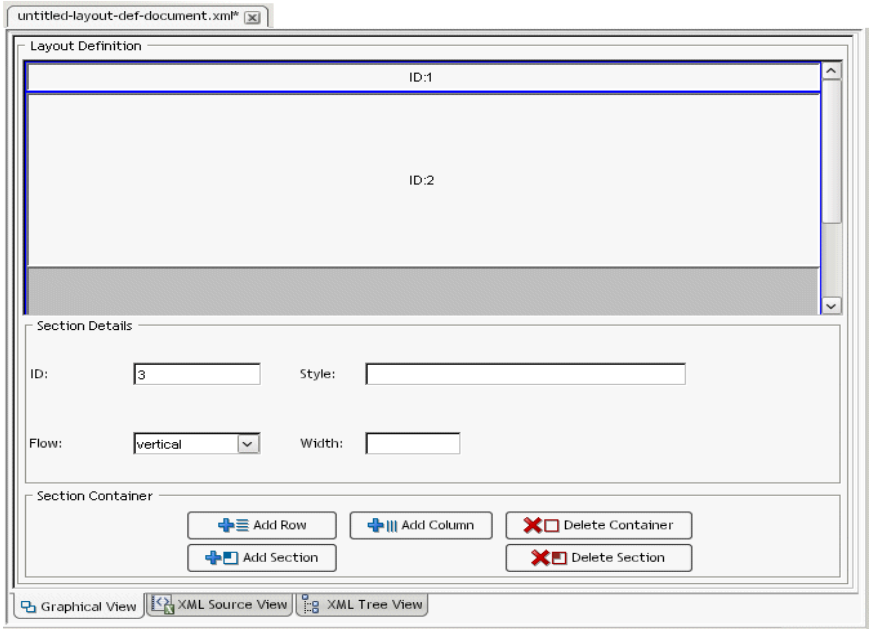
The layout definition editor creates the section container and adds a new section. The ID for the new section is 1 higher than the last section you added.



Specify the flow, style, and width attributes for the new section container, as described in "Editing the attributes of a section or section container" on page 264.

7 To add a section within a section container, select the target section container and click **Add Section**.

The layout definition editor adds a new section. The ID for the new section is 1 higher than the last section you added.



Specify the flow, style, and width attributes for the new section, as described in "Editing the attributes of a section or section container" on page 264.

8 To delete a section, select the section and click **Delete Section**.

9 To delete a section container, select the container and click **Delete Container**.

10 Select **File>Save**.

11 Specify a name for the layout definition file and click **Save**.

TIP: By convention, the layout definition file has the same name as the descriptor file, with **Def** (for definition) appended. For example, if the descriptor file is called HeaderContent.xml, the associated layout definition file might be called HeaderContentDef.xml. This naming convention is recommended, but not required.

The layout definition file is saved in the **portal-layout** directory within a resource set.

For more information about where files are located in a resource set, see the section on subdirectories for resources and Java classes in *Developing exteNd Director Applications*.

For details about the elements of the layout definition file, see "Layout definition file" on page 68.

Editing the attributes of a section or section container

➢ **To edit the layout attributes:**

1. In the layout definition editor, select the section or section container you want to edit.
2. Select one of the following **Flow** attributes:

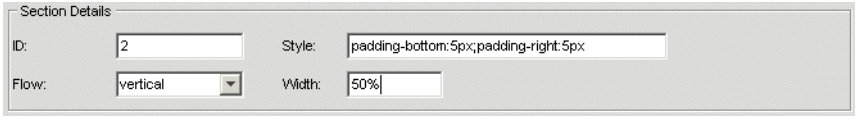
Flow attribute	Description
horizontal	Indicates that portlets should flow from left to right within the section or section container
vertical	Indicates that portlets should flow from top to bottom within the section or section container.

3. (Optional) Specify a **Style** setting. When the section is translated into HTML, the style setting is applied to the style attribute for the generated table row or column.

   You can specify any style for the TD cell, including:

Style	Values
padding	◆ padding-bottom ◆ padding-top ◆ padding-right ◆ padding-left
alignment	◆ vertical-align ◆ horizontal-align

4. (Optional) Specify a **Width** setting. When the section is translated into HTML, the width setting is applied to the width attribute for the generated table row or column. The width can be expressed as a percentage or in pixels.



Here is an example of section container definitions:

```
<section-container type="row">
<s3-section flow="horizontal" id="1" style="padding-right:5px;padding-bottom:5px;vertical-
align:top" width="100%"/>
</section-container>
<section-container type="row">
<s3-section flow="vertical" id="2" style="padding-bottom:5px;padding-right:5px"
width="25%"/>
```

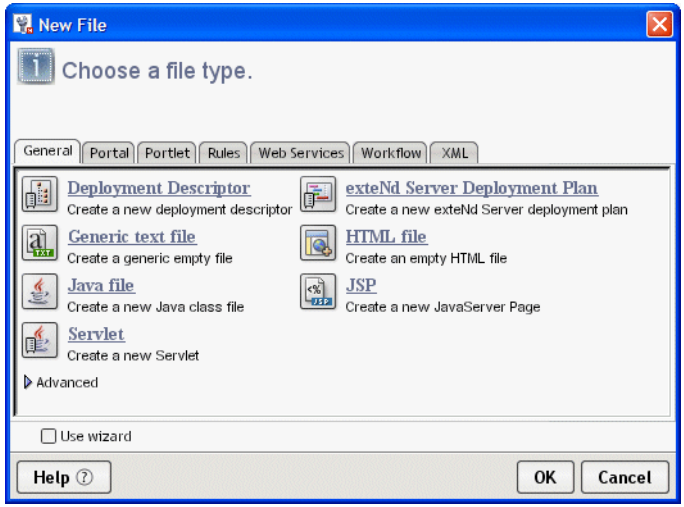
```
<s3-section flow="vertical" id="3" style="padding-bottom:5px;padding-right:5px"
width="50%"/>
<s3-section flow="vertical" id="4" style="padding-bottom:5px;padding-right:5px"
width="25%"/>
</section-container>
```

Creating a layout descriptor

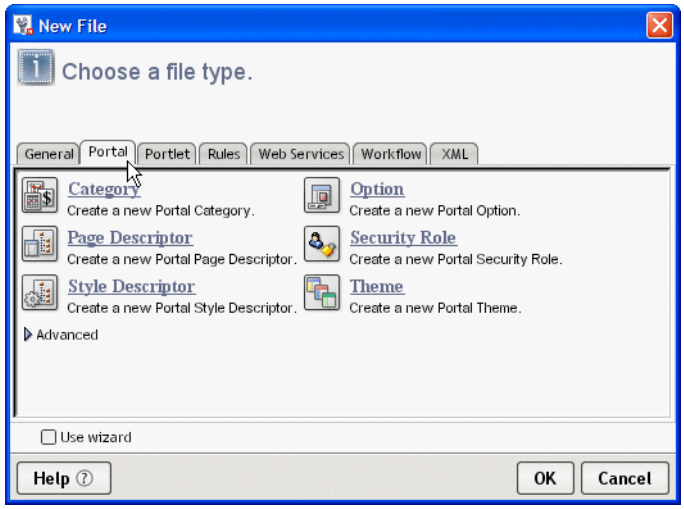
➢ **To create a layout descriptor:**

1. Start the exteNd Director development environment and open the project of interest.
2. Select **File>New>File**.

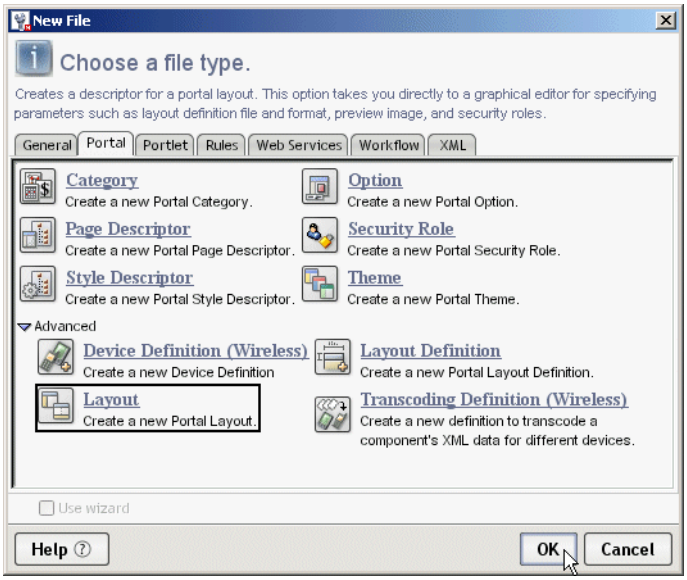
   The New File dialog appears:



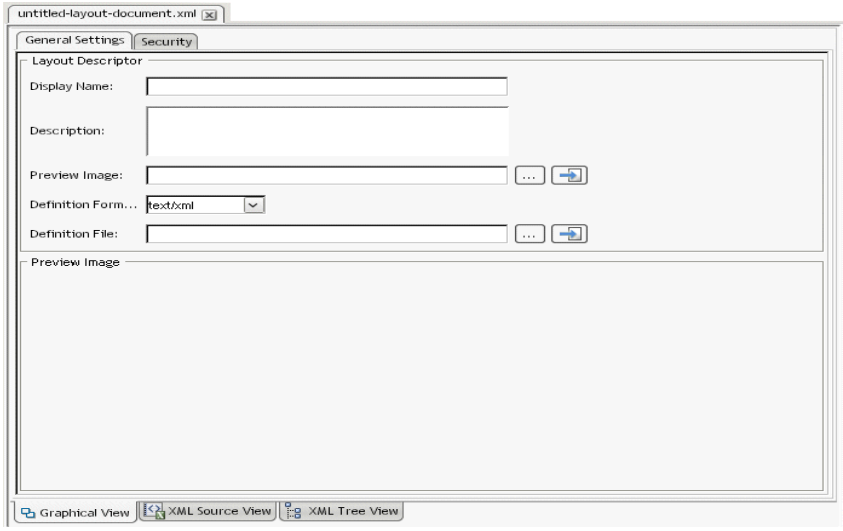
3 Click the **Portal** tab.



4 Click **Advanced**, then choose **Layout** and click **OK**.



The layout descriptor editor displays.



5 In the layout descriptor editor, click the **General Settings** tab to specify the general settings for the layout descriptor file, as follows:

General setting	What to specify
Display Name	Enter a name for the layout. **NOTE:** This name will identify the layout in the user interface of a portal application.
Description (optional)	Enter a description for the layout.
Preview Image (optional)	Enter the path to a preview image.The path can be a fully qualified URL or an URL that contains portal replacement strings, as described in Chapter 23, "Portal Replacement Strings". To search for the image file, click the ellipsis button:  To open the image file, click the arrow button: 
Definition Format	Select a layout definition format from the dropdown menu.
Definition File	Enter the path to the layout definition file. To search for the layout definition file, click the ellipsis button:  To open the layout definition file, click the arrow button: 

For details about these elements of the layout descriptor file, see "Layout descriptor file" on page 67.

6 In the layout descriptor editor, click the **Security** tab to specify the security settings for the layout descriptor file, as follows:

Security setting	What to specify
List Roles (optional)	Select a list role from the Available List Roles list and click the right arrow button. Repeat this procedure for each list role you want to select.
Run Roles (optional)	Select a run role from the Available Run Roles list and click the right arrow button. Repeat this procedure for each run role you want to select. **IMPORTANT:** If you specify a run-role-map for a layout, you should select the equivalent list role.

📖 For details about security settings in the layout descriptor file, see "Layout descriptor file" on page 67.

7 Select **File>Save**.

The layout descriptor file is saved in the **portal-layout** directory within a resource set. The ID for the layout is the name of its descriptor file, without the XML extension.

📖 For more information about where files are located in a resource set, see the section on subdirectories for resources and Java classes in *Developing exteNd Director Applications*.

17 Creating Custom Themes

This chapter explains how to create custom themes using the exteNd Director development environment. It contains the following sections:

About the Portal Themes Wizard

You create a new portal theme by using the **Portal Themes Wizard**, which is available in the exteNd Director development environment.

The Portal Themes Wizard creates a theme descriptor file, which includes the following information about the theme:

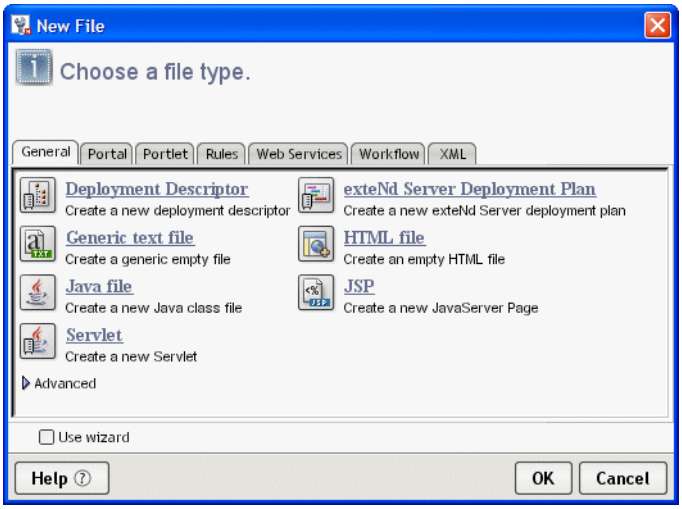
- Display name
- Description
- Preview image
- Thumbnail image

Creating a portal theme

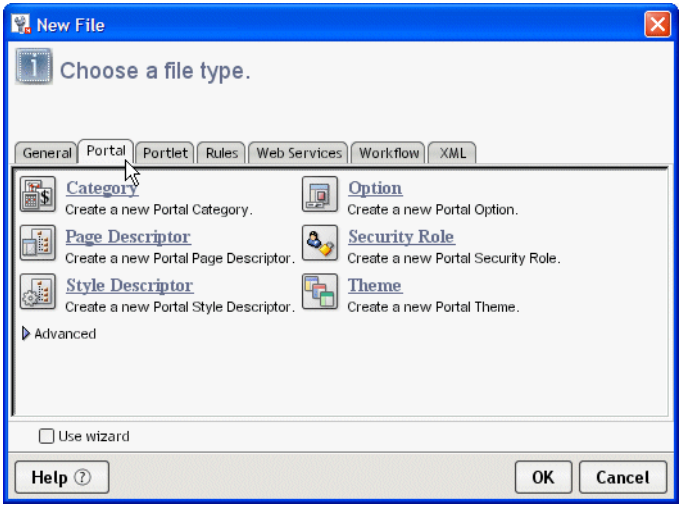
➢ **To create a portal theme:**

1. Start exteNd Director and open the project of interest.
2. Select **File>New>File**.

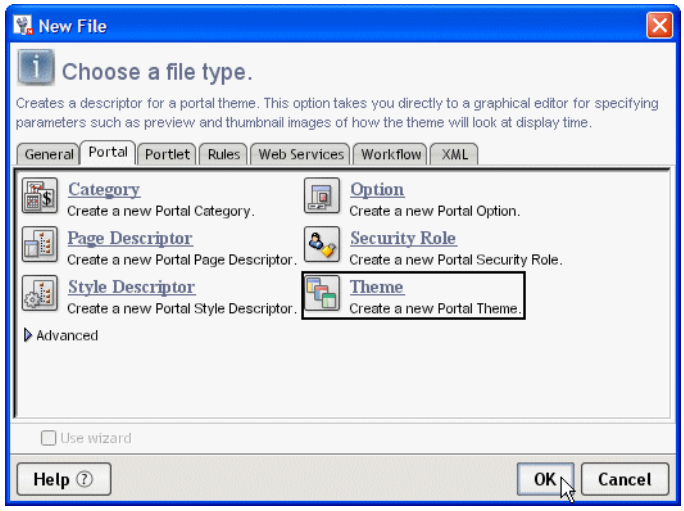
   The New File dialog appears:



3. Click the **Portal** tab.



4 Choose **Theme** and click **OK**.



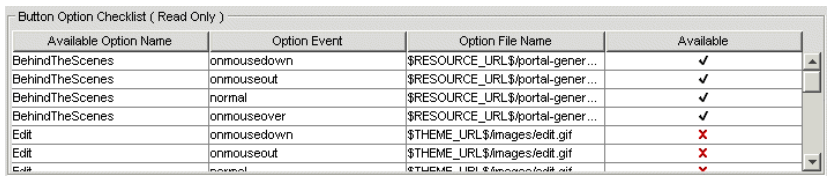
The theme descriptor editor displays.

5 In the theme descriptor editor, specify the settings for the theme descriptor file.

For details about the elements of the theme descriptor file, see "Theme descriptor file" on page 77.

6 In the **Button Option Checklist (Read Only)** box, review the list of portal options in the current resource set to be sure the theme has provided the necessary images.

Whenever an image file has been provided for an option in the current theme, the **Available** column shows a check mark. Whenever an image file is not available in the current theme, the **Available** column shows an X.



Each theme can supply a unique set of images for the various events associated with a portal option. However, the file names for these images should be the same in each portal theme. By using a consistent naming convention across themes, you can ensure that each user event will have an image to display, regardless of which theme has been selected.

7 Select **File>Save**.

The theme descriptor file must have the name **theme.xml**. Each theme used by a portal application must have a separate theme.xml file.

exteNd Director saves the theme descriptor file in a **theme folder**, which is a subdirectory of the **portal-theme** directory within a resource set. The name of the theme folder matches the display name of the theme and provides a key for the theme and is used to uniquely identify the theme.

📖 For more information about where files are located in a resource set, see the section on subdirectories for resources and Java classes in *Developing exteNd Director Applications*.

Creating a theme CSS file

➢ **To create a theme CSS file:**

1 Start exteNd Director and open the project of interest.

2 Open an existing theme CSS file and save it to the theme directory for your custom theme.

3 Edit the file to suit the requirements of your theme.

You can define your own styles by defining new classes and editing the settings for the standard exteNd Director classes (those classes that contain the **nv** label).

IMPORTANT: Do **not** remove or rename any of the standard classes. These classes are required by many of the core portlets that ship with exteNd Director.

If you decide to create your own custom classes, you should use these classes consistently throughout every theme in your application. In other words, each theme should provide appropriate settings for each of these custom classes, just as the installed themes provide settings for each of the standard exteNd Director classes.

4 Select **File>Save**.

The theme CSS file must be saved in the theme folder. The theme folder is a subdirectory of the **portal-theme** directory within a resource set.

For example, if your new theme is called GreenAndGold, follow these steps:

4a Create a subdirectory called **GreenAndGold** in the **portal-theme** directory.

4b Save the theme.css file for the new theme in **portal-theme/GreenAndGold**.

📖 For more information about where files are located in a resource set, see the section on subdirectories for resources and Java classes in *Developing exteNd Director Applications*.

18 Creating Custom Options

This chapter explains how to create custom options using the exteNd Director development environment. It contains the following sections:

About the Portal Option wizard

You create a new portal option by using the **Portal Option Wizard**, which is available in the exteNd Director development environment.

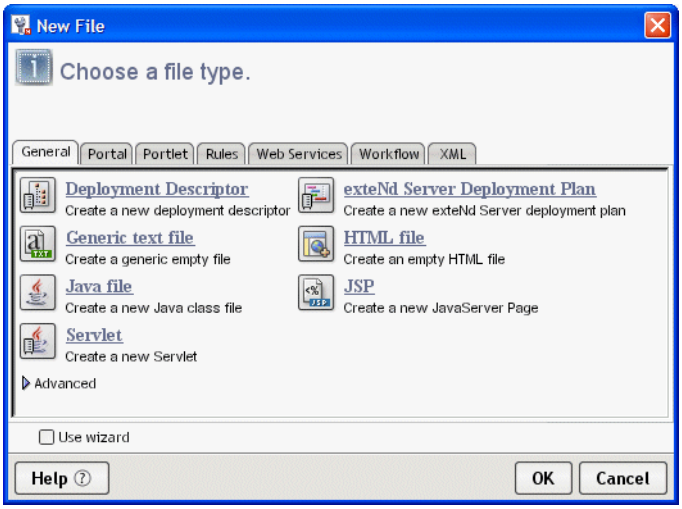
The Portal Option Wizard creates an option descriptor file. This file provides several elements that describe the option, including a display name and description, as well as a link and a set of images for the option.

Creating a portal option file

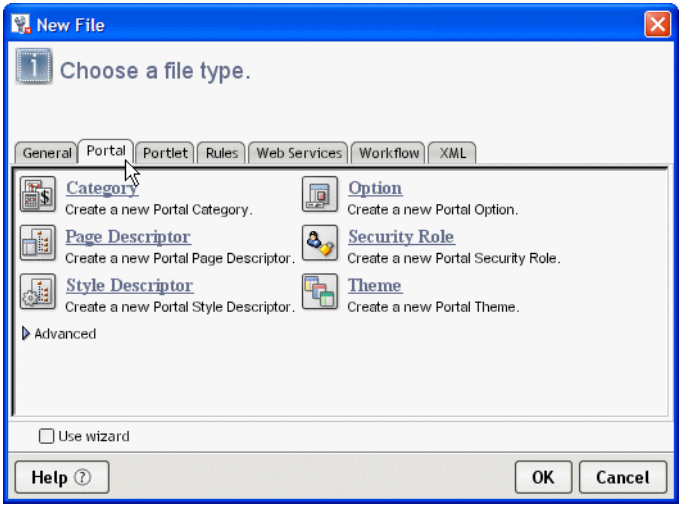
➢ **To create a portal option file:**

1. Start exteNd Director and open the project of interest.
2. Select **File>New>File**.

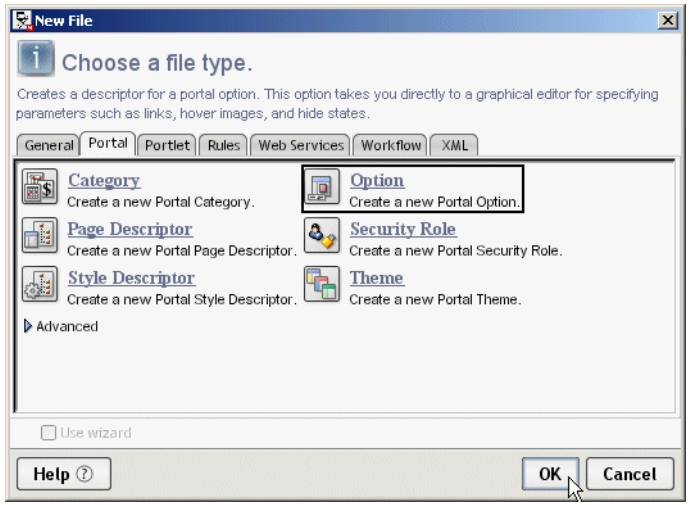
   The New File dialog displays:



3. Click the **Portal** tab.



4 Choose **Option** and click **OK**.

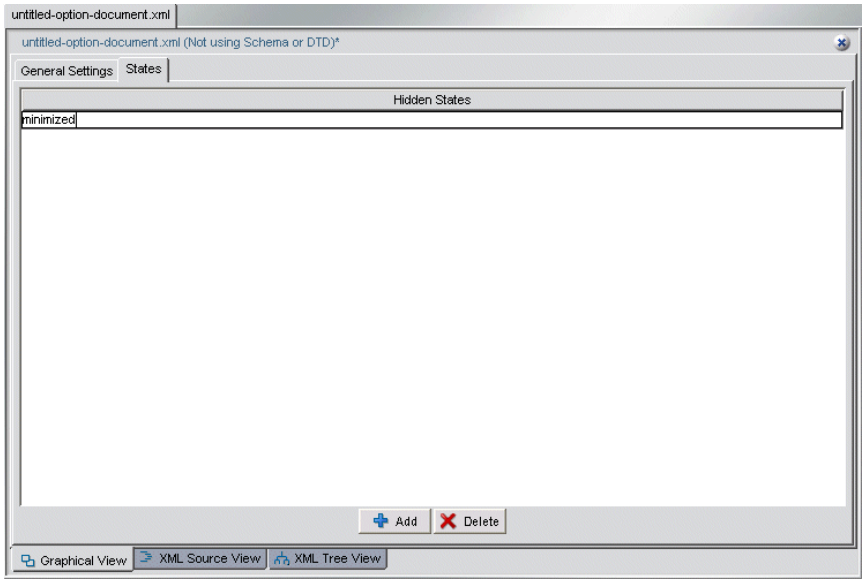


The option descriptor editor displays.

5 In the option descriptor editor, specify the general settings for the option descriptor file.

📖 For details about the elements of the option descriptor file, see "Portal option descriptor file" on page 106.

6 On the **States** tab, define one or more hidden states for the portal option. To add a new state, click **Add**. Then double-click **NEW_STATE** and type the name for the state.



Hidden states are states in which an option should not be displayed. For example, when a portlet is minimized, you would typically want to hide the minimize option on the title bar.

You can specify any of the predefined portal states (**normal**, **minimized**, or **maximized**) as hide-states. If you are using the built-in action handler (EboActionHandler), these are the only states available. If you write your own custom action handler, you can provide support for additional states.

7 Select **File>Save**.

exteNd Director saves the option descriptor file in the **portal-option** directory within a resource set. The ID for the option is the name of its descriptor file, without the XML extension.

 For more information about where files are located in a resource set, see the section on subdirectories for resources and Java classes in *Developing exteNd Director Applications*.

19 Developing Portlets

This chapter describes methods for developing and running custom portlets in exteNd Director. It includes the following topics:

Creating a portlet class

This section describes several ways to create a portlet class to run with the exteNd Director portal. If you use exteNd Director portlet development tools, such as pageflow design tools or the Portlet Wizard, the following operations are performed for you automatically:

- Portlet deployment descriptors are updated
- A portlet definition is created and registered with the exteNd Director portal
- Preferences are set to the default values defined in the portlet class
- Your portlet is dynamically loaded to the server, so you can test the logic without needing to redeploy your entire application

Using pageflow design tools

exteNd Director provides design tools for developing a specialized type of portlet called a pageflow, without the need for Java coding.

For an in-depth discussion about pageflows and how to use these design tools, see the *Pageflow and Form Guide*.

Using the Portlet Wizard

The Portlet Wizard allows you to create Java Portlet 1.0-compliant portlets that can take advantage of exteNd Director extensions to Java Portlet 1.0.

Creating a portlet with the Portlet Wizard

➢ **To create a custom portlet using the Portlet Wizard:**

1. In exteNd Director, select **File>New>File**.

   The New File dialog opens.

2. Select the **Portlet** tab, choose **Portlet**, then click **OK**.

   The first panel of the Create a New Portal Portlet Wizard opens.

3. Enter the following information:

Field	What to specify
Class name	Enter the name you want to use for the portlet class.
Package	Enter a package name for the portlet. If the package does not exist, the wizard creates it in the root of the resource JAR in the target resource set. If you do not specify a package, the wizard puts the portlet class in the root of the resource JAR within the target resource set.
Resource Set	Select the resource set where the portlet fragment deployment descriptor should be stored. Resource sets provide dynamic deployment of design-time changes. 📖 For more information about resource sets, see the chapter on using the resource set in an exteNd Director application.
Include logging code	Check or uncheck this box to indicate whether you want logging code added to your portlet. When you check this box, the wizard automatically adds code that lets you send messages to the log when the logging level is set to the trace level.

Field	What to specify
Content types	Check the content types supported by the portlet. Portlets can support multiple content types.
Data definition (for portlets using the transcoding engine)	Specify a data definition file for the portlet. The data definition file for a portlet determines how the portlet will transcode data for wireless support.

4 Click **Next**.

The second panel of the wizard opens.

5 Enter the following information:

Field	What to specify
Description	Enter a text description of the portlet.
Display Name	Enter a name for the portlet. **NOTE:** The display name is not used by the exteNd Director portal.
Title	Enter a title for your portlet. The title is used for displaying the portlet in lists and on decorators.
Short Title	Enter a short title for the portlet. **NOTE:** The short title is not used by the exteNd Director portal.
Expiration Cache	Enter the time in seconds after which the portlet's cached content expires. To disable caching for the portlet, enter **0**. To specify that the portlet's cached content never expire, enter **-1**.
Modes	Check the modes that the portlet supports. **NOTE:** The new portlet automatically supports View mode, in compliance with Java Portlet 1.0. For more information, see the section on portlet modes.

6 Click **Finish**.

A popup window opens, indicating when the Portlet Wizard has finished creating the portlet.

7 Click **OK** to dismiss the popup window.

The Wizard builds the portlet class and displays Java source code that meets the minimum requirements for a Java Portlet 1.0-compliant portlet. The wizard also creates the necessary files to support dynamic loading, as described in "What the Portlet Wizard generates" on page 282.

8 Add your custom code to the portlet.

9 Compile the portlet.

10 Test your changes, as described in

exteNd Director performs the following actions:

10a Builds the portlet class

10b Displays the Java source file for the portlet

What the Portlet Wizard generates

The Portlet Wizard performs the following actions for each portlet you create:

1 Builds the portlet class file and stores it in the resource set you specified, based on what you entered for **Package Name**.

2 Creates a portlet fragment deployment descriptor in the **portal-portlet** directory of the target resource set. This is an XML file that describes the portlet, based on the preferences you specified in the Portlet Wizard.

The name of the descriptor file matches the name of the portlet class. Following is an example of a simple portlet fragment deployment descriptor whose name is CreateNewUser.xml:

```
<portlet xmlns="http://www.novell.com/xml/ns/portlet-fragment"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <description>Creates new user</description>
    <portlet-name>CreateNewUser</portlet-name>
    <display-name>Create New User</display-name>
    <portlet-class>com.novell.portlets.CreateNewUser</portlet-
class>
    <expiration-cache>30</expiration-cache>
    <supports>
        <mime-type>text/html</mime-type>
        <portlet-mode>edit</portlet-mode>
    </supports>
    <portlet-info>
        <title>Create New User</title>
        <short-title>Create User</short-title>
    </portlet-info>
    <supported-option>edit</supported-option>
    <auto-register enabled="true"/>
</portlet>
```

NOTE: With the portlet class and portlet fragment deployment descriptor stored in the resource set, exteNd Director dynamically loads any changes you make to the portlet during development. As a result, the changes are immediately reflected in the runtime environment on the server, allowing you to test them without having to redeploy the entire application.

 For more information, see the sections on support for dynamic loading of portlets and on portlet fragment deployment descriptors.

3 Sets the portlet to be registered automatically at deployment in the portlet fragment deployment descriptor, as follows:

```
<auto-register enabled="true"/>
```

If you don't want the portlet to be registered automatically, change the setting to:

```
<auto-register enabled="false"/>
```

Importing Java Portlet 1.0-compliant portlets from other sources

If you do not want to use exteNd Director portlet development tools, you can import portlets from other sources into the resource set of an exteNd Director portal application WAR—as long as the following conditions are met:

- Portlets must be Java Portlet 1.0-compliant
- Portlets must be packaged in a JAR that contains:
  - Portlet class files
  - An empty portlet.xml deployment descriptor file
  - One portlet fragment deployment descriptor for each portlet definition

     For more information, see the section on portlet fragment deployment descriptors.

➢ **To import portlets into an exteNd Director portal application WAR:**

1 Package the portlets as described above.

2 Follow the procedure described in the section on importing resources into a view.

Creating a portlet definition

exteNd Director pageflow design tools and the Portlet Wizard automatically create and register one portlet definition that inherits all properties from the portlet class you create. However, in some instances, portal administrators may want to create additional definitions—for example, if a definition is needed that inherits all the characteristics of the portlet class, but uses different initialization parameters.

You create a new portlet definition by creating a new portlet descriptor—either as a new portlet fragment deployment descriptor or new <portlet> sections in portlet.xml and novell-portlet.xml. This section describes both approaches.

➢ **To create a new portlet definition as a portlet fragment deployment descriptor:**

1. Create a new portlet fragment deployment descriptor by copying the one that was created for your portlet class.
2. Change **<portlet-name>** to a new, unique name.
3. Change other properties and add new properties as desired.
4. Keep the reference to the original portlet class in **<portlet-class>**.
5. Save the descriptor file in the **<portal-portlet>** section of your application's resource set with the same name as <portlet-name>.
6. Register the new portlet definition as described in "Registering a portlet definition" on page 285.

📖 For more information, see the section on portlet fragment deployment descriptors.

➢ **To create a new portlet definition by modifying portlet.xml:**

1. With your application open in the Director Designer, open **portlex.xml**.
2. Find the **<portlet>** section that describes the portlet of interest.
3. Copy and paste a new <portlet> section after the original one.
4. In the new <portlet> section:
   - **4a** Change **<portlet-name>** to a new, unique name.
   - **4b** Change other properties and add new properties as desired.
5. Keep the reference to the original portlet class in **<portlet-class>**.
6. Save portlet.xml.
7. If you want to specify exteNd Director value-added properties for the new definition, open novell-portlet.xml and repeat Step 2 through Step 3.
8. Save novell-portlet.xml.
9. Rebuild and redeploy the application.
10. Register the new portlet definition as described in "Registering a portlet definition" on page 285.

Registering a portlet definition

Portal administrators register portlet definitions using the Portlet Management section of the Director Administration Console (DAC). A registered portlet definition is called a portlet registration. At registration time, administrators have the opportunity to change preferences and settings for the portlet registration. See the section on registering a portlet definition.

Testing a portlet

After completing the registration process, you can unit test portlet registrations by running them directly in a browser.

➢ **To run a portlet directly in a browser:**

- Enter the portlet path URL, using this syntax:

  `http://`*`host`*`/`*`context`*`/portal/portlet/`*`name of portlet registration`*

  Example:

  `http://localhost/ExpressPortal/portal/portlet/WelcomeMessage Portlet`

  📖 For more information about the portlet path URL, and other request URLs recognized by the exteNd Director portal, see the section on application requests.

Adding portlets to portal pages

After unit testing your portlet, you can add it to one or more portal pages. exteNd Director supports container, shared, and personal pages, as described in the chapter on working with portal pages.

All users can add portlets to their own personal pages, while only authorized users can add portlets to container and shared pages.

📖 To learn how to use portlets with portal pages, see the following references:

- Creating and maintaining container pages
- Creating and maintaining shared pages
- Creating personal pages

20 Using the Portal Management Section of the DAC

This chapter describes how to use the Portal Management section of the exteNd Director Administration Console (DAC). It has these sections:

About the Portal Management section of the DAC

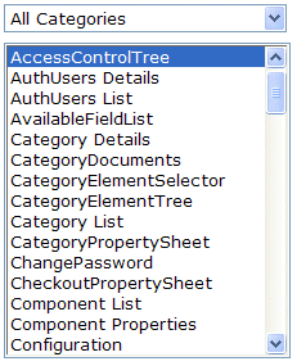
The Portal Management section of the exteNd Director Administration Console (DAC) allows portal administrators to view information about the Portal subsystem of a deployed exteNd Director application. It is read-only.

Using dropdown lists

Each page in the Portal Management section provides one or two dropdown list controls to that you can use to filter the list of items displayed.

Upper

The upper dropdown list allows you display items belonging to each WAR in your project

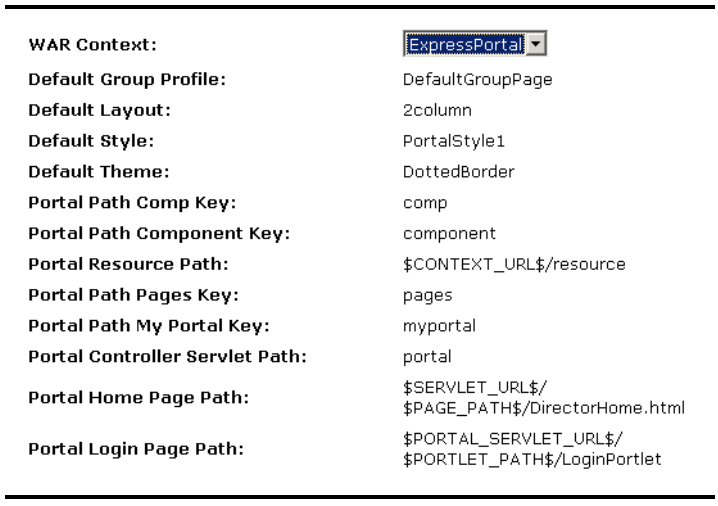


Lower

The lower dropdown list varies from page to page and is explained in the following sections

General

The General section allows you to view servlet definitions and other properties for each WAR, as defined in its web.xml file. The control at the top of the page selects the WAR to examine.



Some of the properties include Portal replacement strings such as $PORTLET_PATH$. These are described in Chapter 23, "Portal Replacement Strings".

Components

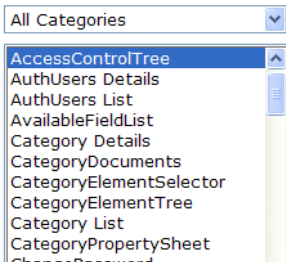
Portal components are deprecated exteNd Director provides runtime support for portal components created in earlier versions of the product. However, you should use portlet technology for all new development.

Subpages

When you click any component in the list, you can choose from four subpages of information:

- "Components: General" on page 290
- "Components: Defaults" on page 290
- "Components: Styles" on page 291
- "Components: Options" on page 291

Lower dropdown list



The lower dropdown list allows you to filter components by category

Components: General

The General section of the Components panel displays key configuration parameters from the component descriptor.

Name:	DocumentList
Component display name:	DocumentList
Class name:	com.sssw.portal.pmc.EboDocumentList
Component description:	PMC - The document list component
Preview Image:	
Default Style:	none
Category:	PmcComponent
Show title bar:	false
Run Roles:	none
List Roles:	Restricted

Components: Defaults

The Components Defaults section displays the default values of each item defined in the component descriptor.

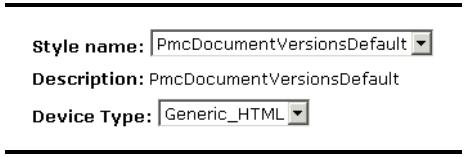
Default Name: debug

Default value:

false

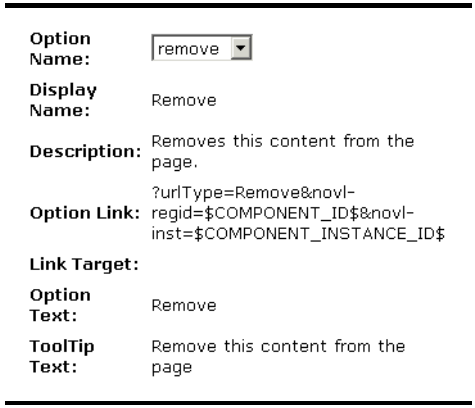
Components: Styles

This section displays the list of styles for a component.



Components: Options

This section displays the list of options for a component.



Pages

The Pages section gives you the ability to view the page source.

📖 For more information about portal PID pages, see Chapter 9, "Working with PID Pages".

Lower dropdown list

The lower dropdown list allows you to filter pages by category.

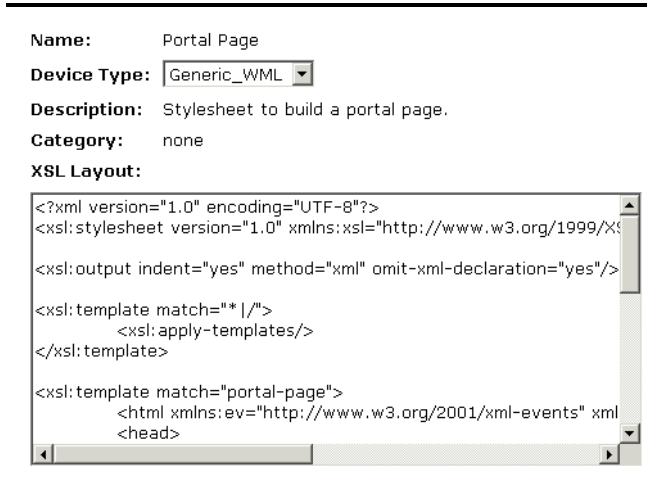
Styles

The styles section displays a list of portal-wide style sheets and allows you to view the code.

For more information about style sheets, see "Styling portlets that generate XML content" on page 202.

Lower dropdown list

The lower dropdown list allows you to filter styles by category.

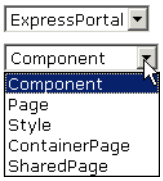


Categories

This section displays a list of *categories*. A category is simply a label you define for the purpose of grouping and managing a set of portlets or a pages.

For more information about categories, see Chapter 8, "Working with Page Categories"

Lower dropdown list



In Categories, the lower dropdown list allows you to list the portlets, pages, styles, container pages, and shared pages that belong to a category

Layouts

This section displays a list of *layouts*. A layout is a template that defines how a set of selected portlets should appear on a page.

 For more information about layouts, see Chapter 4, "Working with Portal Layouts".

Name: HeaderNavContentFooter

Description: Header, Navigation bar, content area, and Footer

Preview Image: $RESOURCE_URL$/portal-layout/images/HeaderNavContentFooter.gif

Definition Format text/xml

Definition ID: HeaderNavContentFooterDef.xml

Section IDs: 1, 2, 3, 4

Layout Definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<portal-layout-def>
<section-container type="row">
<s3-section flow="horizontal" id="1" style="padding-right:5px;paddin
</section-container>
<section-container type="row">
<s3-section flow="vertical" id="2" style="padding-bottom:5px;paddin
<s3-section flow="vertical" id="3" style="padding-bottom:5px;paddin
</section-container>
<section-container type="row">
<s3-section flow="horizontal" id="4" style="padding-right:5px;paddin
</section-container>
```

Run Roles: none

List Roles: none

Themes

This section displays a list of *themes*. A theme is a set of visual characteristics that apply to an entire exteNd Director portal application.

For more information about themes, see Chapter 5, "Working with Portal Themes".

Name:	BasicBlue
Display Name:	Basic Blue
Description:	Basic Blue theme, smooth plastic
Preview Image:	$RESOURCE_URL$/portal-theme/BasicBlue/images/preview.gif
Thumbnail Image:	$RESOURCE_URL$/portal-theme/BasicBlue/images/thumbnail.gif

Options

This section displays a list of *options*. An option is an image or text string that appears in the title bar of a portlet.

For more information about options, see Chapter 7, "Working with Portal Options".

Name:	edit
Display Name:	Edit
Description:	Displays a screen to edit the preferences
Option Link:	?urlType=Render&wsrp-mode=edit&wsrp-windowstate=normal&novl-regid=$COMPONENT_ID$&novl-inst=$COMPONENT_INSTANCE_ID$
Link Target:	
Option Text:	Edit
ToolTip Text:	Edit the preferences of this content

21 Using the Portlet Management Section of the DAC

This chapter describes how to use the Portlet Management section of the Director Administration Console (DAC). It has these sections:

About the Portlet Management section of the DAC

The Portlet Management section of the Director Administration Console (DAC) allows portal administrators to view and modify information about portlets in a deployed exteNd Director application.

Using the DAC, portal administrators can monitor the following portlet elements:

Element	Description
Portlet application	A Java Portlet 1.0-compliant WAR that contains the portlet deployment descriptor portlet.xml and, optionally, other portlet runtime artifacts. See "Administering portlet applications" on page 298.
Portlet definitions	Descriptors in portlet.xml that specify portlet and other configuration parameters. There is one definition for each portlet in an application. See "Administering portlet definitions" on page 302.

Element	Description
Portlet registrations	Registrations of portlets, based on their definitions. Multiple registrations of the same portlet can exist in a single portlet application. See “Administering registered portlets” on page 308.

Administering portlet applications

The Portlet Management section of the DAC allows portal administrators to perform the following tasks related to portlet applications:

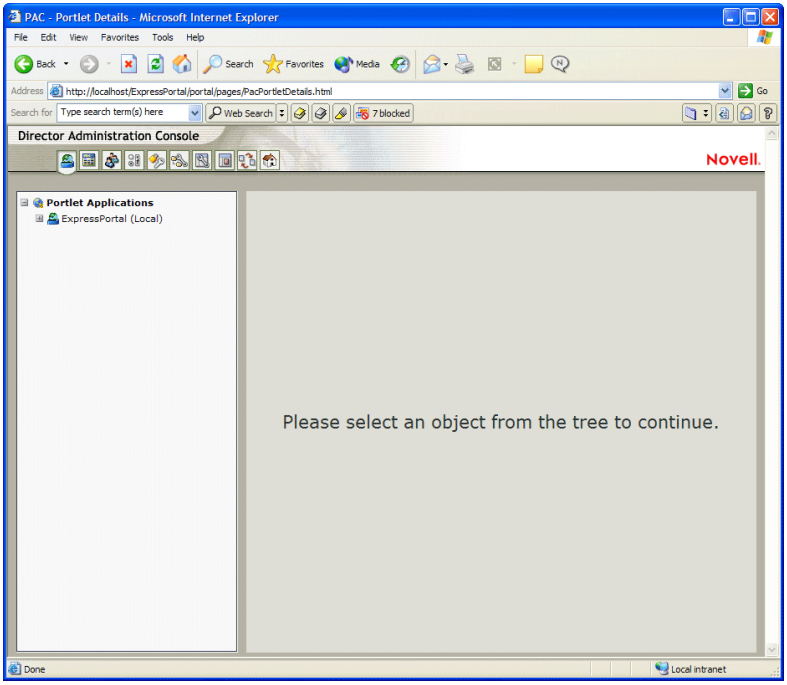
- Access portlet applications on the server
- View information about portlet applications in a read-only panel
- Unregister portlet applications

Accessing portlet applications on the server

➢ **To access a portlet application on the server:**

1. Start the DAC, as described in accessing the DAC.
2. Click the Portlet Management button:

3 A list of all portlet applications deployed to your server appears in the left navigation frame.



Viewing information about portlet applications

Using the DAC, the portal administrator can view the following information about each deployed portlet application:

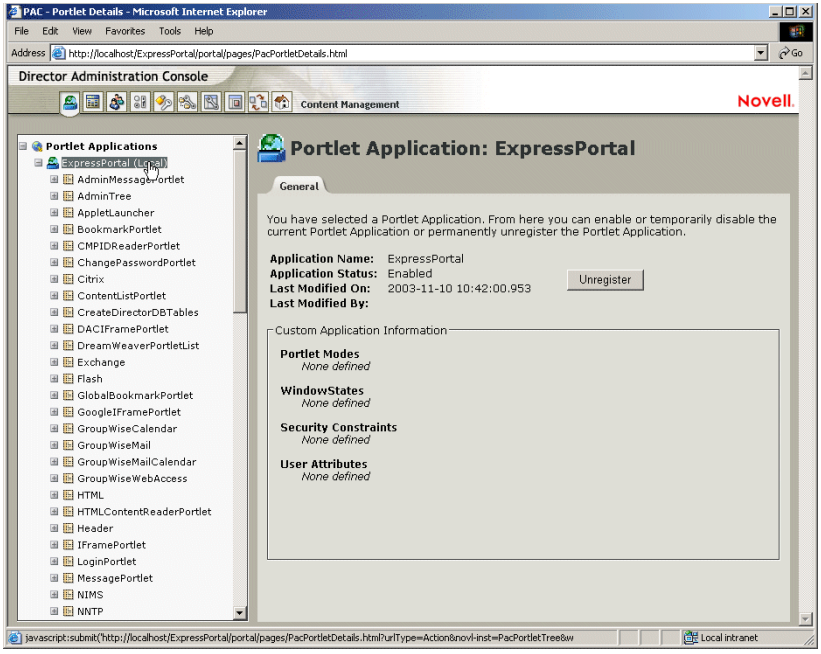
- Name
- Status (whether enabled or disabled)
- Date last modified
- User who last modified the application
- Custom application information: portlet modes, window states, security constraints, and user attributes

  NOTE: exteNd Director allows you to define this custom information. However, the exteNd Director portal recognizes only standard portlet modes (View, Edit, and Help) and standard window states (Normal, Minimized, and Maximized).

➢ **To view information about a portlet application:**

1. Start the DAC, as described in accessing the DAC.
2. Navigate to portlet applications, as described in "Accessing portlet applications on the server" on page 298.
3. Select the portlet application of interest.

   A General panel opens in the right content frame, displaying information about the portlet application:



Unregistering portlet applications

When you want to remove a portlet application from your server, you must unregister the portlet application before undeploying it.

If you unregister a portlet application without first undeploying it, the application is automatically redeployed when the server restarts.

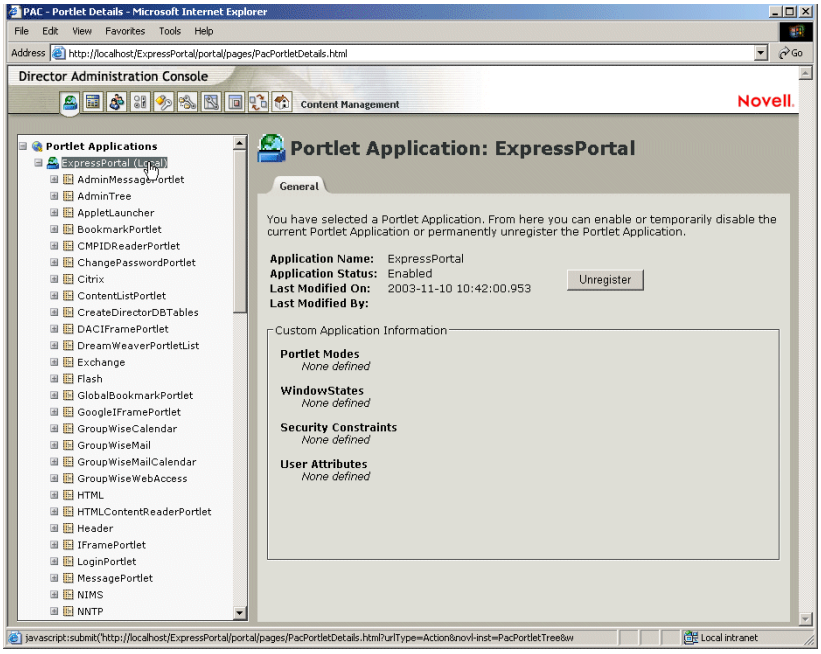
When you unregister a portlet application, all preferences and settings are removed from the exteNd Director database that stores your application data.

NOTE: You cannot unregister the local portlet container, which is a portlet application that is local to the portal. The local portlet container manage portlets that are contained within the portal application.

➤ **To unregister a portlet application:**

1. Start the DAC, as described in accessing the DAC.
2. Navigate to portlet applications, as described in "Accessing portlet applications on the server" on page 298.
3. Select the portlet application of interest.

   A tree view of portlet definitions appears in the left navigation frame and a General panel opens in the right content frame:



4. Click **Unregister**.

   A confirmation window appears.
5. Click **OK** to confirm the action.

   When the process completes, the unregistered portlet application is removed from the list in the navigation window.
6. To remove the portlet application from the server, use your server's tools to undeploy the archive containing the portlet application.

NOTE: To reregister an unregistered portlet application, you must redeploy it.

Administering portlet definitions

The Portlet Management section of the DAC allows portal administrators to perform the following tasks related to portlet definitions in a portlet application:

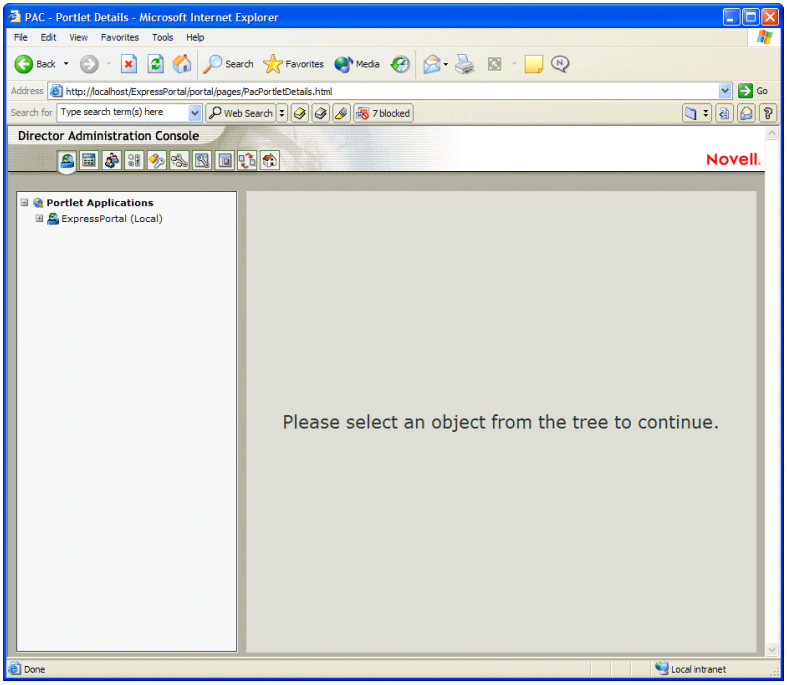
- Access portlet definitions in the deployed portlet application
- Register portlet definitions
- View information about portlet definitions in a read-only panel

Accessing portlet definitions in the deployed portlet application

➢ **To access portlet definitions in the deployed portlet application:**

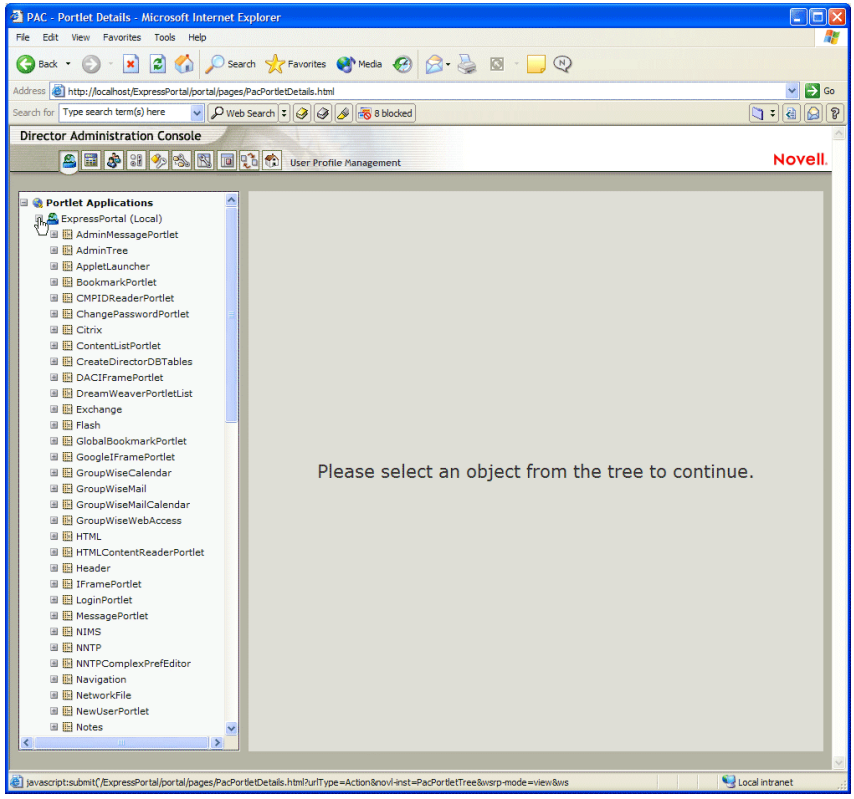
1. Start the DAC, as described in accessing the DAC.
2. Click the Portlet Management button:

3. A list of all portlet applications deployed to your server appears in the left navigation frame.



4 Expand the portlet application of interest.

A tree view appears in the left navigation frame listing all the portlet definitions in the selected portlet application:



Registering portlet definitions

To use a portlet in an application, the portal administrator must create at least one portlet registration based on the portlet definition.

The portlet registration inherits all the preferences and settings of the portlet class, but the portlet administrator has several opportunities to modify these values:

- At registration time using the Portlet Management section of the DAC, as described in “Administering registered portlets” on page 308.
- At page assignment time using the Portal Administration tool, as described in the chapter on administering your portal.

If the portlet definition provides an Edit mode, the portal user can modify specific preferences of the portlet registration at runtime, according to the logic of the portlet's doEdit() method.

TIP: exteNd Director also provides a default implementation for Edit mode. If doEdit() is not explicitly implemented, a default preference sheet is displayed. This sheet lets the user manage preferences for a given portlet and user.

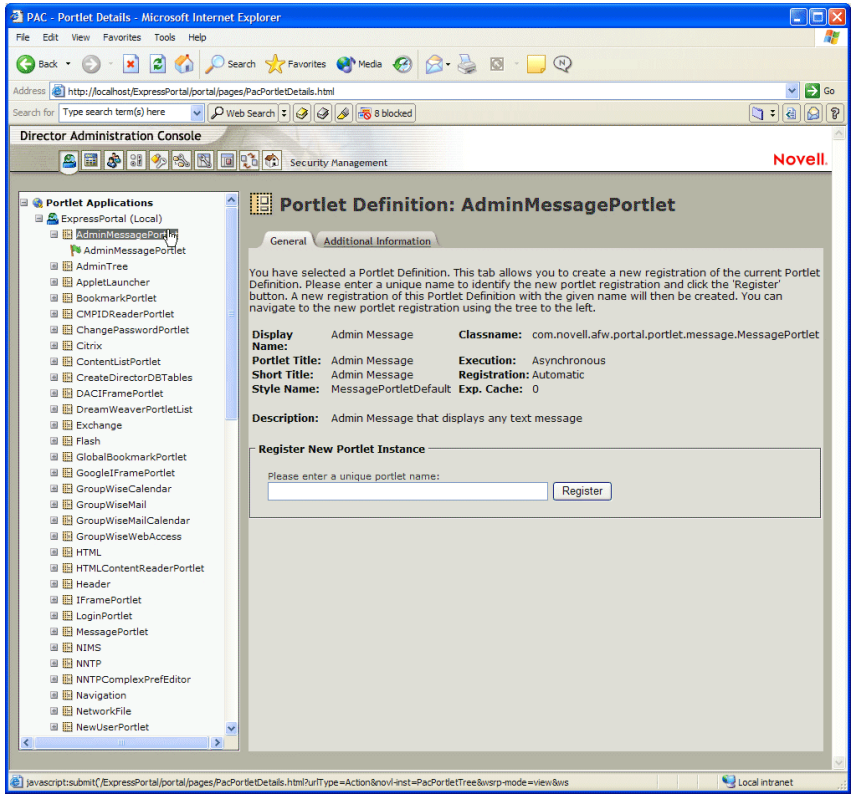
You can put multiple instances of a single portlet on the same page.

➢ **To register a portlet definition:**

1. Start the DAC, as described in accessing the DAC.
2. Navigate to portlet definitions, as described in "Accessing portlet definitions in the deployed portlet application" on page 302.

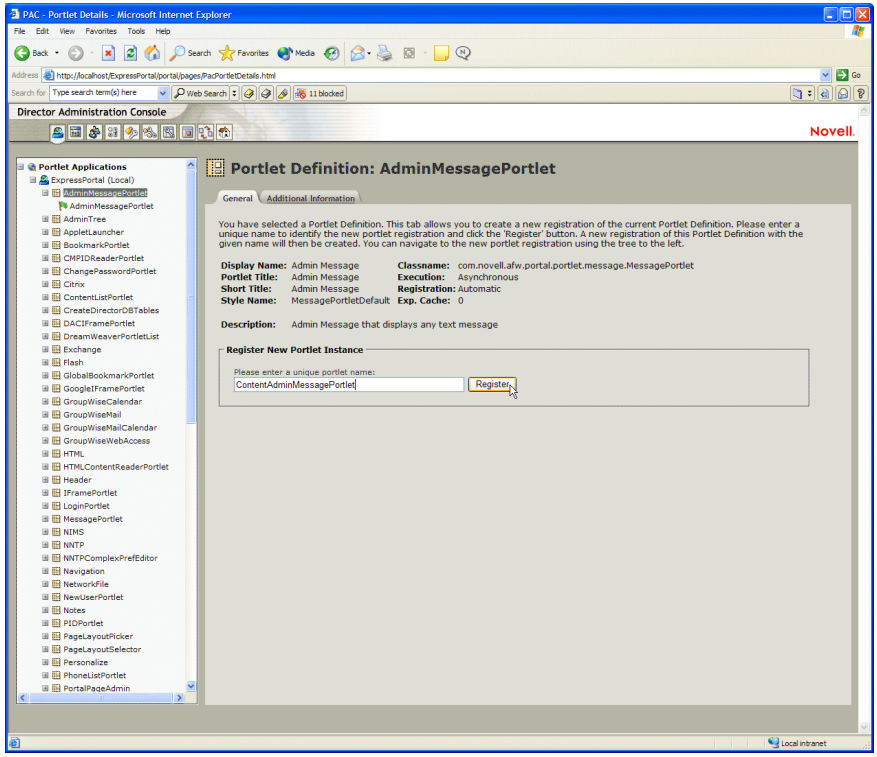
3 Select the portlet definition of interest.

A **General** tab opens in the right content frame, displaying a panel at the bottom called **Register New Portlet Instance**:



All existing registrations of the selected portlet are displayed under the definition, marked by this symbol:

4 In the panel's text field, enter a unique name for the portlet registration, then click **Register**.



The portlet definition is registered with the portlet application. The name of the new definition is displayed in the left navigation frame under the portlet class.

5 If you want to modify the preferences and settings of the new portlet registration, see "Administering registered portlets" on page 308.

Viewing information about portlet definitions

The portal administrator can view the following information in the DAC about portlet definitions:

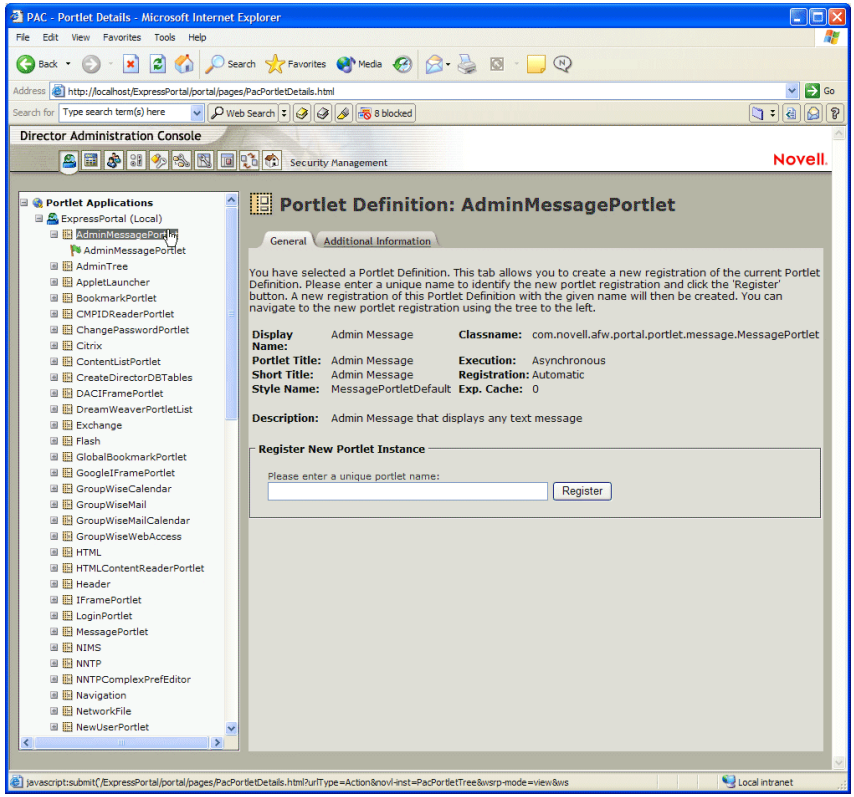
- Display name
- Class name
- Portlet title
- Type of execution (synchronous or asynchronous)
- Short title
- Type of registration
- Style name

- Cache expiration time
- Description
- Initialization parameters
- Keywords
- Supported mime types
- Modes supported by the portlet
- Supported locales
- Supported devices

➢ **To view information about portlet definitions:**

1. Start the DAC, as described in accessing the DAC.
2. Navigate to portlet definitions, as described in "Accessing portlet definitions in the deployed portlet application" on page 302.
3. Select the portlet definition of interest.

   A **General** panel opens in the right content frame, displaying information about the selected portlet definition:



4 Select the **Additional Information** tab to view more information about the selected portlet definition:

Administering registered portlets

The Portlet Management section of the DAC allows portal administrators to perform the following tasks related to registered portlets in a portlet application (also called *portlet registrations*):

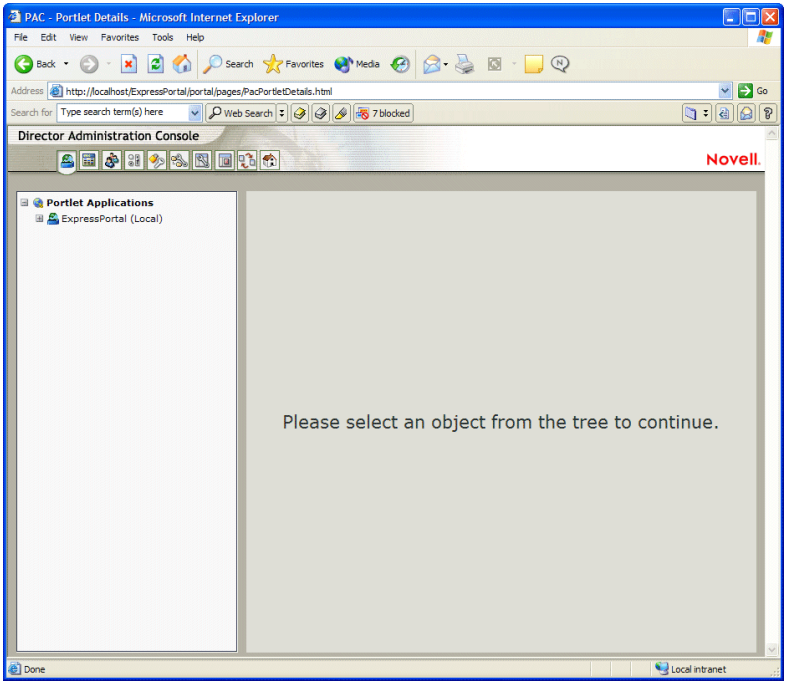
- Accessing portlet registrations in the deployed portlet application
- Viewing information about portlet registrations
- Assigning categories to portlet registrations
- Modifying settings for portlet registrations
- Modifying preferences for portlet registrations
- Assigning security permissions for portlet registrations

Accessing portlet registrations in the deployed portlet application

1 Start the DAC, as described in accessing the DAC.

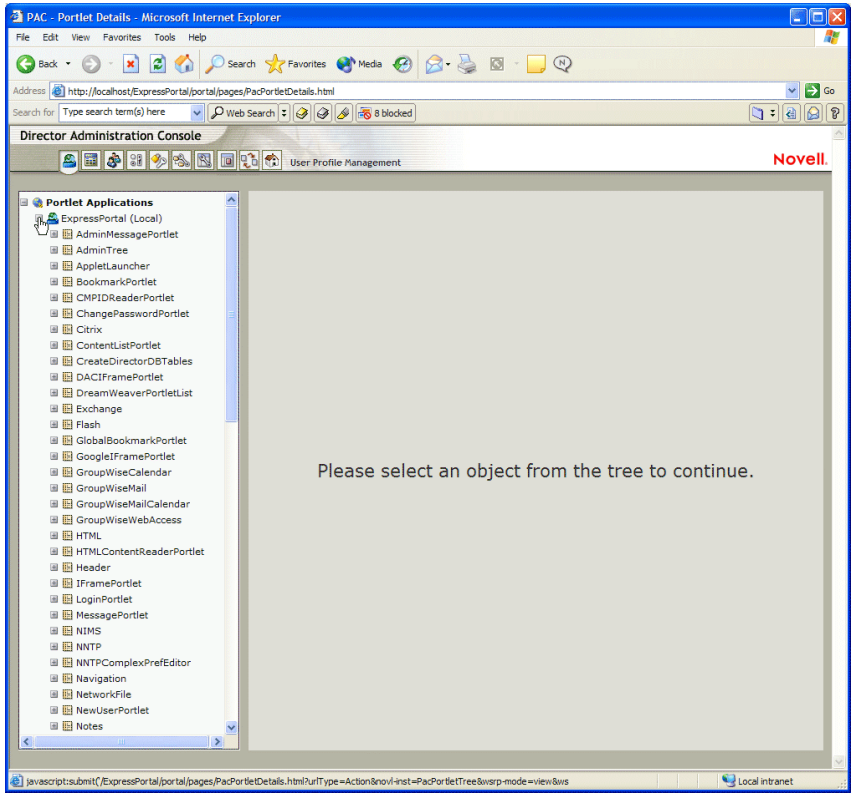
2 Click the Portlet Management button:

3 A list of all portlet applications on your server appears in the left navigation frame, as in this example:



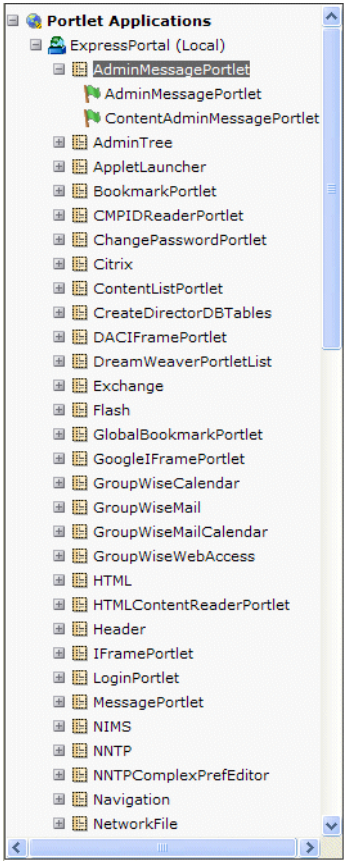
4 Expand the portlet application of interest.

A tree view appears in the left navigation frame listing all the portlet definitions in the selected portlet application:



5 Select the portlet definition of interest.

All registrations of the selected portlet appear in the left navigation window under the definition. Registered portlets are marked with the symbol, as in this example:



Viewing information about portlet registrations

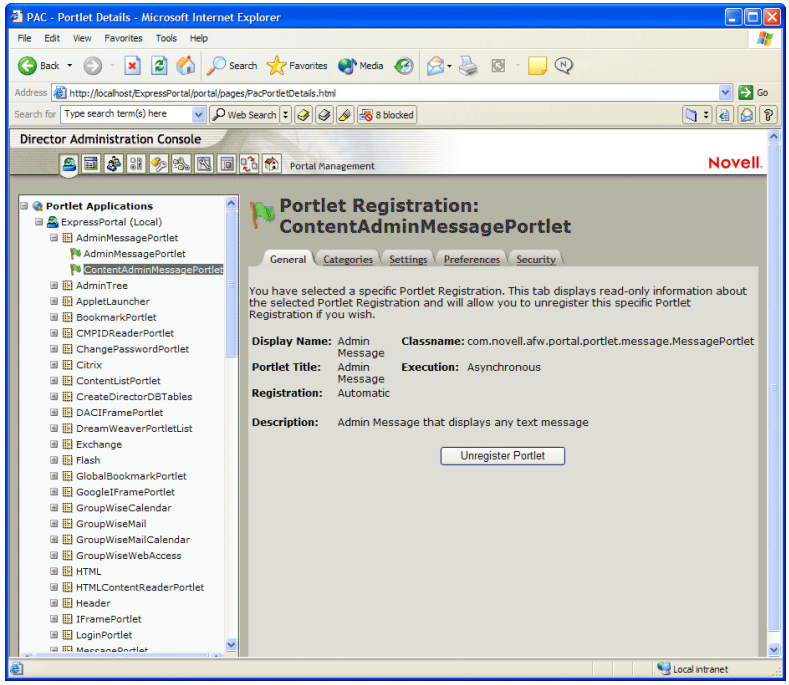
The portal administrator can view the following information in the DAC about registered portlets:

- Display name
- Class name
- Portlet title
- Type of execution (synchronous or asynchronous)
- Type of registration
- Description

➢ **To view information about portlet registrations:**

1. Start the DAC, as described in accessing the DAC.
2. Navigate to portlet registrations, as described in "Accessing portlet registrations in the deployed portlet application" on page 308.
3. Select the portlet registration of interest.

   A **General** panel opens in the right content frame, displaying information about the selected portlet registration:



Assigning categories to portlet registrations

A single production-quality portal can contain a large number of portlets. To facilitate searching for specific portlets in a portlet application, you can organize portlets by category.

➢ **To assign categories to portlet registrations:**

1. Start the DAC, as described in accessing the DAC.
2. Navigate to portlet registrations, as described in "Accessing portlet registrations in the deployed portlet application" on page 308.
3. Select the portlet registration of interest.

   A General panel opens in the right content frame.

4 Select the **Categories** tab.

A list of portal categories appears. The list displays only categories that were defined for portlets and legacy components in the portlet application.

5 Choose one of these actions:

Objective:	Action:
Assign one or more categories to the portlet registration	1 Select each category you want to assign 2 Click: 
Assign all categories to the portlet registration	1 Click: 
Remove one or more category assignments	1 Select each category you want to remove 2 Click: 
Remove all category assignments	1 Click: 

Modifying settings for portlet registrations

Portlet settings define how the portal interacts with individual portlets. Each portlet is configured with the same settings:

- Title
- Maximum time-out
- Requires authentication?
- Display title bar?
- Hidden from user?
- Option defined in the portlet application

Standard Java Portlet 1.0 settings are defined in **portlet.xml**; exteNd Director settings are defined in **novell-portlet.xml**.

The portal administrator can change the values of these settings on an registration by registration basis. In other words, when the portal administrator modifies settings, the new values take effect only for the selected portlet registration.

➢ **To modify portlet registration settings:**

1. Start the DAC, as described in accessing the DAC.
2. Navigate to portlet registrations, as described in "Accessing portlet registrations in the deployed portlet application" on page 308.
3. Select the portlet registration of interest.

   A General panel opens in the right content frame.
4. Select the **Settings** tab.

   A panel opens in the right content pane, listing all the settings defined for the selected portal registration along with their current values. The default values for settings are defined in the portlet definition.
5. Modify settings as desired.

   TIP: After you modify a setting, the **Reset** control becomes active, allowing you to revert back to the default value at any time.
6. Click **Save Settings**.

Modifying preferences for portlet registrations

Portlet preferences are defined by the portlet developer at design time in the portlet deployment descriptors. Preferences vary from portlet to portlet, based on the portlet developer's implementation.

When the portal administrator modifies preferences, the new values take effect only for the selected portlet registration.

➢ **To modify portlet registration preferences:**

1. Start the DAC, as described in accessing the DAC.
2. Navigate to portlet registrations, as described in "Accessing portlet registrations in the deployed portlet application" on page 308.
3. Select the portlet registration of interest.

   A General panel opens in the right content frame.
4. Select the **Preferences** tab.

   A panel opens in the right content pane, listing all the preferences defined for the selected portal registration along with their current values. The default values for preferences are defined in the portlet definition.
5. To get more information about preferences, click the **Descriptions** button at the bottom of the panel
6. Modify preferences as desired.

   TIP: After you modify a preference, the **Reset** control becomes active, allowing you to revert back to the default value at any time.

7 To create a localized version of the preference for each locale specified in the portlet definition, follow these steps:

7a Click the **Detail** control next to the preference of interest.

A new panel appears, displaying the current default values for the selected preference and a list of all locales defined for the portlet definition.

7b For each locale, select **Create New Localized Preference**, enter localized values, and click **Apply**.

7c When you have created all localized preferences, click **OK** to return to the main Preferences panel.

8 Click **Save Preferences**.

Assigning security permissions for portlet registrations

You can assign the following security permissions to users and groups for registered portlet registrations:

Type of permission	What it means
List	Users can **view** the portlet registration from a selection list
Execute	Users can **run** the portlet registration on a portal page

When you modify security permissions, the new values take effect only for the selected portlet registration.

➢ **To assign security permissions for portlet registrations:**

1 Start the DAC, as described in accessing the DAC.

2 Navigate to portlet registrations, as described in "Accessing portlet registrations in the deployed portlet application" on page 308.

3 Select the portlet registration of interest.

A General panel opens in the right content frame.

4 Select the **Security** tab.

5 Choose the **List** or **Execute** tab, depending on what type of permission you want to assign

6 Enter the following information:

Field	What to specify
Search for	Select **Users** or **Groups** from the dropdown menu.
Starts with	1 Enter a text string to streamline your search. 2 Click **Go**.

7 Choose one of these actions:

Objective	Action
Assign security permission	**1** Select one or more users (or groups) in the Results list. **2** Click: 
Remove security permissions	**1** Select one or more users (or groups) in the Current Assignments list. **2** Click: 
Restrict List or Execute security permission to the portal administrator **NOTE:** The portal administrator is a user assigned to the PortalAdmin administration group	**1** Select **List** (or **Execute**) **Permission is Locked Down** **NOTE:** When you enable this option, all other security permission assignments are ignored.

8 Click **Save**.

IV Reference

Provides reference information on the portal tag library and the portal resource descriptors:

22 Portal Tag Library

This chapter provides reference information for the Portal Tag Library (**PortalTag.jar**).

 For background information, see the chapter on using the Director tag libraries in *Developing exteNd Director Applications*.

definition

Description Specifies the profiling definition for the transcoding engine. This tag is used with the deviceProfiling tag.

Syntax `<prefix:definition>`

deviceProfiling

Description Performs a transcoding operation. The definition and sourceXML tags can be nested within this tag.

This tag is used to render content for use with multiple devices using the transcoding engine. The definition tag should contain the appropriate XML data for rendering the content.

JSP pages that use this tag may have to set the return MIME type before calling this custom tag in order to have the content properly displayed. The default content type for JSP pages is text/html. Here is an example of the code you might need to add to the JSP page in order to support a WAP (Wireless Application Protocol) device:

```
public int getBrowser(HttpServletRequest request) {
  String accept = request.getHeader("ACCEPT");
  if (null != accept && -1 !=
    accept.indexOf("wml")) {
    return WML;
  }
  return -1;
}
<%
  switch (getBrowser(request)) {
  case WML:
    response.setContentType("text/vnd.wap.wml");
    break;
  }
%>
```

An alternative way to set the content type is to use the id attribute of the deviceProfiling tag to specify a variable where the content type for the current device should be stored. You can then get the value of this variable by calling the getAttribute() method on the pageContext:

```
<% response.setContentType((String)pageContext.getAttribute
("contenttype")); %>
```

Syntax

```
<prefix:deviceProfiling name="name" style="style" id="id" />
```

Attribute	Required?	Request-time expression values supported?	Description
name	Yes	No	Specifies the name of this rendering.
style	Yes	No	Sets the style ID for this rendering.
id	No	No	Specifies the name of the variable that will be used to hold the content type for the current device. If no value is specified, a default id of **trans_contentType** is used.

Example

This example shows how to use the deviceProfiling tag to perform a transcoding operation. In this example, the profiling definition is hardcoded within the definition tag. Typically, you would use the getResource tag (in the Framework tag library) to get the XML from the resource set dynamically.

```
<%@ taglib uri="/fw" prefix="epfw" %>
<%@ taglib uri="/portal" prefix="ep" %>

<% try { %>
<ep:deviceProfiling style="PhoneListStyle" name="testtwo"
id="phone">
  <ep:definition>
   <transcoding lastOpen="d:\PhoneListData.xml"
    xmlns="urn:novell:dp_metadata">
    <separator tagname='employee'/>
    <block name='employeeList' type='list' style='' listlength="4">
     <element name='firstName' type='' location='//first-
name/text()'/>
     <element name='lastName' type='' location='//last-
name/text()'/>
    </block>
    <block name='employeeDetails' type='content' style=''>
     <element name='firstName' type='' location='//first-
name/text()'/>
     <element name='phoneNumber' type='' location='//phone-
number/text()'/>
     <element name='email' type='' location='//email/text()'/>
     </block>
    <link from-block='employeeList' from-element='firstName' to-
block='employeeDetails'/>
   </transcoding>
  </ep:definition>
  <ep:sourceXML>
```

```
   <results querystring="s">
    <employee>
    <first-name>Katy</first-name>
    <last-name>Summers</last-name>
    <phone-number>(617) 343-6530</phone-number>
    <email>ksummers@silverdemo.com</email>
    </employee>
    <employee>
    <first-name>Hank</first-name>
    <last-name>Smiley</last-name>
    <phone-number>(617) 343-6532</phone-number>
    <email>hsmiley@silverdemo.com</email>
    </employee>
...
   </results>
  </ep:sourceXML>
  </ep:deviceProfiling>
<%} catch (Exception e) {
    System.out.println("Error");
    e.printStackTrace();
} %>
<%
response.setContentType((String)pageContext.getAttribute("phone"));
%>
```

displayComponent

Description

Displays a portal component within a JSP page. To pass parameters to the component, you can nest the param tag inside the displayComponent tag.

This tag wraps the displayComponent() method on the EbiPresentationMgr interface.

Syntax

```
<prefix:displayComponent compID="compID" name="name"
decorate="decorate" id="id"/>
```

Attribute	Required?	Request-time expression values supported?	Description
compID	Yes	Yes	Specifies the ID for a portal component. The ID of a component is the name given to the XML file (without the extension) that describes the component. For example, if the XML file name for a component is MyComponent1.xml, the component ID is MyComponent1.
name	Yes	No	Specifies a component instance name that uniquely identifies the component on the page.
decorate	No	Yes	Specifies a boolean value (true or false) that indicates whether or not to show the title bar. Although the toolbars display when this option is set to true, the toolbar buttons will not automatically function as they would on the MyPortal page or a PID page. For example, minimize and maximize will not work automatically. In addition, the processRequest method for the component will not fire, unless the fireComponentProcessRequest tag is used as well. To theme-enable your component, you also need to be sure to include a getThemeLink tag on the page. If no value is specified, this attribute defaults to false.
id	Yes	No	Specifies the name of the variable that will be used to hold the component content. If no value is specified, the component content is returned directly.

Example 1

This example shows how to use the displayComponent to send component content to the output stream:

```
<%@ taglib uri="/portal" prefix="ep" %>
<html>
<body>

<b>DisplayComponent</b>

<ep:displayComponent compID="MyComponent1" name="mycomp1">
<ep:param name="Parm1" value="Val1"/>
<ep:param name="Parm2" value="Val2"/>
<ep:param name="Parm3" value="Val3"/>
</ep:displayComponent>

</body>
</html>
```

Example 2

This example show how to use the displayComponent to assign component content to the id variable. In this example, the id variable is used to store a theme ID generated by the PortalUrlHelper component:

```
<head>
<ep:displayComponent compID="PortalUrlHelper" name="helper"
id="themeid">
 <ep:param name="SUBST_STRING" value="$THEME_ID$"/>
</ep:displayComponent>

<%=pageContext.getAttribute("themeid")%>

</head>
```

displayPID

Description

Displays a PID page within a JSP page.

This tag wraps the displayPage() method on the EbiPresentationMgr interface.

Syntax

`<prefix:displayPID PID="pid" />`

Attribute	Required?	Request-time expression values supported?	Description
PID	Yes	No	Specifies the name of a PID page.

Example

```
<% taglib uri="/portal" prefix="ep" %>
...
<ep:displayPID PID="MyPID.html" />
```

fireComponentProcessRequest

Description

Fires the processRequest method of a component. To pass parameters to the processRequest method, you can nest the param tag inside the fireComponentProcessRequest tag.

This tag wraps the processRequest() method on the EbiPortalComponent interface.

Syntax

```
<prefix:fireComponentProcessRequest compID="compID" name="name" />
```

Attribute	Required?	Request-time expression values supported?	Description
compID	Yes	Yes	Specifies the ID for a portal component. The ID of a component is the name given to the XML file (without the extension) that describes the component. For example, if the XML file name for a component is MyComponent1.xml, the component ID is MyComponent1.
name	No	Yes	Specifies a component instance name that uniquely identifies the component on the page.

Example

```
<%@ taglib uri="/portal" prefix="ep" %>
<html>
<body>
<ep:fireComponentProcessRequest compID="MyComponent1" name="mycomp1">
<ep:param name="Parm1" value="Val1"/>
<ep:param name="Parm2" value="Val2"/>
<ep:param name="Parm3" value="Val3"/>
</ep:fireComponentProcessRequest>

</body>
</html>
```

getCompList

Description

Retrieves a list of all portal components. To use this tag, the user must be part of the administrators group and have proper authority to get this list of objects. Before using this tag, the user must log in.

This tag wraps the getComponentInfoList() method on the EbiComponentManager interface.

Syntax

```
<prefix:getCompList id="id" iterate="iterate" restricted="restricted"
category="category"/>
```

Attribute	Required?	Request-time expression values supported?	Description
id	No	No	Specifies the name of the variable that will be used to hold the resulting component list. The list is an object of type List that contains several objects of type EbiPortalComponentInfo. If no value is specified, a default id of **compList** is used.
iterate	Yes	No	Specifies a boolean value (true or false) that indicates whether this tag will operate as a body tag so that each row can be processed separately. If the iterate attribute is set to true, the following values can be accessed from within the getCompList tag: ◆ compName ◆ description ◆ displayname Each of these variables has a scope of NESTED. If the iterate attribute is set to false, this tag will operate as a nonbody tag that returns an object of type List that contains a list of objects of type EbiPortalComponentInfo.

Attribute	Required?	Request-time expression values supported?	Description
restricted	No	Yes	Specifies a boolean value (true or false) that indicates whether the list of components should be filtered based on security permissions. Defaults to true.
category	No	Yes	Specifies the name of a category.

Example 1

This example shows how to use the getCompList tag with the iterate attribute set to **true**:

```
<%@ page language="java"
        session="true"
        isThreadSafe="true"
        contentType="text/html; charset=ISO-8859-1"
        buffer="none"
        autoFlush="true"
        import="com.sssw.portal.*"%>
<%@ taglib uri="/fw" prefix="fw" %>
<%@ taglib uri="/portal" prefix="ep" %>
<html>
<head>
</head>
<body>
<fw:login userid="admin" password="admin"/>
<!-- login a user to the portal...-->
<ep:getCompList iterate="true">
Component name = <%=compName%><br/>
Description = <%=description%><br/>
DisplayName = <%=displayname%><br/>
<p/>
</ep:getCompList>
<fw:logoff />
</body>
</html>
```

Example 2

This example shows how to use the getCompList tag with the iterate attribute set to **false**:

```
<%@ page language="java"
        session="true"
        isThreadSafe="true"
        contentType="text/html; charset=ISO-8859-1"
        buffer="none"
        autoFlush="true"
        import="com.sssw.portal.*"%>
<%@ taglib uri="/portal" prefix="ep" %>
```

```
<%@ taglib uri="/fw" prefix="fw" %>
<html>
<head>
</head>
<body>
<fw:login userid="admin" password="admin"/>
<!-- login a user to the portal...-->
<ep:getCompList iterate="false"/>

<%= ((java.util.List)pageContext.getAttribute("compList")).size()
%> = the size of the list...
<fw:logoff />
</body>
</html>
```

getObjectInCache

Description

Retrieves an object from the portal cache. Objects are retrieved based on the cache key passed in on the key attribute and cast to the class provided by the className attribute. The result is placed within a scripting variable with a page scope that is named by the tag's id attribute value.

This tag wraps the getObjectInCache() method on the EbiCacheHolder interface.

Syntax

```
<prefix:getObjectInCache id="ID" key="key" className="class"
sessionCache="sessionCache" />
```

Attribute	Required?	Request-time expression values supported?	Description
id	No	No	Specifies the name of the variable that will be used to store the cached object. If no value is specified, a default id of **cacheObject** is used.
key	Yes	Yes	Specifies the key for the object in the portal cache.
className	No	No	Specifies the name of the class to which the cached object should be cast. If no value is specified, the object is cast to an Object.

Attribute	Required?	Request-time expression values supported?	Description
sessionCache	No	No	Indicates whether the object should be retrieved from the session-specific cache. If a value of true is specified, the object is retrieved from the session cache. If a value of false is specified, the object is retrieved from the global cache. If no value is specified, this attribute defaults to false.

Example

```
<% taglib uri="/portal" prefix="ep" %>
...
<ep:getObjectInCache id="myID" key="portalkey"
className="java.lang.String" />
...
Value is: <%=myID%>
```

getObjectInSessionCache

Description

Retrieves an object from the session cache. Objects are retrieved based on the cache key passed in on the key attribute and cast to the class provided by the className attribute. The result is placed within a variable with a page scope that is keyed by the tag's id attribute value.

This tag wraps the getObjectInCache() method on the EbiSession interface.

NOTE: This tag has been deprecated. Use the getObjectInCache tag instead.

Syntax

```
<prefix:getObjectInSessionCache id="ID" key="key" className="class" />
```

Attribute	Required?	Request-time expression values supported?	Description
id	No	No	Specifies the name of the variable that will be used to store the cached object. If no value is specified, a default id of **sessionObject** is used.

Attribute	Required?	Request-time expression values supported?	Description
key	Yes	Yes	Specifies the key for the object in the portal cache.
className	No	No	Specifies the name of the class to which the cached object should be cast. If no value is specified, the object is cast to an Object.

Example

```
<% taglib uri="/portal" prefix="ep" %>
...
<ep:getObjectInSessionCache id="sesobj" key="sessionkey"
className="java.lang.String" />
...
Value is: <%=sesobj%>
```

getPIDList

Description

Retrieves a list of all PID pages. To use this tag, the user must be part of the administrators group and have proper authority to get this list of objects. Before using this tag, the user must log in.

This tag wraps the getPageInfoList() method on the EbiPageManager interface.

Syntax

```
<prefix:getPIDList id="id" iterate="iterate" category="category"/>
```

Attribute	Required?	Request-time expression values supported?	Description
id	No	No	Specifies the name of the variable that will be used to hold the resulting list of pages. The list is an object of type List that contains several objects of type EbiPageInfo. If no value is specified, a default id of **PIDList** is used.

Attribute	Required?	Request-time expression values supported?	Description
iterate	Yes	No	Specifies a boolean value (true or false) that indicates whether this tag will operate as a body tag so that each row can be processed separately. If the iterate attribute is set to true, the following values can be accessed from within the getPIDList tag: ◆ identifier ◆ mime ◆ displayname Each of these variables has a scope of NESTED. If the iterate attribute is set to false, this tag will operate as a nonbody tag that returns an object of type List that contains a list of objects of type EbiPageInfo.
category	No	Yes	Specifies the name of a category.

Example 1

This example shows how to use the getPIDList tag with the iterate attribute set to **true**:

```
<%@ page language="java"
        session="true"
        isThreadSafe="true"
        contentType="text/html; charset=ISO-8859-1"
        buffer="none"
        autoFlush="true"
        import="com.sssw.portal.*"%>
<%@ taglib uri="/fw" prefix="fw" %>
<%@ taglib uri="/portal" prefix="ep" %>
<html>
<head>
</head>
<body>
<fw:login userid="admin" password="admin"/>
<!-- login a user to the portal...-->
<ep:getPIDList iterate="true">
Identifier = <%=identifier%><br/>
mime = <%=mime%><br/>
Description = <%=displayname%><br/>
<p/>
</ep:getPIDList>
<fw:logoff />
```

```
</body>
</html>
```

Example 2 This example shows how to use the getPIDList tag with the iterate attribute set to **false**:

```
<%@ page language="java"
        session="true"
        isThreadSafe="true"
        contentType="text/html; charset=ISO-8859-1"
        buffer="none"
        autoFlush="true"
        import="com.sssw.portal.*"%>
<%@ taglib uri="/portal" prefix="ep" %>
<%@ taglib uri="/fw" prefix="fw" %>
<html>
<head>
</head>
<body>
<fw:login userid="admin" password="admin"/>
<!-- login a user to the portal...-->
<ep:getPIDList iterate="false"/>

<%= ((java.util.List)pageContext.getAttribute("PIDList")).size() %>
= the size of the list of pages...

<fw:logoff />
</body>
</html>
```

getUserPortalInfo

Description Retrieves portal information for the current user or a specified user.

This tag wraps the getUserPortalInfo() method on the EbiPortalManager interface.

Syntax

```
<prefix:getUserPortalInfo userIID="useriid" id="id"/>
```

Attribute	Required?	Request-time expression values supported?	Description
userID	No	Yes	Specifies the UUID for a user. If no value is specified, the current user is used.

Attribute	Required?	Request-time expression values supported?	Description
id	No	No	Specifies the name of the variable that will be used to hold the portal information object. The object returned is of type EbiUserPortalInfo. If no value is specified, a default id of **portalinfo** is used.

Example

This example shows how to use the getUserPortalInfo tag to get information about the current user:

```
<%@ page language="java"
        session="true"
        isThreadSafe="true"
        contentType="text/html; charset=ISO-8859-1"
        buffer="none"
        autoFlush="true"
        import="com.sssw.portal.*"%>
<%@ taglib uri="/fw" prefix="fw" %>
<%@ taglib uri="/portal" prefix="ep" %>
<html>
<head>
</head>
<body>
<fw:login userid="admin" password="admin"/>
<!-- login a user to the portal...-->
<ep:getUserPortalInfo/>

<%=
((com.sssw.portal.api.EbiUserPortalInfo)pageContext.getAttribute("p
ortalinfo")).getDefaultUserPage() %> is the default page for this
user...

<fw:logoff />
</body>
</html>
```

getUserPageInfo

Description

Retrieves portal page information for the current user or a specified user.

This tag wraps the getUserPageInfo() method on the EbiPortalManager interface.

Syntax

```
<prefix:getUserPageInfo userIID="useriid" id="id"/>
```

Attribute	Required?	Request-time expression values supported?	Description
userPageName	No	Yes	Specifies the name of the personal page.
userID	No	Yes	Specifies the UUID for a user. If no value is specified, the current user is used.
id	No	No	Specifies the name of the variable that will be used to hold the page information object. The object returned is of type EbiUserPageInfo. If no value is specified, a default id of **pageinfo** is used.

Example

This example shows how to use the getUserPageInfo tag to get information about a particular personal page:

```
<%@ page language="java"
        session="true"
        isThreadSafe="true"
        contentType="text/html; charset=ISO-8859-1"
        buffer="none"
        autoFlush="true"
        import="com.sssw.portal.*"%>
<%@ taglib uri="/fw" prefix="fw" %>
<%@ taglib uri="/portal" prefix="ep" %>
<html>
<head>
</head>
<body>
<fw:login userid="admin" password="admin"/>
<!-- login a user to the portal...-->
<ep:getUserPortalInfo userPageName="MyUserPage"/>
```

```
<%=
((com.sssw.portal.api.EbiUserPageInfo)pageContext.getAttribute("pag
einfo")).getPortalLayoutID() %> is the layout ID for the page...

<fw:logoff />
</body>
</html>
```

getUserComponentInfo

Description

Retrieves the requested user component information for the current user or a specified user.

This tag wraps the getUserComponentInfo() method on the EbiPortalManager interface.

Syntax

```
<prefix:getUserComponentInfo compID="compID" userIID="useriid"
id="id"/>
```

Attribute	Required?	Request-time expression values supported?	Description
compID	Yes	Yes	Specifies a component ID.
userID	No	Yes	Specifies the UUID for a user. If no value is specified, the current user is used.
id	No	No	Specifies the name of the variable that will be used to hold the component information object. The object returned is of type EbiUserComponentInfo. If no value is specified, a default id of **componentinfo** is used.

Example

This example shows how to use the getUserComponentInfo tag to get information about a particular component:

```
<%@ page language="java"
        session="true"
        isThreadSafe="true"
        contentType="text/html; charset=ISO-8859-1"
        buffer="none"
        autoFlush="true"
```

```
        import="com.sssw.portal.*"%>
<%@ taglib uri="/fw" prefix="fw" %>
<%@ taglib uri="/portal" prefix="ep" %>
<html>
<head>
</head>
<body>
<fw:login userid="admin" password="admin"/>
<!-- login a user to the portal...-->
<ep:getUserComponentInfo compID="PhoneList"/>

<%=
((com.sssw.portal.api.EbiUserPortalInfo)pageContext.getAttribute("c
omponentinfo")).getComponentName() %> is the default page for this
user...

<fw:logoff />
</body>
</html>
```

getUserPageList

Description

Retrieves a list of all available pages for the current user or a specified user.

This tag wraps the getUserPageInfoList() method on the EbiPortalManager interface.

Syntax

```
<prefix:getUserPageList id="id" userIID="userIID"
iterate="iterate"/>
```

Attribute	Required?	Request-time expression values supported?	Description
id	No	No	Specifies the name of the variable that will be used to hold the resulting list of pages. The list is an object of type List that contains several objects of type EbiUserPageInfo. If no value is specified, a default id of **userPageList** is used.
userID	No	Yes	Specifies the UUID for a user. If no value is specified, the current user is used.

Attribute	Required?	Request-time expression values supported?	Description
iterate	Yes	No	Specifies a boolean value (true or false) that indicates whether this tag will operate as a body tag so that each row can be processed separately. If the iterate attribute is set to true, the following values can be accessed from within the getUserPageList tag: ◆ pagename ◆ description ◆ displayname Each of these variables has a scope of NESTED. If the iterate attribute is set to false, this tag will operate as a nonbody tag that returns an object of type List that contains a list of objects of type EbiUserPageInfo.

Example 1

This example shows how to use the getUserPageList tag with the iterate attribute set to **true**:

```
<%@ page language="java"
        session="true"
        isThreadSafe="true"
        contentType="text/html; charset=ISO-8859-1"
        buffer="none"
        autoFlush="true"
        import="com.sssw.portal.*"%>
<%@ taglib uri="/fw" prefix="fw" %>
<%@ taglib uri="/portal" prefix="ep" %>
<html>
<head>
</head>
<body>
<fw:login userid="admin" password="admin"/>
<!-- login a user to the portal...-->
<ep:getUserPageList iterate="true">
Page name = <%=pagename%><br/>
Description = <%=description%><br/>
Display name = <%=displayname%><br/>
<p/>
</ep:getUserPageList>
<fw:logoff />
</body>
</html>
```

Example 2

This example shows how to use the getUserPageList tag with the iterate attribute set to **false**:

```
<%@ page language="java"
        session="true"
        isThreadSafe="true"
        contentType="text/html; charset=ISO-8859-1"
        buffer="none"
        autoFlush="true"
        import="com.sssw.portal.*"%>
<%@ taglib uri="/portal" prefix="ep" %>
<%@ taglib uri="/fw" prefix="fw" %>
<html>
<head>
</head>
<body>
<fw:login userid="admin" password="admin"/>
<!-- login a user to the portal...-->
<ep:getUserPageList iterate="false"/>

<%=
((java.util.List)pageContext.getAttribute("userPageList")).size()
%> = the size of the list of pages...

<fw:logoff />
</body>
</html>
```

getThemeLink

Description

Retrieves the URL for the current theme and generates an appropriate link to a theme CSS file. If the id is not set, it will return the string for the link tag directly. If the id is set it will return the link as an attribute of that name. For inline calls, this tag should be placed in the <head> section of the HTML so that the link to the CSS file is positioned properly.

This tag wraps the replaceAllKeywordStrings() method on the EboPortalUrlHelper class.

Syntax

```
<prefix:getThemeLink id="id"/>
```

Attribute	Required?	Request-time expression values supported?	Description
id	No	No	Specifies the name of the variable that will be used to hold the link. If no value is specified, the link is placed inline on the page.

Example

This example shows how to use the getThemeLink tag to generate a theme link in the HEAD section of a page:

```
<%@ page language="java"
        session="true"
        isThreadSafe="true"
        contentType="text/html; charset=ISO-8859-1"
        buffer="none"
        autoFlush="true"
        import="com.sssw.portal.*"%>
<%@ taglib uri="/fw" prefix="fw" %>
<%@ taglib uri="/portal" prefix="ep" %>
<html>
<head>
<ep:getThemeLink/>
</head>
<body>
...
</body>
</html>
```

param

Description

Specifies a parameter that will be passed to the tag within which it is nested. The param tag can be nested within the displayComponent and the fireComponentProcessRequest tags.

This tag is a nested tag. It can only be used as the child of another tag.

Syntax

```
<prefix:param name="name" value="value" />
```

Attribute	Required?	Request-time expression values supported?	Description
name	Yes	Yes	Specifies the name of the parameter.
value	Yes	Yes	Specifies a value for the parameter.

Example

```
<%@ taglib uri="/portal" prefix="ep" %>
<html>
<body>

<b>DisplayComponent</b>

<ep:displayComponent compID="MyComponent1" name="mycomp1">
<ep:param name="Parm1" value="Val1"/>
<ep:param name="Parm2" value="Val2"/>
<ep:param name="Parm3" value="Val3"/>
</ep:displayComponent>

</body>
</html>
```

PortalUrlHelper

Description

Provides a mechanism for retrieving and parsing portal URLs. To do this, it instantiates a portal context object and translates the provided string, substituting any portal replacement strings with current valid URL information.

 For more information on portal replacement strings, see Chapter 23, “Portal Replacement Strings”.

This tag wraps the replaceAllKeywordStrings() method on the EboPortalUrlHelper class.

Syntax

```
<prefix:PortalUrlHelper urlString="urlString" id="id"/>
```

Attribute	Required?	Request-time expression values supported?	Description
urlString	Yes	Yes	Specifies the URL string that should be parsed.
id	No	No	Specifies the name of the variable that will be used to hold the requested URL. If no value is specified, the URL is placed inline on the page.

Example

This example shows how to use the PortalUrlHelper tag to get the current theme ID:

```
<%@ page language="java"
        session="true"
        isThreadSafe="true"
        contentType="text/html; charset=ISO-8859-1"
        buffer="none"
        autoFlush="true"
        import="com.sssw.portal.*"%>
<%@ taglib uri="/fw" prefix="fw" %>
<%@ taglib uri="/portal" prefix="ep" %>
<html>
<head>
</head>
<body>
The current theme is: <ep:PortalUrlHelper urlString="$THEME_ID$"/>

</body>
</html>
```

putObjectInCache

Description

Puts an object in the portal cache. The object passed in on the object attribute is cached based on the cache key passed in on the key attribute.

This tag wraps the putObjectInCache() method on the EbiCacheHolder interface.

Syntax

```
<prefix:putObjectInCache key="key" object="object"
sessionCache="sessionCache" />
```

Attribute	Required?	Request-time expression values supported?	Description
key	Yes	Yes	Specifies the key for the object in the portal cache.
object	Yes	Yes	Passes in the object that will be cached.
sessionCache	No	No	Indicates whether the object should be put in the session-specific cache. If a value of true is specified, the object is put in the session cache. If a value of false is specified, the object is put in the global cache. If no value is specified, this attribute defaults to false.

Example

```
<% taglib uri="/portal" prefix="ep" %>
...
<ep:putObjectInCache key="portalkey"
object="<%=pageContext.getAttribute("test")%>" />
```

putObjectInSessionCache

Description

Puts an object in the portal cache that is linked to the current session. The object passed in on the object attribute is cached based on the cache key passed in on the key attribute.

This tag wraps the putObjectInCache() method on the EbiSession interface.

NOTE: This tag has been deprecated. Use the putObjectInCache tag instead.

Syntax

```
<prefix:putObjectInSessionCache key="key" object="object" />
```

Attribute	Required?	Request-time expression values supported?	Description
key	Yes	Yes	Specifies the key for the object in the portal cache.
object	Yes	Yes	Passes in the object that will be cached.

Example

```
<% taglib uri="/portal" prefix="ep" %>
...
    <ep:putObjectInSessionCache key="sessionkey" object="mytestvalue"
    />
```

removeObjectFromCache

Description

Removes an object from the portal cache.

This tag wraps the removeObjectInCache() method on the EbiCacheHolder interface.

Syntax

```
<prefix:removeObjectInCache key="key" sessionCache="sessionCache" />
```

Attribute	Required?	Request-time expression values supported?	Description
key	Yes	Yes	Specifies the key for the object in the portal cache.
sessionCache	No	No	Indicates whether the object should be removed from the session-specific cache. If a value of true is specified, the object is removed from the session cache. If a value of false is specified, the object is removed from the global cache. If no value is specified, this attribute defaults to false.

Example

```
<% taglib uri="/portal" prefix="ep" %>
...
<ep:removeObjectInCache key="portalkey" %>" />
```

sourceXML

Description

Specifies source XML that is to be transformed by the transcoding engine. Used with the deviceProfiling tag. The sourceXML tag should contain appropriate XML data for rendering the content.

Syntax

```
<prefix:sourceXML>
```

23 Portal Replacement Strings

This chapter provides reference information for the portal replacement strings. It includes the following sections:

About replacement strings

exteNd Director allows you to use *replacement strings* to include runtime, context-based information in a portal application. Replacement strings are keywords that reference things that can change dynamically at runtime, such as a user's current theme or the currently rendered portlet ID. When these keywords are detected at runtime, they are replaced based on information from the current HTTP request and/or the current portal context.

All portal replacement strings are defined as constants in the *EboPortalUrlHelper* class in the com.sssw.portal.util package.

Where replacement strings can be used

Replacement strings can be used in several places in a portal application:

- Portal pages (PID pages and JSP pages)
- Descriptors for portal resources
- WAR descriptor (web.xml) for a custom Web application

Portal pages

In a PID page or JSP page, you can use replacement strings to construct URLs, URIs, and query strings for the following HTML elements:

- HREF attribute of anchor tags (<A>)
- SRC attribute of image tags (<IMG>)

exteNd Director provides a portlet called **PortalUrlHelper** to give an easy way to use replacement strings on a portal page. exteNd Director also provides a custom tag called **PortalUrlHelper** that performs the same function.

📖 For details on using the PortalUrlHelper custom tag, see "PortalUrlHelper" on page 340.

Descriptors for portal resources

You can use replacement strings in the descriptors for the following portal resources:

Resource type	Elements searched for replacement strings
Portal layouts	◆ preview-image
Portal themes	◆ preview-image ◆ thumbnail-image

Resource type	Elements searched for replacement strings
Portal options	◆ link ◆ image
Portlets	◆ link ◆ image ◆ option ◆ default

WAR descriptor for a custom Web application

Replacement strings can also be used in the following context-param values in the web.xml file for a custom Web application:

- PortalLoginPage
- NewUserPage
- Portal Home Page
- Portal Resource Path

Categories of replacement strings

There are several categories of replacement string keywords:

Category	Keywords
URLs	◆ $context$ ◆ $CONTEXT_URL$ ◆ $ENCODED_REQUEST_URL$ ◆ $PORTAL_URL$ ◆ $PORTAL_URL$ ◆ $PORTLET_PATH$ ◆ $RESOURCE_URL$ ◆ $SERVLET_URL$ ◆ $THEME_URL$

Category	Keywords
Paths	◆ $COMP_PATH$ ◆ $COMPONENT_PATH$ ◆ $CONTEXT_PATH$ ◆ $PAGE_PATH$ ◆ $PAGE_PATH$ ◆ $PORTAL_URL$ ◆ $PORTLET_PATH$ ◆ $REQUEST_URI$ ◆ $SERVLET_PATH$ ◆ $THEME_PATH$
IDs	◆ $COMPONENT_ID$ ◆ $COMPONENT_INSTANCE_ID$ ◆ $THEME_ID$
Other	◆ $HOST$ ◆ $HOST_PORT$ ◆ $SCHEME$

$COMP_PATH$

Description Extra path information following the path to the controller servlet that tells the servlet to serve a component's **undecorated** content.

The $COMP_PATH$ keyword is equivalent to the PortalPathCompKey context parameter, defined as **comp** in the web.xml file for the portal WAR.

Example `$PORTAL_URL$/$COMP_PATH$/HelloWorldComponent`

This example is parsed and converted to:

`http://host/context/main/comp/HelloWorldComponent`

$COMPONENT_ID$

Description Portlet Registration ID or Component ID of the user's currently selected portlet or component.

Example

The following example is based on a link on a page that references the URL `http://host/context/main/MyPage/finance` and a current component called **HelloWorld**:

```
?ss_action=remove&ss_comp=$COMPONENT_ID$&ss_instance=$COMPONENT_INS
TANCE_ID$
```

This example is parsed and converted to:

```
http://host/context/main/MyPage/finance?ss_action=remove&ss_comp=He
lloWorld&ss_instance=123456
```

$COMPONENT_INSTANCE_ID$

Description

Instance ID of the user's currently selected component.

Example

The following example is based on a link on a page that references the URL `http://host/context/main/MyPage/finance` and a current component called **HelloWorld**:

```
?ss_action=remove&ss_comp=$COMPONENT_ID$&ss_instance=$COMPONENT_INS
TANCE_ID$
```

This example is parsed and converted to:

```
http://host/context/main/MyPage/finance?ss_action=remove&ss_comp=He
lloWorld&ss_instance=123456
```

$COMPONENT_PATH$

Description

Extra path information following the path to the controller servlet that tells the servlet to serve a single **decorated** component.

The $COMPONENT_PATH$ keyword is equivalent to the PortalPathComponentKey context parameter, defined as **component** in the web.xml file for the portal WAR.

Example

```
$PORTAL_URL$/$COMPONENT_PATH$/HelloWorldComponent
```

This example is parsed and converted to:

```
http://host/context/main/component/HelloWorldComponent
```

$context$

Description Used for compatibility with exteNd Director 3.0. $CONTEXT_URL$ should be used in its place going forward.

The $context$ keyword is equivalent to:

```
$SCHEME$://$HOST_PORT$/$CONTEXT_PATH$
```

Or:

```
$CONTEXT_URL$
```

Example

```
$context$/jsp/PropertySheet.jsp
```

This example is parsed and converted to:

```
http://host/context/jsp/PropertySheet.jsp
```

$CONTEXT_PATH$

Description Path to the context on HttpServletRequest.

The $CONTEXT_PATH$ keyword is equivalent to the value returned from HttpServletRequest.getContextPath() excluding the preceding slash.

Example

```
http://host/$CONTEXT_PATH$/
```

If HttpServletRequest.getContextPath() returns the value **Portal**, this example is parsed and converted to:

```
http://host/Portal/
```

$CONTEXT_URL$

Description URL reference to the servlet context.

The $CONTEXT_URL$ keyword is equivalent to:

```
$SCHEME$://$HOST_PORT$/$CONTEXT_PATH$
```

Example

```
$CONTEXT_URL$/jsp/PropertySheet.jsp
```

This example is parsed and converted to:

```
http://host/context/jsp/PropertySheet.jsp
```

$ENCODED_REQUEST_URL$

Description Complete encoded URL reference, including extra path information and query parameters.

The $ENCODE_REQUEST_URL$ keyword is equivalent to the URL-encoded value returned from HttpServletRequest.getRequestURL().

Example `http://host/catalog?callingpage=$ENCODED_REQUEST_URL$`

This example is parsed and converted to:

```
http://host/catalog/callingpage=http%3A%2F%2Flocalhost%2FPortal%2Fmain%2FMyPortal%2FMyProfile
```

$HOST$

Description The host for the HttpServletRequest. Use this keyword when you want to specify a port other than the port on the current request.

The $HOST$ keyword is equivalent to the value returned from HttpServletRequest.getServerName() for the current request.

Example `$SCHEME$://$HOST$`

If the host is **myhost**, this example is parsed and converted to:

```
http://myhost
```

$HOST_PORT$

Description Host and port for the HttpServletRequest.

The $HOST_PORT$ keyword is equivalent to the value returned from HttpServletRequest.getServerName() and HttpServletRequest.getServerPort() for the current request.

If the current port is the default for the scheme specified, then the port is left out.

Example `$SCHEME$://$HOSTP_PORT$`

If the string returned from HttpServletRequest.getServerName() is **myhost** and the string returned from HttpServletRequest.getServerPort() is **9090** for the current request, the example parsed and converted to:

```
http://myhost:9090
```

$PAGE_PATH$

Description Extra path information following the path to the controller servlet that tells the servlet to serve a PID page.

The $PAGE_PATH$ keyword is equivalent to the PortalPathPagesKey context parameter, defined as **pages** in the web.xml file for the portal WAR.

Example `$PORTAL_URL$/$PAGE_PATH$/helloWorldPID.html`

This example is parsed and converted to:

`http://host/context/portal/pages/helloWorldPID.html`

$PORTAL_URL$

Description Absolute path to the portal on the base servlet. It includes the PortalPathEntryPointKey—the main entry point for all portal requests and the key on which the Portal Aggregator listens. This key is defined in web.xml as **portal**.

The $PORTAL_URL$ keyword is equivalent to:

`$CONTEXT_URL$/portal`

Example `$PORTAL_URL$/pages/DirectorHome.html`

This example is parsed and converted to:

`http://host/context/portal/pages/DirectorHome.html`

$PORTLET_PATH$

Description Path to the portlet on the current request.

The $PORTLET_PATH$ keyword is the portlet path information on the base servlet. It includes the PortalPathPortletKey which instructs the portal to serve a single non-decorated portlet. This key is defined in web.xml as **portlet**.

$PORTLET_PATH$ is equivalent to:

`$PORTAL_URL$/portlet`

Example `$PORTLET_PATH$/Header`

This example is parsed and converted to:

`http://host/context/portal/portlet/Header`

$REQUEST_URI$

Description Absolute URI path to the current request

The $REQUEST_URI$ keyword is equivalent to the value returned form req.getRequestURI() on the current request.

Example For the request `http:/localhost/MyPortal/portal/pg/up_MyPage`, $REQUEST_URI$ is parsed and converted to:

```
/MyPortal/portal/pg/up_MyPage
```

$REQUEST_URL$

Description Absolute URL path to the current request (without the query parameters).

The $REQUEST_URL$ keyword is equivalent to the value returned from req.getRequestURL() on the current request.

Example For the request `http:/localhost/MyPortal/portal/pg/up_MyPage`, $REQUEST_URL$ is parsed and converted to:

```
http:/localhost/MyPortal/portal/pg/up_MyPage
```

$RESOURCE_URL$

Description URL reference to the resource path.

The $RESOURCE_URL$ keyword is equivalent to the PortalResourcePath context parameter, defined as **`$CONTEXT_URL$/resource`** in the web.xml file for the portal WAR.

Example

```
$RESOURCE_URL$/portal-theme/Titanium/images/preview.gif
```

This example is parsed and converted to:

```
http://host/context/resource/portal-
theme/Titanium/images/preview.gif
```

$SCHEME$

Description The scheme for the HttpServletRequest.

The $SCHEME$ keyword is equivalent to the value returned from HttpServletRequest.getScheme() for the current request.

Example `$SCHEME$://$HOST$`

If HttpServletRequest.getScheme() returns **http** and the host is **myhost**, this example is parsed and converted to:

```
http://myhost
```

$SERVLET_PATH$

Description Path to the servlet on HttpServletRequest.

The $SERVLET_PATH$ keyword is equivalent to the value returned from HttpServletRequest.getServletPath(), excluding the preceding slash:

Example `$CONTEXT_URL$/$SERVLET_PATH$`

If HttpServletRequest.getServletPath() returns **/custom.jsp**, this example is parsed and converted to:

```
http://host/context/custom.jsp
```

$SERVLET_URL$

Description URL reference to the current servlet path.

The $SERVLET_URL$ keyword is equivalent to:

```
$SCHEME$://$HOST_PORT$/$CONTEXT_PATH$/$SERVLET_PATH$
```

Example `$SERVLET_URL$`

This example may be parsed and converted to:

```
http://host/ExpressPortal/portal/main
```

$THEME_ID$

Description Theme ID of the user's currently selected theme. If no user is logged in or the user does not have a selected theme, the portal's default theme will be used.

The $THEME_ID$ keyword is equivalent to the PortalDefaultTheme context parameter, defined in the web.xml file for the portal WAR.

Example

```
$PORTAL_URL$/resource/portal-theme/$THEMEID$/images/edit.gif
```

If the current theme is Titanium, this example is parsed and converted to:

```
http://host/context/main/resource/portal-theme/Titanium/edit.gif
```

$THEME_PATH$

Description Absolute path to the current user's theme resources in the resource set.

The $THEME_PATH$ keyword is equivalent to:

```
$RESOURCE_URL$/portal-theme
```

Example

```
$THEME_PATH$/Titanium/images/preview.gif
```

This example is parsed and converted to:

```
http://host/context/resource/portal-
theme/Titanium/images/preview.gif
```

$THEME_URL$

Description URL reference to a theme.

The $THEME_URL$ keyword is equivalent to:

```
$SCHEME$://$HOST_PORT$/$CONTEXT_PATH$/$RESOURCE_PATH$/portal-
theme/$THEME_ID$
```

The $THEME_URL$ keyword is also equivalent to:

```
$RESOURCE_URL$/portal-theme/$THEME_ID$
```

Example

```
$THEME_URL$/images/edit.gif
```

Would be parsed and converted to:

```
http://host/context/resource/portal-theme/theme-ID/images/edit.gif
```

24 System Portlets for Portal Pages

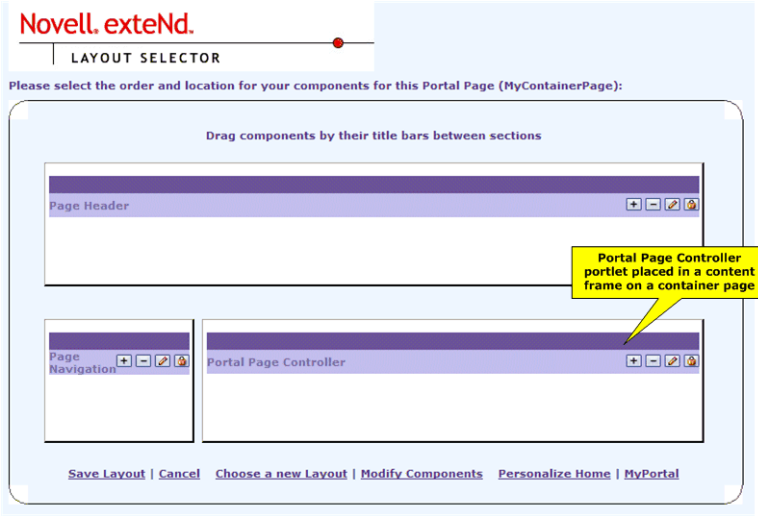
This chapter describes system portlets designed to provide out-of-the-box functionality for portal pages. It covers the following topics:

- Portal Page Controller portlet
- Page Navigation portlet
- Page Header portlet

Portal Page Controller portlet

The **Portal Page Controller** portlet displays the currently selected personal page or shared page at a designated location on a container page. This portlet **must** be added to all container pages.

Typically, the portal administrator places this portlet in a content pane on a container page, as in this example:



In this layout, the Portal Page Controller portlet automatically renders the page selected from the navigation pane by displaying its content and layout in the content pane of the container page, as shown:

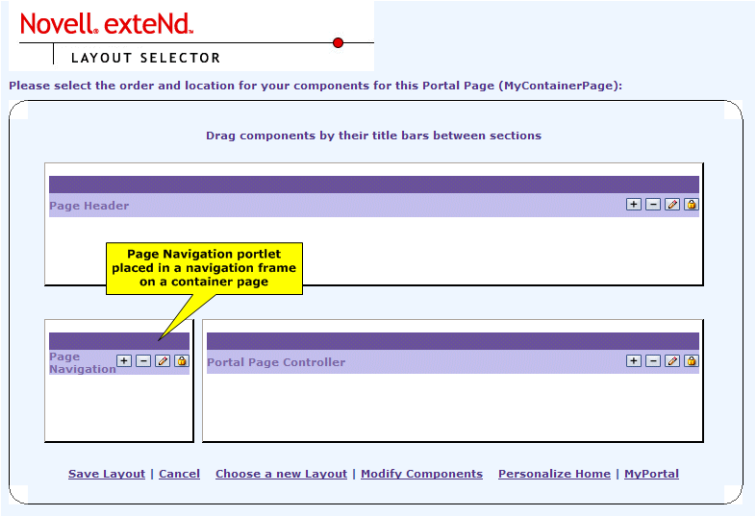
The links in the navigation pane are generated automatically by the Page Navigation portlet, described in “Page Navigation portlet” on page 359.

Portal Page Controller preferences

There are no preferences to set for the Portal Page Controller portlet.

Page Navigation portlet

The **Page Navigation** portlet provides a graphical user interface that lets users easily navigate to all container, shared, and personal pages that they are authorized to access.Typically, the portal administrator places this portlet in a navigation pane on a container page, as in this example:



The Navigation portlet automatically creates a set of links to the pages available to the logged-in user. In the following example, the Navigation portlet presents the container, shared, and personal pages that are available to the logged-in user:

Page Navigation portlet preferences

You can configure the following preferences of Navigation portlets

Preference	Description	What to specify
width	Width of portlet content display	A number of pixels or percentage
layout	HTML layout for this portlet	HTML content or a scoped path pointing to HTML content
tooltip	Display tooltips for page links	**true** or **false**
navigation-justification	Justification of navigation text	**none**, **left**, **right**, or **center**

Preference	Description	What to specify
listmaxrows	Maximum number of displayed rows when navigation type is set to **List**	Integer
containerpages-show	Display links to container pages	**true** or **false**
containerpages-title	Text to display as a title for container page links	Text string
containerpages-navigation-type	Format for presenting links to container pages	One of these constants: ◆ **simpleRow** ◆ **simpleColumn** ◆ **simpleTree** ◆ **list** ◆ **menu** ◆ **hierVMenu** ◆ **hierHMenu** 📖 For more information, see "Navigation type formats" on page 363.
containerpages-urlmode	Method used for navigation: regular HTML links or use of an invisible HTML form	**query(invisible form)** or **path(link)**
containerpages-sorting	Order in which to display links to container pages	**ascending** or **descending**
containerpages-sortmode	Field used for sorting container page's name or priority	**alphabetical** or **priority**
containerpages-categories	List of page categories to filter the list of container pages. **Uncategorized** filters uncategorized container pages.	List of categories (text strings)

Preference	Description	What to specify
sharedpages-show	Display links to shared pages?	**true** or **false**
sharedpages-title	Text to display as a title for shared page links	Text string
sharedpages-navigation-type	Format for presenting links to shared pages	One of these constants: ◆ **simpleRow** ◆ **simpleColumn** ◆ **simpleTree** ◆ **list** ◆ **menu** ◆ **hierVMenu** ◆ **hierHMenu** 📖 For more information, see "Navigation type formats" on page 363.
sharedpages-urlmode	Method used for navigation: regular HTML links or use of an invisible HTML form	**query(invisible form)** or **path(link)**
sharedpages-sorting	Order in which to display links to shared pages	**ascending** or **descending**
sharedpages-sortmode	Field used for sorting shared page's name or priority	**alphabetical** or **priority**
sharedpages-categories	List of page categories to filter the list of shared pages. **Uncategorized** filters uncategorized pages	List of categories (Text strings)
userpages-show	Display links to personal pages?	**true** or **false**
userpages-title	Text to display as a title for personal page links	Text string

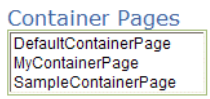
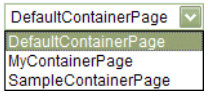
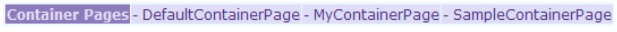
Preference	Description	What to specify
userpages-navigation-type	Format for presenting links to personal pages	One of these constants: ◆ **simpleRow** ◆ **simpleColumn** ◆ **simpleTree** ◆ **list** ◆ **menu** ◆ **hierVMenu** ◆ **hierHMenu** 📖 For more information, see "Navigation type formats" on page 363.
userpages-urlmode	Method used for navigation: regular HTML links or use of an invisible HTML form	**query(invisible form)** or **path(link)**
userpages-sorting	Order in which to display links to personal pages	**ascending** or **descending**
userpages-sortmode	Field used for sorting personal page's name or priority	**alphabetical** or **priority**
quicklinks-show	Display quicklinks?	**true** or **false**
quicklinks-title	Text to display as a title for quicklinks	Text string

Preference	Description	What to specify
quicklinks-navigation-type	Format for presenting quicklinks	One of these constants: ◆ **simpleRow** ◆ **simpleColumn** ◆ **simpleTree** ◆ **list** ◆ **menu** ◆ **hierVMenu** ◆ **hierHMenu** 📖 For more information, see "Navigation type formats" on page 363.
quicklinks	List of links, external or internal	**Syntax:** label~url **Example:** Novell~http://www.novell.com
personalize-display	Display link to Portal Personalizer?	**true** or **false**
portalpageadmin-display	Display link to Portal Administration page?	**true** or **false**
login-logout display	Display link to portal login or logout?	**true** or **false**
quicklink-display	Display quick links?	**true** or **false**

Navigation type formats

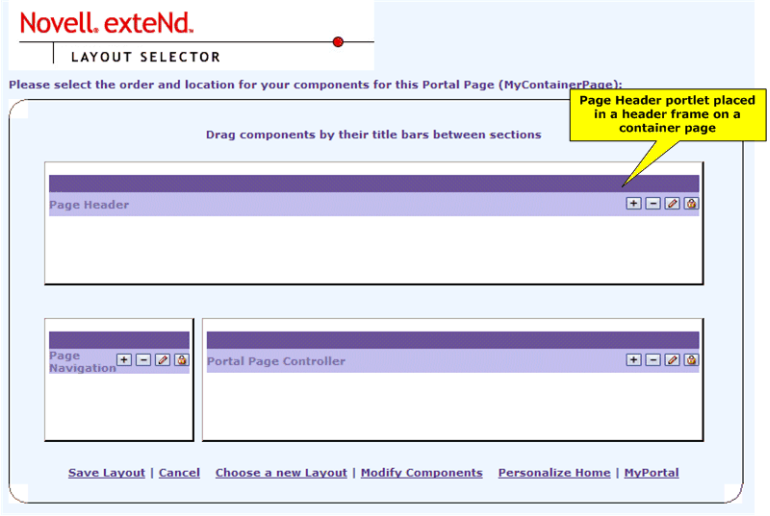
The following table shows each format that you can specify for **containerpages-navigation-type**, **sharedpages-navigation-type**, and **userpages-navigation-type** preferences in the Navigation portlet:

Navigation type	Preview
simpleRow	Container Pages : DefaultContainerPage MyContainerPage SampleContainerPage

Navigation type	Preview
simpleColumn	
simpleTree	
list	
menu	
hierVMenu	
hierHMenu	

Page Header portlet

The **Page Header** portlet provides a mechanism for creating a customized look and feel for Web pages in an organization. Typically, the portal administrator places this portlet in a header pane on a container page to establish a corporate identity.



Page Header portlet preferences

You can configure the following preferences for Header portlets:

Preference	Description	What to specify
width	Width of portlet content display	A number of pixels or a percentage
logo-image	Logo to be displayed in the header	URL
background-image	Background image for the header	URL
background-repeat	Repeat pattern for the background image	**no-repeat** **repeat-x** **repeat-y** **repeat**

Preference	Description	What to specify
background-position	Position of the background image	**top**, **middle**, **bottom** OR **left**, **center**, **right**
layout	HTML layout for this portlet	HTML content or a scoped path pointing to HTML content
sitename	Portal main title	Text string
salutation	Welcoming words to precede the username display in the header	Text string
anonymous-salutation	Welcoming words to anonymous users	Text string
userformat-html	HTML layout to define the display pattern for the username	HTML content with uses of User scoped path
date-display	Display today's date?	**true** or **false**
date-style	Date format patterns	Date format See Format scoped path for available patterns
personalize-display	Display link to Portal Personalizer?	**true** or **false**
portalpageadmin-display	Display link to Portal Administration page?	**true** or **false**
login-logout display	Display link to portal login or logout?	**true** or **false**
Tooltip	Display tooltips for page links?	**true** or **false**
navigation-justification	Justification of navigation text	**none**, **left**, **right**, or **center**
listmaxrows	Maximum number of displayed rows when navigation is set to **List**	Integer
containerpages-show	Display links to container pages?	**true** or **false**

Preference	Description	What to specify
containerpages-title	Text to display as a title for container page links	Text string
containerpages-navigation-type	Format for presenting links to container pages	One of these constants: One of these constants: ◆ **simpleRow** ◆ **simpleColumn** ◆ **simpleTree** ◆ **list** ◆ **menu** ◆ **hierVMenu** ◆ **hierHMenu** 📖 For more information, see "Navigation type formats" on page 363.
containerpages-urlmode	Method used for navigation: regular HTML links or use of an invisible form	**query(invisible form)** OR **path(link)**
containerpages-sorting	Order in which to display links to container pages	**ascending** OR **descending**
containerpages-sortmode	Field used for sorting container page's name or priority	**alphabetical** OR **priority**
containerpages-categories	List of page categories to filter the list of container pages. **Uncategorized** filters uncategorized container pages.	List of categories (text strings)
sharedpages-show	Display links to shared pages?	**true** or **false**
sharedpages-title	Text to display as a title for shared page links	Text string

Preference	Description	What to specify
shared pages-navigation-type	Format for presenting links to shared pages	One of these constants: ◆ **simpleRow** ◆ **simpleColumn** ◆ **simpleTree** ◆ **list** ◆ **menu** ◆ **hierVMenu** ◆ **hierHMenu** 📖 For more information, see "Navigation type formats" on page 363.
sharedpages-urlmode	Method used for navigation: regular HTML links or use of an invisible HTML form	**query(invisible form)** OR **path(link)**
sharedpages-sorting	Order in which to display links to shared pages	**ascending** or **descending**
sharedpages-sortmode	Field used for sorting shared page's: name or priority	**alphabetical** or **priority**
sharedpages-categories	List of page categories to filter the list of shared pages. **Uncategorized** filters uncategorized pages	List of categories (Text strings)
userpages-show	Display links to personal pages?	**true** or **false**
userpages-title	Text to display as a title for personal page links	Text string

Preference	Description	What to specify
userpages-navigation-type	Format for presenting links to personal pages	One of these constants: ◆ **simpleRow** ◆ **simpleColumn** ◆ **simpleTree** ◆ **list** ◆ **menu** ◆ **hierVMenu** ◆ **hierHMenu** 📖 For more information, see "Navigation type formats" on page 363.
userpages-urlmode	Method used for navigation: regular HTML links or use of an invisible HTML form	**query(invisible form)** or **path(link)**
userpages-sorting	Order in which to display links to personal pages	**ascending** or **descending**
userpages-sortmode	Field used for sorting personal page's name or priority	**alphabetical** or **priority**
quicklinks-show	Display quicklinks?	**true** or **false**
quicklinks-title	Text to display as a title for quicklinks	Text string

Preference	Description	What to specify
quicklinks-navigation-type	Format for presenting quicklinks	One of these constants: ◆ **simpleRow** ◆ **simpleColumn** ◆ **simpleTree** ◆ **list** ◆ **menu** ◆ **hierVMenu** ◆ **hierHMenu** ◆ 🕮 For more information, see "Navigation type formats" on page 363.
quicklinks	List of links, external, or internal	**Syntax**: label~URl **Example**: Novell~http://www.novell.com

Index

Symbols

A

C

D

E

F

G

H

J

L

M

N

O

P

R

S

T

U

W

X