



BEA WebLogic Portal™

Customizing the Portal Administration Console

Copyright

Copyright © 1995-2007 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

This software is protected by copyright, and may be protected by patent laws. No copying or other use of this software is permitted unless you have entered into a license agreement with BEA authorizing such use. This document is protected by copyright and may not be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form, in whole or in part, without prior consent, in writing, from BEA Systems, Inc.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE DOCUMENTATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA SYSTEMS DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE DOCUMENT IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Trademarks and Service Marks

Copyright © 1995-2007 BEA Systems, Inc. All Rights Reserved. BEA, BEA WebLogic Portal, BEA WebLogic Server, BEA WebLogic Workshop, BEA Workshop for WebLogic Platform, BEA WebLogic RFID Mobile SDK, Built on BEA, Jolt, JoltBeans, SteelThread, Top End, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA AquaLogic, BEA AquaLogic Data Services Platform, BEA AquaLogic Enterprise Security, BEA AquaLogic Interaction, BEA AquaLogic Interaction Analytics, BEA AquaLogic Interaction Collaboration, BEA AquaLogic Interaction Integration Services, BEA AquaLogic Interaction Process, BEA AquaLogic Interaction Publisher, BEA AquaLogic Interaction Studio, BEA AquaLogic Service Bus, BEA AquaLogic Service Registry, BEA AquaLogic BPM Designer, BEA AquaLogic BPM Studio, BEA AquaLogic BPM Enterprise Server – Standalone, BEA AquaLogic BPM Enterprise Server – BEA WebLogic, BEA AquaLogic BPM Enterprise Server – IBM WebSphere, BEA AquaLogic BPM Enterprise Server – JBoss, BEA AquaLogic BPM Process Analyzer, BEA AquaLogic Interaction Development Kit, BEA AquaLogic Interaction JSR-168 Consumer, BEA AquaLogic Interaction Identity Service – Active Directory, BEA AquaLogic Interaction Identity Service – LDAP, BEA AquaLogic Interaction Content Service – Microsoft Exchange, BEA AquaLogic Interaction Content Service – Lotus Notes, BEA AquaLogic Interaction Logging Utilities, BEA AquaLogic Interaction WSRP Consumer, BEA AquaLogic Interaction Portlet Framework – Microsoft Excel, BEA AquaLogic Interaction .NET Application Accelerator, AquaLogic Interaction Content Service – Documentum, BEA AquaLogic Interaction Content Service – Windows Files, BEA AquaLogic Interaction Portlet Suite – IMAP, BEA AquaLogic Interaction Portlet Suite – Lotus Notes, BEA AquaLogic Interaction Portlet Suite – Exchange, BEA AquaLogic Interaction Portlet Suite – Documentum, BEA AquaLogic Interaction IDK Extension, BEA AquaLogic HiPer Workspace for BPM, BEA AquaLogic HiPer Workspace for Retail, BEA AquaLogic Sharepoint Console, BEA AquaLogic Commerce Services, BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Kodo, BEA Liquid Data for WebLogic, BEA Manager, BEA MessageQ, BEA SALT, BEA Service Architecture Leveraging Tuxedo, BEA WebLogic Commerce Server, BEA WebLogic Communications Platform, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Enterprise Security, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Java Adapter for Mainframe, BEA WebLogic JDriver, BEA WebLogic Log Central, BEA WebLogic Mobility Server, BEA WebLogic Network Gatekeeper, BEA WebLogic Personalization Server, BEA WebLogic Personal Messaging API, BEA WebLogic Platform, BEA WebLogic Portlets for Groupware Integration, BEA WebLogic Real Time, BEA WebLogic RFID Compliance Express, BEA WebLogic RFID Edge Server, BEA WebLogic RFID Enterprise Server, BEA WebLogic Server Process Edition, WebLogic Server Virtual Edition, WebLogic Liquid Operations Control, BEA LiquidVM, BEA WebLogic SIP Server, BEA WebLogic WorkGroup Edition, BEA Workshop for WebLogic Platform, BEA Workshop for JSF, BEA Workshop for JSP, BEA Workshop for Struts, BEA Workshop Studio, Dev2Dev, Liquid Computing, and Think Liquid are trademarks of BEA Systems, Inc. Accelerated Knowledge Transfer, AKT, BEA Mission Critical Support, BEA Mission Critical Support Continuum, CollabraSuite – BEA Edition, BEA Guardian and BEA SOA Self Assessment are service marks of BEA Systems, Inc. All other names and marks are property of their respective owners.

Contents

1. Introduction to Portal Administration Console

| | |
|---------------------------------------|-----|
| Document Scope and Audience | 1-2 |
| Getting Started | 1-2 |
| In This Guide | 1-2 |

2. Re-using Administration Console Portlets in Your Portal

| | |
|-----------------------------------------------------------------------------|-----|
| Overview of the Administration Console Portlets | 2-1 |
| Adding Administration Console Portlets to Your Portal Application | 2-3 |
| Adding Administration Console Resources to Your Web Project | 2-4 |
| Using Portlet Resource IDs to Generate Events | 2-5 |

3. Extending the Portal Administration Console

| | |
|----------------------------------------------------------------|------|
| What Is an Administration Console Extension? | 3-1 |
| Setting up Your Development Environment | 3-2 |
| Creating an Administration Console Web Project | 3-2 |
| Creating your Extension Web Project | 3-3 |
| Building Your Extension Module | 3-4 |
| Creating Configuration Files for Your Extension | 3-4 |
| Adding Your Extension to Your Administration Console | 3-12 |
| Using your Extended Administration Console | 3-13 |

Introduction to Portal Administration Console

The WebLogic Portal Administration Console is the tool that portal administrators use to control the behavior, content, and appearance of portals, and to perform many traditional system administration activities such as user management and security management.

The WebLogic Portal Administration Console is organized according to the following categories of tasks:

- **Portal Management** – Portals, desktops, books, pages, portlets, and other portal resources.
- **User, Groups, & Roles** – User and group management, security provider configuration, Delegated Administration, and Visitor Entitlements.
- **Configuration Settings** – Server settings for Cache Management, Server Maintenance Mode, Personalization, Security, Unified User Profiles, and WSRP.
- **Interaction Management** – Campaigns, placeholders, user segments, and content selectors.
- **Content Management** – Content and repositories.

The Administration Console is a portal application and is built with the portal framework. Using this framework, you can customize the Administration Console by changing the look and feel of your Administration Console, adding new portlets (extending) to the Administration Console, or re-using Administration Console portlets within a separate portal application.

This chapter includes the following sections:

- [Document Scope and Audience](#)

- [Getting Started](#)

Document Scope and Audience

This document is a resource for software developers who want to customize how the Portal Administration Console is used. It is assumed that the reader is already familiar with using WebLogic Portal to develop portal applications.

Getting Started

Before you begin working with the Portal Administration Console, you should be familiar with working with WebLogic Portal applications. WebLogic Portal provides specific documentation on developing portals, portlets, and deploying portal applications. It is recommended that you become familiar with the following guides and their tasks before customizing your Administration Console.

- [Portal Development Guide](#)
- [Portlet Development Guide](#)
- [Production Operations Guide](#)

In This Guide

This guide includes the following chapters:

- [Chapter 2, “Re-using Administration Console Portlets in Your Portal”](#)
- [Chapter 3, “Extending the Portal Administration Console”](#)

Re-using Administration Console Portlets in Your Portal

You can re-use portlets from the Administration Console within your portal. Doing this allows you to access portal resources directly from your portal, such as content, user and group information, or campaign settings.

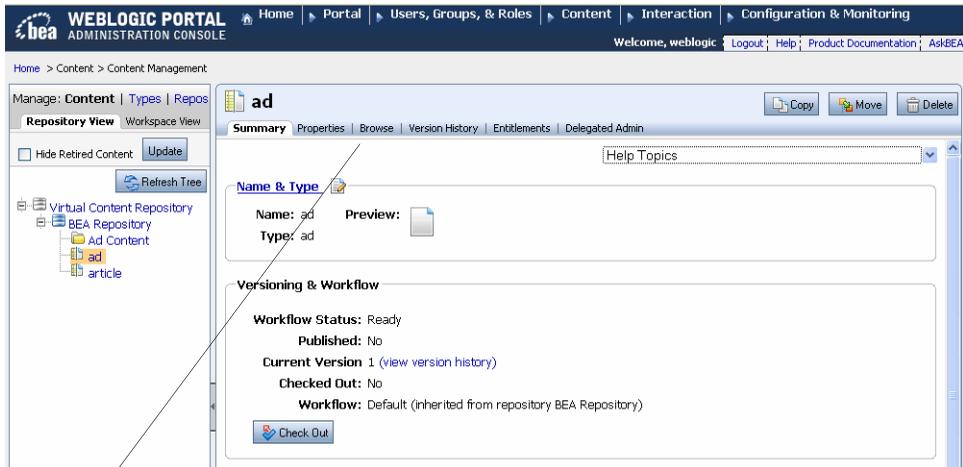
This chapter includes the following topics:

- [Overview of the Administration Console Portlets](#)
- [Adding Administration Console Portlets to Your Portal Application](#)

Overview of the Administration Console Portlets

The Administration Console is comprised of many portlets that access respective resources in different ways. Each tab within the Administration Console is a page containing a portlet, see [Figure 2-1](#) for an illustration.

Figure 2-1 Administration Console Tabs



Each tab within the Administration Console (Summary, Properties, Browse, and so on) is represented by a page containing a portlet.

You can re-use these portlets in your portal application if you want to access portal resources without accessing the Administration Console. Portlets associated with the following portal resources are available to add to your portal:

- Interaction Management (campaigns, content selectors, placeholders, and user segments)
- Content Management
- User and Group Management

Note: Portal Management, Delegated Administration, Entitlements, and Service Administration portlets cannot be re-used in portal applications.

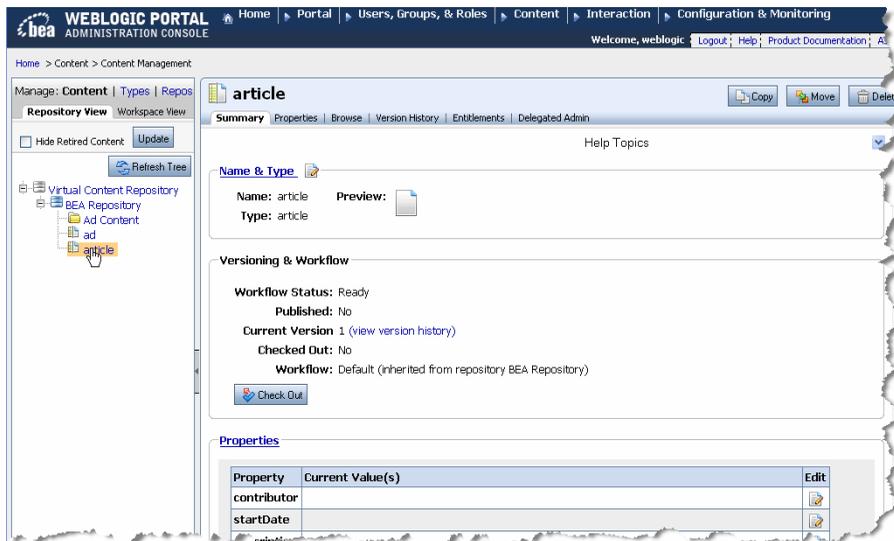
Portlets within the Administration Console react to tree events. For example, when you use the Content Resource tree to select content, the content Summary tab (the content summary portlet) is activated, see [Figure 2-2](#).

However, within your portal, resource IDs are used in inter-portlet communication events to activate Administration Console portlets. If another portlet refers to the resource ID that is used by a Administration Console portlet, that portlet is activated.

Note: Administration Console tree portlets (the Portal Resource tree, the Content Resource tree, and so on) cannot be re-used in your portal application. If you want to use entire

Administration Console functionality such as the Content Management Resource tree as well as its portlets, BEA recommends using the Administration Console rather than adding an Administration Console portlet to your portal.

Figure 2-2 Selecting Article in the Repository tree Calls the Summary Portlet



For example, you can use a combination of the Content Presenter portlet and the edit content portlet from the Administration Console to allow users to modify content on the fly. By adding an “Edit Content” button within the content presenter portlet that activates the edit content portlet.

WebLogic Portal provides specific APIs for working with Administration Console portlets that provides an interface-based framework that integrates with inter-portlet communication events. Instead of using events and event handlers explicitly, Administration Console portlets generate and receive events using resource IDS and this interface-based framework.

Adding Administration Console Portlets to Your Portal Application

Before you can add any administration console portlets to your portal, you must first add the respective J2EE libraries to your web project. These library modules provide the common framework and the respective portlets that you can use.

After ensuring your web project includes the library modules you need, you can add administration portlets to your portal

Administration Console portlets require communication from another portlet in order to be activated. You need to build a portlet that will be used to activate or call an Administration Console portlet. Activating an Administration Console portlet is done by using resource IDs. You need to include code in the primary portlet that generates an event using a resource ID that the Administration Console portlet recognizes.

This section includes the following topics:

- [Adding Administration Console Resources to Your Web Project](#)
- [Using Portlet Resource IDs to Generate Events](#)

Adding Administration Console Resources to Your Web Project

Administration Console portlets are included in respective library modules that you must add to your web project. Copy the J2EE modules that you need. The wlp-tools-common-web-lib and wlp-tools-framework-web-lib libraries are required for all portlets.

To add administration console J2EE libraries:

1. Right-click your web project folder and select **Build Path > Add Libraries**.
2. In the Add Library dialog, select **WebLogic J2EE library**. Click **Next**.
3. Click **Add...** and browse to the respective library module and click **OK**.
4. Click **Finish**.
5. Repeat for each of the modules listed in [Table 2-1](#).

Table 2-1 J2EE Library Modules for Administration Console Portlets

| Library Module | Portlets | Description |
|-----------------------------|----------|-------------------------------------------------------------------------------|
| wlp-tools-common-web-lib | All | Contains the common framework needed for all administration console portlets. |
| wlp-tools-framework-web-lib | All | Contains the common framework needed for all administration console portlets. |

Table 2-1 J2EE Library Modules for Administration Console Portlets

| Library Module | Portlets | Description |
|---------------------------|-----------------------------------------------------------------|-----------------------------------------------------|
| wlp-tools-ugm-web-lib | User Management Group Management | Contains the portlets for user and group management |
| wlp-tools-content-web-lib | Content Management | Contains all content management portlets. |
| wlp-tools-im-web-lib | Campaigns Content Selectors Placeholders User Segments | Contains all interaction management portlets. |

Using Portlet Resource IDs to Generate Events

Administration Console portlets are designed to react to inter-portlet communication events that are generated by resource IDs. When a portal resource such as content or a user name is entered or selected in a portlet, the respective Administration Console portlet is activated.

For example, you can allow portal users to edit content displayed in a portlet by adding code in that portlet that activates the Edit Content Properties portlet, see [Listing 2-1](#) for an example.

The following WebLogic APIs are used to interact with or “activate” Administration Console portlets:

- [ResourceID](#)
- [PortletORB](#)

Listing 2-1 Code Snippet Showing How to Trigger a Resource ID Event

```
// Create a resource ID (see Javadoc)
final ResourceID resourceID = ... ;

// Get a TreeSelectListener
final HttpServletRequest request = ...;
```

Re-using Administration Console Portlets in Your Portal

```
final HttpServletResponse response = ...;
final PortletORB portletORB = PortletORB.getPortletORB(request, response);
final PortletHandle portletHandle = portletORB.getBroadcastPortletHandle();
final TreeSelectListener listener = portletHandle.
    narrow(TreeSelectListener.class);

// Call selected(...)
final ResourcePath resourcePath = new ResourcePath(new ResourceID[] { resourceID
});
final ResourceType resourceType = resourceID.getResourceType();
listener.selected(resourcePath, resourceType, resourceID);
```

Extending the Portal Administration Console

Administration Console extensions are web applications that you integrate with the existing Administration Console. You can use an extension to do something as simple as add a portlet to a page within the console, or add an entire editor module that includes multiple books and pages.

To create an extension, you re-create the Administration Console web application within your EAR and import the associated libraries., you can add or remove functionality to the web application.

After creating your custom Administration Console, including any extensions you added, you package this new custom Administration Console as a J2EE library and add that library to your existing web application, replacing the Administration Console that was previously associated with your portal.

The section includes the following topics:

- [What Is an Administration Console Extension?](#)
- [Setting up Your Development Environment](#)
- [Setting up Your Development Environment](#)
- [Building Your Extension Module](#)

What Is an Administration Console Extension?

You can extend the Administration Console by adding books, pages or portlets to an existing book or page within the Administration Console. For example, you can add a login portlet to the

Administration Console that allows users to login to another administrative application, such as a Network Operations Center (NOC).

When you extend the Portal Administration Console, you need to first assemble the application files that are needed in the extension (new books, pages, JSPs, portlets, and so on), and then configure your extension so that the Portal Administration Console can locate and communicate with its resources.

The simplest extension adds content to the Administration Console's home page (desktop). The JAR file for such an extension contains:

- A NetUI Extension XML file that describes the location in the UI in which you want your extension to display.
- An editors XML file that provides the metadata used to include your extension in the Administration Console menu.
- A book or page that contains the content you want to display (and all of its contents).

Setting up Your Development Environment

Setting up your development environment involves creating two additional web projects within your EAR. One project will be used to re-create the Administration Console with all of its libraries. This project will become your new extended Administration Console. You use an additional web project to create your extension.

Creating an Administration Console Web Project

To create your custom Administration Console web project:

1. Create a web project by right-clicking your EAR project and choosing **New > Portal Web Project**.
2. In the New Portal Web Project dialog, associate your web project with the same EAR that your portal web application is associated with.
3. In the Project Facets dialog, de-select the Portal Visitor Tools facet.
4. Click **Finish**.

Add the Administration Console J2EE Libraries to your Extension Project

The following J2EE modules are required for the Administration Console. Even though you may be customizing only a page of the console, import ALL of these modules to your web project. Remember you are re-creating the entire Administration Console to ensure that your deployment remains consistent.

Note: J2EE libraries not imported to your web project can be overwritten when you upgrade WebLogic Portal. In most cases, this is desirable. However, upgraded libraries will delete your Administration Console customizations and extensions.

To copy a J2EE module to your web project:

1. In the Project Explorer, right-click the `//web_project_name/WebLogic Deployment Descriptor > J2EE Libraries` folder and select **Add Multiple**.
2. In the Select WebLogic J2EE library, select the modules you need. [Figure 3-1](#) lists the required libraries.

Table 3-1 Required Administration Console J2EE Libraries

| Required Administration Console J2EE Libraries |
|------------------------------------------------|
| wlp-tools-common-web-lib |
| wlp-tools-content-web-lib |
| wlp-tools-framework-web-lib |
| wlp-tools-im-web-lib |
| wlp-tools-portal-web-lib |
| wlp-tools-serviceadmin-web-lib |
| wlp-tools-ugm-web-lib |

3. Click **OK**.

Creating your Extension Web Project

Create a web project where you will build your extension. Your extension can include books, pages, portlets, or an entire portal.

To create your extension web project:

1. Create a web project by right-clicking your EAR project and choosing **New > Portal Web Project**.
2. In the New Portal Web Project dialog, associate your web project with the same EAR that your portal web application is associated with.
3. In the Project Facets dialog, de-select the Portal Visitor Tools facet.
4. Click **Finish**.
5. Create the books, pages, and other resources that you want to include in your extension. For more information about building portal resources, see the [Portal Development Guide](#).

Building Your Extension Module

When you are finished creating the resources for your extension, you are ready to create the configuration files necessary to integrate your extension with the Administration Console project you created. You need to create two configuration files to include with your project, package the project and then add it to your Administration Console web project.

Creating Configuration Files for Your Extension

When you create an extension, you are creating a new web application that can be used in your Administration Console via a `netuix-extension.xml` file that is contained within the extension application itself. If you want your extension to be accessible via the main menu of the Administration Console, you also need to create an `editors.xml` file.

- The `netuix-extension.xml` file defines target locations within the existing Administration Console (via definition labels) where these new portlets, books and pages should be added.
- The `editors.xml` file defines which Administration Console menu to use (Portal Management, Content Management, and so on) to access your extension.

Note: The `beehive-netui-config.xml` file in `//WEB-INF/` directory specifies netui-specific tags and handler classes. Add the following elements to the `beehive-netui-config.xml` file between the `<pageflow-config>` and the `<request-interceptors>` elements:

```
<jsp-tag-config>
    <id-javascript>legacy</id-javascript>
```

```
<tree-renderer-class>com.bea.jsptools.patterns.tree.  
ToolsTreeRenderer</tree-renderer-class>  
  
</jsp-tag-config>
```

Determining Where to Place Your Extension

The Administration Console is built out of several library modules which represent the features supported within the console. The main portal of the Administration Console resides in the `wlp-tools-common-web-lib`. This main `portal.portal` file is extended by each subsequent library module. To extend the portal, each library uses an `netuix-extension.xml` that references its resources and indicates where it will extend the `portal.portal`.

You can extend the Administration Console at the book or page level. To determine which book to extend, you use the `netuix-extension.xml` files for each library module to familiarize yourself with how the Administration Console is comprised.

After determining the Administration Console book or page you want to extend, you refer to the definition labels of the respective module's books or pages in your own extension module's `netuix-extension.xml` to indicate where your extension should appear within the Administration Console.

To view these definition labels, you open the respective `*.war` files that contain the associated books for your extension point. Keep in mind that you do NOT modify the existing Administration Console files or modules in any way. Instead, you create an extension that points to the existing files. [Table 3-2](#) lists the books that contain definition labels of commonly used extension points.

Table 3-2 Common Extensions for the Administration Console

| Book | Extension Point | netuix-extension.xml Location |
|--------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------|
| /im/interactionToolsMainBook.book | <p>The main book for Interaction Management.</p> <p>Look in this book's XML file for the definition labels of subsequent books that you want to extend.</p> | <p>//WebLogic_Home/portal/lib/modules/wlp-im-web-lib.war</p> |
| /ugm/TopLevelEditor.book | <p>The main book for User and Group Management.</p> <p>Look in this book's XML file for the definition labels of subsequent books that you want to extend.</p> | <p>//WebLogic_Home/portal/lib/modules/wlp-ugm-web-lib.war</p> |
| /portalTools/portalTopLevelEditorBook.book | <p>The main book for Portal Management.</p> <p>Look in this book's XML file for the definition labels of subsequent books that you want to extend.</p> | <p>//WebLogic_Home/portal/lib/modules/wlp-pm-web-lib.war</p> |

Table 3-2 Common Extensions for the Administration Console

| Book | Extension Point | netuix-extension.xml Location |
|-----------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|
| /serverTools/serverToolsTopLevelEditorBook.book | The main book for Service Administration. Look in this book's XML file for the definition labels of subsequent books that you want to extend. | //WebLogic_Home/portal/lib/modules/wlp-serviceadmin-web-lib.war |
| /content/contentTopLevelEditorBook_twoPageTree.book | The main book for Content Management when using library services. Look in this book's XML file for the definition labels of subsequent books that you want to extend. | //WebLogic_Home/portal/lib/modules/wlp-cm-web-lib.war |
| /content/contentTopLevelEditorBook_onePageTree.book | The main book for content management if not using library services. Look in this book's XML file for the definition labels of subsequent books that you want to extend. | //WebLogic_Home/portal/lib/modules/wlp-cm-web-lib.war |

Creating a netuix-extension.xml File

The easiest way to create a new `netuix-extension.xml` file to include in your extension application is to copy an existing file from one of the `*.war` files included in a library module. The `netuix-extension.xml` file must reside in the `//WEB-INF/` directory of your project.

Before beginning these steps, copy the text from an existing `netuix-extension.xml` to use in your new file. [Listing 3-1](#) provides an example file.

Listing 3-1 Example of a netuix-extension File that Extends the Main Book of the Administration Console

```
<?xml version="1.0" encoding="UTF-8" ?>

<weblogic-portal-extension
xmlns="http://www.bea.com/servers/portal/weblogic-portal/8.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.bea.com/servers/portal/weblogic-portal/8.0
netuix-extension-1_0_0.xsd">

  <!-- The provider-info element allows you to enter a description for your
extension -->

  <provider-info>

    <title>BEA WebLogic Portal Administration Console - Content Management
Extension</title>

    <description>Content Management books that are inserted into theBEA
WebLogic Portal Administration Console</description>

    <author>BEA Systems, Inc.</author>

  </provider-info>

  <! -- The portal-file element defines which portal file you are extending. To
extend the Administration Console, the value of this element should be
portal.portal -->

  <portal-file>/portal.portal</portal-file>

  <! -- Use the book-extension element and its children to define which book in
the Adminstration Console you are extending. The <book-content content-uri>
element is wehre you put that path to your extension. In the following example,
content is the name of the web extension application that contains the
contentTopLevelEditorBook-twoPageTree.book -->

  <book-extension>
```

```

<book-location>
    <parent-label-location label="mainToolsBook" />
    <book-insertion-point page-label="homeBook" action="append" />
</book-location>

<book-content
content-uri="/content/contentTopLevelEditorBook_twoPageTree.book" />
</book-extension>

<!-- This example file includes two extensions. Both of which extend the home
book of the mainToolsBook of the Administration Console -->

<book-extension>
    <book-location>
        <parent-label-location label="mainToolsBook" />
        <book-insertion-point page-label="homeBook" action="append" />
    </book-location>

    <book-content
content-uri="/content/contentTopLevelEditorBook_onePageTree.book" />
</book-extension>

</weblogic-portal-extension>

```

To create a `netuix-extension.xml` file within your web project:

1. Right-click the `//WEB-INF` folder within your extension web project and select **New > Other**.
2. In the Select a Wizard dialog, select **XML > XML** and click **Next**.
3. In the Create an XML File dialog, mark **Create XML File from Scratch** and click **Next**.
4. In the XML File Name dialog, type `netuix-extension` as the name of your file and click **Finish**.
5. In the Package Explorer, right-click the `netuix-extension.xml` file and select **Open With > Text Editor**.
6. Paste the text you have copied from a valid `netuix-extension.xml` file.

7. Make modifications to the text that match the extension module you have created and save the file.

Creating an editors.xml File

If you want to include your extension in the header menu of the Administration Console (shown in [Listing 3-1](#)), you need to include an `editors.xml` file in your extension web project.

Figure 3-1 The Administration Console Header Menu



The `editors.xml` file allows you to extend an existing menu entry or add a new menu entry.

The easiest way to create a new `editors.xml` file to include in your extension application is to copy an existing file from one of the `*.war` files included in a library module. The `editors.xml` file must reside in the `//WEB-INF/` directory of your project.

Note: For WebLogic Portal 10.0, if you want to add an editor to an existing menu, you need to add a properties file to your system path. See the [WebLogic Portal Release Notes](#) for more information.

Before beginning these steps, copy the text from an existing `editors.xml` to use in your new file. [Listing 3-1](#) provides an example file.

To create an `editors.xml` file:

1. Right-click the `//WEB-INF` folder within your extension web project and select **New > Other**.
2. In the Select a Wizard dialog, select **XML > XML** and click **Next**.
3. In the Create an XML File dialog, mark **Create XML File from Scratch** and click **Next**.
4. In the XML File Name dialog, type `editors` as the name of your file and click **Finish**.
5. In the Package Explorer, right-click the `editors.xml` file and select **Open With > Text Editor**.
6. Paste the text you have copied from a valid `editors.xml` file.
7. Make modifications to the text that match the extension module you have created and save the file. [Listing 3-2](#) shows an example of a valid `editors.xml` file.

Listing 3-2 Example of an editors.xml File

```

<?xml version="1.0" encoding="UTF-8"?>

<!-- @name: the name of the top-level menu item. This is used as the key for
getting the title from the properties file specified in the bundleName attribute.
-->

<!-- @defLabel: the definition label of the default book to show for this editor
group. This is not currently used by the menu. -->

<!-- @defaultEditor: the name attribute of the default editor for this group -->

<!-- @bundleName: the name of the applications scoped (EAR) properties file from
which to retrieve the title for @name -->

<edtrs:editor-group

    xmlns:edtrs="http://com.bea.portal/tools/admin/xsd/editors-1_0_0"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="http://com.bea.portal/tools/admin/xsd/editors-1_0_0
editors-1_0_0.xsd"

    name="myPAT"

    defLabel="NOT_USED"

    defaultEditor="myPATeditor"

    bundleName="myPATeditor.i18n.myPATeditor" >

<!-- @name: the name of this editor which is used as a second level menu item.
This is used as the key for getting the title from the properties file specified
in the bundleName attribute. -->

<!-- @landingDefLabel: the definition label of the landing page for this editor;
IT MUST END WITH _landingPage -->

<!-- @headerIcon: the icon to show for this editor -->

<!-- @bundleName: the name of the applications scoped (EAR) properties file from
which to retrieve the title for @name -->

    <edtrs:editor name="myPATeditor"

        landingDefLabel="myPATeditor_landingPage"

        headerIcon="wlp-content-mgmt-24.gif"

        bundleName="myPATeditor.i18n.myPATeditor" >

```

```
<!-- tree builders are only used in the Administration Console editors. However,
this entry is required.-->

    <edtrs:tree-builder>
        <edtrs:mode>NOT_USED</edtrs:mode>
    </edtrs:tree-builder>
</edtrs:editor>
</edtrs:editor-group>
```

Adding Your Extension to Your Administration Console

When you have finished building your extension resources and added the necessary configuration files (a `netuix-extension.xml` file and optionally, an `editors.xml` file), you need to export your extension to your custom Administration Console application.

Note: You add your extension to the custom Administration Console project you created, see [“Creating an Administration Console Web Project” on page 3-2](#).

To add your extension to your Administration Console application:

1. In your Administration Console web project, add a folder called **bea-ext** under the WEB-INF directory. You will export your extension project to this directory.
2. In your extension web project, right-click the root directory of your extension web project and select **Export > WAR**.
3. In the Export WAR dialog, click **Browse** to navigate to the `\\WEB-INF\bea-ext` directory in your Administration Console web project. For example,
`c:\bea\user_projects\w4WP_workspaces\myWorkspace\adminConsoleWebProject\webContent\web-inf\bea-ext\`
4. Click **Finish**.
5. Using the Project Explorer, navigate to the WAR file you just exported to your custom Administration Console. You may need to press F5 to refresh the directory’s contents.
6. Right-click on the *.war file (for example, `myExtension.war`) and select **Copy**.
7. Right-click the `//WEB-INF/bea-ext` directory and select **Paste**.

8. In the Naming Conflict dialog, rename your WAR file to the same name yet with a *.jar file extension. For example, `myExtension.jar`.
9. Delete the *.war file.

Using your Extended Administration Console

You access your extended Administration Console via a URL, as with the WebLogic Portal Administration Console. For example,

`http://localhost:7001/myCustomConsoleWebProject/portal.portal`

Extending the Portal Administration Console