Oracle® WebLogic Portal

Developer's Quick Start Guide

10*g* Release 3 (10.3)

September 2008

ORACLE®

Contents

Running and Debugging an Application

Sharing Configuration Objects

Portal Web Application Organization

Developer Productivity

Server Memory and Other Settings

Application Location

Introduction

This guide provides guidance, best practices, and tips to developers for setting up, using, and increasing productivity when using WebLogic Portal. It is organized in roughly the way you would develop a portal application.

The information in this guide shows one way of accomplishing creating a portal. Because of WebLogic Portal's versatility, you can accomplish the same thing in other ways. This guide provides a minimal amount of step-by-step instructions—just enough to get you started and not waste your time with elementary use of the product. To fill in the details, each section provides references to the relevant documentation.

If you are starting from scratch, start here:

- Preparing and Creating Your Workspace – Organizing and getting your project set up.

- Creating New Portal Projects – Creating projects for your portal EAR, WAR, and other objects.

- Using Source Control – Using Eclipse Team plug-ins for interacting with source control to share projects with your development and quality assurance teams.

If someone has already created the project, start here:

- Starting from an Existing Project – Starting from already existing projects.

- Creating a Domain and Server – Creating a domain and set it up to run from Workshop for WebLogic.

- Sharing Configuration Objects – Sharing artifacts, such as content management nodes, desktops, entitlements, and roles.

- Running and Debugging an Application – Controlling your server and deploying your application.

- Portal Web Application Organization – Tips on organizing your web applications.

- Developer Productivity – Tips for being more productive.

- Server Memory and Other Settings – Tweaking your server and domain environment.

- Application Location – Notes on how Workshop for WebLogic deploys your application.

Preparing and Creating Your Workspace

Workshop for WebLogic is an Eclipse-based tool used for portal development. Workshop for WebLogic uses workspaces to organize your projects and keep track of your preferences.

This chapter includes the following topics:

- Setting Up Your Workspace and Project Directories
- Creating a Workspace
- Related Information

Setting Up Your Workspace and Project Directories

A workspace is a directory that contains all your configuration and state information, such as your Workshop for WebLogic color preferences, recently opened files, caches, and so on. By default your portal projects go in your workspace.

Tip: As a best practice, keep your projects outside of the workspace. This organization allows better sharing among team members, more flexibility for linking projects when working with multiple enterprise applications, and easier reassembly if the enterprise application is restructured.

Rather than using the default project location, which is within the workspace directories, your projects should be in their own directories—where they can be managed by source control.

You will need to create separate directories for these projects:

- EAR

- WAR

- Datasync (if used)

- Java utilities (if needed)

- EJB projects (if needed)

Listing 2-1 shows an example structure for all the modules. Although, you don't need to create every directory at this point, it may be easier to create them now rather than later (from Workshop for WebLogic).

Listing 2-1   Module Directories Example

```
HOME/workspaces/projectX/          # workspace directory for Project X

HOME/projects/projectX/            # directory for the Project X projects
             projectXapp/          # your EAR project
             projectX/             # your WAR project
             projectXdatasync/     # a Datasync project
             projectXejb/          # an EJB project
             projectXutility/      # a J2EE utility project

HOME/domains/projectX/             # domain directory for Project X
```

Creating a Workspace

After deciding on the structure or setting up your project directories, create a workspace in Workshop for WebLogic. Workshop for WebLogic is built on the Eclipse IDE framework. As a result, many of the standard features of Workshop for WebLogic are described in the Eclipse documentation, available at http://eclipse.org and Workshop for WebLogic online help.

To start Workshop for WebLogic on:

Microsoft Windows: From the Start menu, click **All Programs > Oracle Products > Workshop for WebLogic**.

Linux: Enter `<BEA_HOME>/workshop_10.3/workshop`

When Workshop for WebLogic launches, it asks you for your workspace. Select the workspace folder you created in "Setting Up Your Workspace and Project Directories" on page 2-1 or create a clean folder.

Tip: As a best practice, create your workspace outside of the installation directory. This reduces problems if you reinstall or upgrade WebLogic Portal later.

When Workshop for WebLogic opens, the initial window displays the Welcome screen. This screen displays useful links for various local and online resources. You can return to the welcome screen at any time with the **Help > Welcome** command. It may be helpful to explore the content before continuing.

To start writing code, click the button ⊚ in the upper-right corner of the welcome screen. Workshop for WebLogic is displayed.

Figure 2-1 Workshop for WebLogic IDE
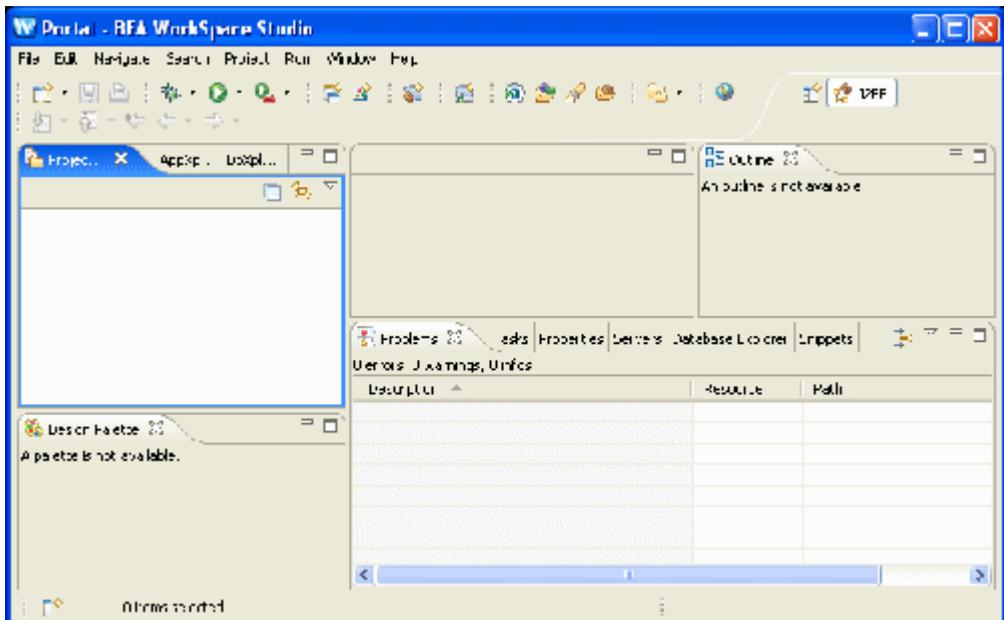
Workshop for WebLogic is the basic development environment for Eclipsed-based applications. Workshop for WebLogic contains a menu bars, tool bars, code editors, and information panes called views. Views let you browse properties, projects, and so on.

The window layout is called a perspective. A perspective is a preset arrangement of editors and views tailored for performing specific tasks with specific resources. For example, Workshop for WebLogic provides perspectives for Java EE (formerly, J2EE) development, debugging, and portal development. Each perspective can be extensively customized or you can create your own.

The perspectives that you recently used are displayed in the upper-right corner of Workshop for WebLogic, as shown in Figure 2-2. You can switch back and forth between them at any time without loosing any work; you are just changing the way you look at your work.

Figure 2-2 Recently Used Perspectives
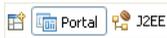
Related Information

The following topics provide more detail for preparing and creating your workspace:

- Tutorial: Getting Started with Workshop for WebLogic

- Tutorials - Getting Started with WebLogic Portal

- Workshop for WebLogic Help > Cheat Sheets > Oracle WebLogic Portal > Develop a Portal application > Open the Portal Perspective

Creating New Portal Projects

To get started developing a portal, at a minimum, you need to create a portal EAR and a portal WAR. These are just regular EAR and WAR projects that have been configured to include portal libraries and other artifacts.

This chapter includes the following topics:

- Creating Your EAR Project
- Creating Your WAR Project
- Datasync Project
- Utility Projects
- EJB Projects
- Related Information

Creating Your EAR Project

To create your EAR project:

1. Start Workshop for WebLogic.

   On the left side of the Workshop for WebLogic is the Project Explorer view. You use this view to navigate within your projects.

Figure 3-1 Project Explorer View
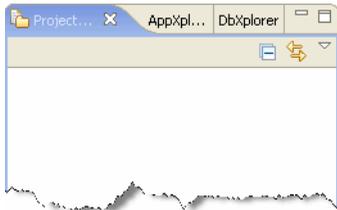
2. Right-click within the Project Explorer and select **New > Portal Ear Project**.

    Note: Although other ways exist for creating a new portal ear project, such as **File > New > Project... > Portal Ear Project**, the path from the Project Explorer is the easiest and most direct.

3. Complete the pages in the wizard as required, using the information provided in Table 3-1.

    Tip: Workshop for WebLogic includes cheat sheets for developing portal applications, how to use Workshop for WebLogic features, developing Java applications, and other topics. To access a cheat sheet from the menu bar, select **Help > Cheat Sheets**.

Table 3-1 New Portal EAR Project Wizard

In this Wizard ...	Select or Enter...
Project name	This name will become the default base name for several things, such as the portal administration tools URL. For example, if the name is projectXapp, the administration tools URL will be /projectXappAdmin.
Use default	Deselect this check box. The default location for projects is within your workspace. However, as recommended in Chapter 2, "Preparing and Creating Your Workspace", your project should not go within your workspace directory.

Table 3-1 New Portal EAR Project Wizard (Continued)

In this Wizard ...	Select or Enter...
Browse	Browse to the location where you want to put your EAR project. This should be an empty directory with the same name as the project. Based on the example in Listing 2-1, it would be something like `HOME/projects/projectX/projectXapp`. **Note:** Naming your directories with the same root name is not required, but does eliminate ambiguity. You can create any new directories you need by using the **Make New Folder** button in the Browse For Folder dialog.
Project Facets page	By default you get the facets needed for a portal. You can also add other facets such as Collaboration (Groupspace), Commerce, or Analytics. If you do not know if you if need these facets, you can add them later.

4. After clicking **Finish**, click **Yes** when the dialog asks if you want to open the associated perspective.

   Your EAR project is created, but shows an error, as indicated by the red X 🐞 on the project in the Project Explorer. The Problems view (Figure 3-2) shows EAR projects are required to contain one or more modules.

Figure 3-2 Problems View
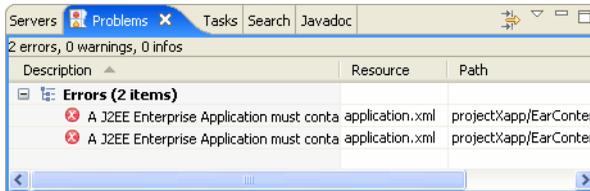
The error occurs because you have not yet created a WAR project (or other kind of project) in the EAR.

Creating Your WAR Project

To create your WAR Project:

1. Within the Project Explorer, right-click your project and select **New > Portal Web Project**.

2. Complete the pages in the wizard as required.

Table 3-2 New Portal Web Project Wizard

In this Wizard ...	Select or Enter...
Project name	By default this name is used as the context root of your web application. For example, if you choose a project name of `projectX`, your URLs for the web application will start with `/projectX`.
Use default	Deselect this check box.
	The default location for projects is within your workspace. However, as recommended in Chapter 2, "Preparing and Creating Your Workspace", your project should not go within your workspace directory.
Browse	Browse to the location where you want to put your WAR project. This should be an empty directory with the same name as the project. Based on the example in Listing 2-1, it would be something like `HOME/projects/projectX/projectX`.
Add project to an EAR	Make sure that EAR you created is selected.
	This adds the WAR as a module within the EAR and clears the EAR's build error.
Project Facets Page	Carefully look at the facets list and see what you might need. The default set covers most situations. You can always go back and add and remove facets as needed.

Datasync Project

A datasync project is an optional project that stores general purpose portal services data used in the development of personalized applications and portals. These portal services include User profiles, session properties, and campaigns.

To create the Datasync project:

1. From the menu bar, click **File > New > Datasync Project**.

2. Complete the pages in the wizard as required.

Table 3-3 Datasync Project Wizard

| In this Wizard ...	Select or Enter...
Project name	Name of the Datasync project.
Use Default	Deselect this check box.
	The default location for projects is within your workspace. However, as recommended in Chapter 2, "Preparing and Creating Your Workspace", your project should not go within your workspace directory.
Browse	Browse to the location where you want to put your Datasync project. This should be an empty directory with the same name as the project. Based on the example in Listing 2-1, it would be something like `HOME/projects/projectX/projectXdatasync`.
EAR Membership	Make sure that the EAR you created is selected.

Utility Projects

Use a Utility project for application-scoped source code. In particular, for plain Java classes that are shared by other projects that are included in your EAR. The Utility project code is in the EAR's `APP-INF/classes` directory. This means it is not a deployed submodule and does not get a `<java>` entry in `META-INF/application.xml`.

Note: A submodule can be deployed separately from an EAR, which can significantly improve redeployment time. For more information, see "Prefer Quicker Redeploys" on page 10-2.

To create the Utility project:

1. Within the Project Explorer, right-click and select **New > Other**.

2. In the Select a wizard dialog box, select **J2EE > Utility Project**, and then click **Next**.

3. Complete the pages in the wizard as required.

Table 3-4 Utility Project Wizard

In this Wizard ...	Select or Enter...
Project name	The name of the utility.
Use Default	Deselect this check box.
	The default location for projects is within your workspace. However, as recommended in Chapter 2, "Preparing and Creating Your Workspace", your project should not go within your workspace directory.
Browse	Browse to the location where you want to put your Datasync project. This should be an empty directory with the same name as the project. Based on the example in Listing 2-1, it would be something like `HOME/projects/projectX/projectXutility`.
Add project to an EAR	Select the appropriate EAR project from the drop-down list.
Project Facets page	If you will be writing EJB or Java code that uses the WebLogic Portal APIs, be sure to select the facets that give access to those APIs.

EJB Projects

If you are planning to write any EJBs, you need to create an EJB project. An EJB project requires an `ejb-jar.xml` with at least one EJB. If you do not want to create an EJB project now, you can always do it later.

Tip: EJB projects are also useful for plain Java code because an EJB is deployed as a submodule. Submodules are advantageous because they can significantly improve the time it takes to redepoy. So you might want to create an EJB project even if you are not using EJBs. For more information, see "EJB Projects in a Sub-classloader Deploy Faster" on page 10-3.

To create an EJB project:

1. From the menu bar, click **File > New > Other**.

2. In the Select a wizard dialog box, select **EJB > WebLogic EJB Project**.

3. Complete the pages in the wizard as required.

Table 3-5 EJB Project Wizard

In this Wizard ...	Select or Enter...
Project name	The name of your EJB Project.
Use Default	Deselect this check box.
	The default location for projects is within your workspace. However, as recommended in Chapter 2, "Preparing and Creating Your Workspace", your project should not go within your workspace directory.
Browse	Browse to the location where you want to put your EJB project. This should be an empty directory with the same name as the project. Based on the example in Listing 2-1, it would be something like `HOME/projects/projectX/projectXejb`.
Add project to an EAR	Select the appropriate EAR project from the drop-down list.
Project Facets page	If you will be writing EJB or Java code in this project that uses the WebLogic Portal APIs, make sure you select the correct facets to give access to those APIs.
	Note: The WebLogic EJB Project uses WebLogic EJBGen to generate the Home and Remote interfaces and the `ejb-jar.xml` file. An EJBGen project does not build or deploy without EJBGen-style EJBs.

The new EJB project shows an error until you create at least one EJB. You can create a placeholder EJB for now. For information on how to do this, see Tutorial: Building Enterprise JavaBeans.

If your EJB project depends on code in your Utility project:

1. In Project Explorer, right-click your EJB project and select **Properties**.

2. On the left side of the Properties dialog box, select **J2EE Module Dependencies**.

3. Under Available dependent JARS, select your utility JAR.

4. Click **OK**.

Related Information

The following topics provide more detail for creating new portal projects:

- Starting Workshop for WebLogic in *WebLogic Portal Getting Started*

- Setting up Your Portal Development Environment in the *Portal Development Guide*

- Tutorial: Building Enterprise JavaBeans

- "Prefer Quicker Redeploys" on page 10-2

- "EJB Projects in a Sub-classloader Deploy Faster" on page 10-3

- "Creating a Workspace" on page 2-2

- Portal EAR Project Wizard in the *Portal Development Guide*

- Portal Web Project Wizard in the *Portal Development Guide*

- Portal Datasync Project Wizard in the *Portal Development Guide*

- Workshop for WebLogic Help > Cheat Sheets > Oracle WebLogic Portal > Develop a Portal application >

    – Create a portal EAR project

    – Create a portal Web project

    – Create a datasync project

    – Associate the portal Web and portal EAR projects

    – Associate the datasync and portal EAR projects

    – Open the Portal Perspective

Using Source Control

After you have created a basic set of projects, you can add them to your source control system.

This chapter includes the following topics:

- Team Plug-ins
- Manual Source Control
- Related Information

Team Plug-ins

Workshop for WebLogic integrates with source control systems using Eclipse Team plug-ins. Team plug-ins are useful because they handle source control so you can concentrate on developing your projects. Team plug-ins are also smart. For example, they exclude generated files from being checked in because they know the difference between your source code and generated files and classes. Additionally, most Team plug-ins respond properly when you create new files by adding them to source control, and to refactoring by checking out the appropriate files.

Eclipse has integrated features for working with CVS (Concurrent Versions System) repositories. You can also install a plug-in that integrates your source control system with Eclipse.

The following list provides useful information when setting up and working with source control:

- Working in Source Control – In Eclipse, select **Help > Help Contents > Oracle Workshop for WebLogic User's Guide > Common IDE Tasks> Working with Source Control**.

- Team CVS tutorial – In Eclipse, select **Help > Help Contents > Workbench User Guide > Getting started > Team CVS tutorial**.

- Eclipse Plugin Centra**l** – http://www.eclipseplugincentral.com.

Manual Source Control

Note: This method is not recommended. The following information is provided for reference only.

If you use source control manually, you will need to exclude generated files and classes. One way to do this is to clean the projects.

1. From the Project menu in the menu bar, turn off **Build Automatically**.

2. Again from the Project menu, select **Clean > Clean all projects**.

3. Deselect the **Start a build immediately** check box and then click **OK**.

4. Once the projects are clean, check in the remaining project files.

   Note: Make sure you don't miss the dot files, such as `.project`, `.classpath`, and `.settings`.

5. Turn **Build Automatically** back on after you have checked the files in.

Note: You will have to keep track of any files you add or change.

Related Information

The following topics provide more detail for using source control:

- Managing a Team Development Environment in the *Production Operations Guide*

- Understanding the Build Process in Building Applications

- Eclipse Plugin Central – http://www.eclipseplugincentral.com.

- Workshop for WebLogic Help > Cheat Sheets > Team/CVS

- Working with Source Control (Workshop for WebLogic document)

- Workshop for WebLogic Help > Help Contents > Workbench User Guide > Getting started > Team CVS tutorial

Starting from an Existing Project

If your project has already been created, start here.

This chapter includes the following topics:

- Check Out Projects from Source Control
- Importing Existing Projects
- Related Information

Check Out Projects from Source Control

Check out your projects from source control.

Importing Existing Projects

To import an existing project:

1. Start Workshop for WebLogic.

2. If needed, create a workspace in Workshop for WebLogic.

Tip: Create your workspace in the same place as your other working files, but not within the project directories. See "Creating a Workspace" on page 2-2.

3. From the menu bar, select **File > Import**.

4. In the Select dialog box, select **General > Existing Projects into Workspace**, and then click **Next**.

5. Select the top-level project directory (the one that contains all the EJB, WAR, and other project folders). If no common root directory exists, select each project individually.

6. In the Projects list, make sure you select all the projects and import them.

    Note: Do not select the **Copy projects into workspace** check box; leave them where they were checked out.

7. After importing, right-click each project and select **Team > Share Project** for each project.

    This associates your projects with your source control. See Chapter 4, "Using Source Control."

Related Information

The following topics provide more detail for starting from an existing project:

- "Creating a Workspace" on page 2-2

- Managing a Team Development Environment in the *Production Operations Guide*

- Tutorial: Getting Started in Workshop for WebLogic

- Importing Existing Web Applications (Oracle Workshop for WebLogic document)

Creating a Domain and Server

At this point you may not have not written any code, except perhaps a placeholder EJB. Even so the generated projects are valid applications, and the web application contains a default Page Flow Controller, which provides access to a web page. For more information about page flows, see Page Flows in Portals in the *Portal Development Guide*.

The chapter contains:

- Creating a Server

- Domain Configuration Wizard

- Related Information

Creating a Server

To create a server in Workshop for WebLogic:

1. From either the Portal or J2EE perspective, click the **Servers view**.

2. Right-click inside the Servers view and select **New > Server**.

3. Complete the pages in the wizard as required.

Table 6-1 New Server Wizard

In this Wizard ...	Select or Enter...
Server's host name	localhost
Server type	Oracle WebLogic Server v10.3
Server runtime	Oracle WebLogic Server 10.3
Domain home	Your domain home directory. For example, HOME/domains/projectX. **Tip:** As a best practice, create your domain outside of the installation directory. If you do not already have a domain, create one by selecting **Click here to launch the Configuration Wizard to create a new domain**. For information about what to enter in this wizard, see "Domain Configuration Wizard" on page 6-3.
Add and Remove Projects	Your EAR project should be visible. As a best practice, do not move the EAR to the server yet. This way you can make sure the server starts cleanly by itself. You can add the project later.
Selected Tasks	This page should be empty unless you have added your EAR Project to the server.

After completing the New Server Wizard, the Servers view displays the server you just created along with the server's status and state.

- Status – Refers to the server itself: Stopped (not running), Starting, Started, and so on.

- State – Refers to any projects deployed on the server: Republish (require redeployment) or Synchronized (all is well).

Note: Eclipse uses the term *Publish* where Workshop for WebLogic uses *Deploy*; these terms are synonyms.

Domain Configuration Wizard

If you are creating a new domain, complete the pages in the domain wizard as required.

Table 6-2 Domain Wizard

In this Wizard ...	Select or Enter...
Welcome page	Create a new WebLogic domain.
Select Domain Source	In the **Generate a domain configured automatically to support the following Oracle Products**, select either **WebLogic Portal** or **WebLogic Portal Collaboration Repository**. Be sure you select the template with the features you need.
Configure Administrator Username and Password	The user name and password of your choice.
Configure Server Start Mode and JDK	Development Mode. Oracle recommends JRockit SDK 1.5.0_11, as it is faster than the Hotspot JVM for iterative development.
Customize Environment and Services Settings	No.

Table 6-2 Domain Wizard (Continued)

In this Wizard ...	Select or Enter...
Create WebLogic Domain	The name entered in the Domain name field becomes a subdirectory under the Domain location directory. For example, if you want your domain to be `HOME/domains/projectX`, enter `projectX` as the domain name, and `HOME/domains` for the location.
Creating Domain	If the path to your domain is long, you can copy the path information from the final and then paste it into the Domain home field in the New Server wizard. Preparing... Extracting Domain Contents... Creating Domain Security Information... Saving the Domain Information... Storing Domain Information... String Substituting Domain Files... Performing OS Specific Tasks... Performing Post Domain Creation Tasks... Domain Created Successfully! **Copy** Domain Location: C:\HOME\domains\projectX
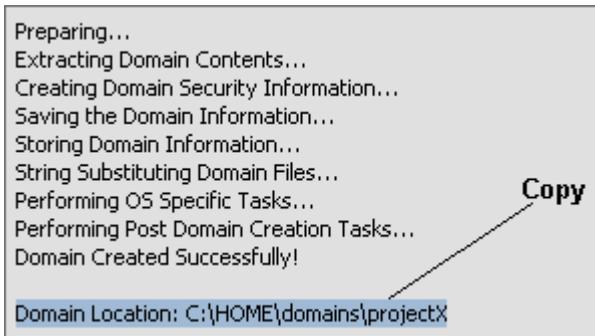
Related Information

The following topics provide more detail for creating a domain for your server:

- Tutorials - Getting Started with WebLogic Portal – These tutorials provide instructions on using the various configuration wizards.

- Tutorial: Getting Started with Workshop for WebLogic (Workshop for WebLogic document)

- Understanding Oracle JRockit in the *JRockit Diagnostic Guide*.

- *Creating Templates Using the Domain Template Builder*.

Running and Debugging an Application

This chapter includes the following topics:

- Starting the Server

- Deploying or Debugging the Application

- Related Information

Starting the Server

The Server view provides two modes for starting the server:

- ⬤ Run mode – For faster performance, use this mode when you aren't doing any debugging with Workshop for WebLogic.

- ✿ Debug mode – Use this mode when debugging, If you aren't debugging, it will slow the server slightly.

Note: If you start the server in run mode and later decide to debug, you will have to stop the server, and then restart in Debug mode.

After starting the server in either mode, the Console view is displayed. The Console view shows the running log output. The Server view shows Starting while the server is launching; the status changes to Started after start up is complete.

> **Tip:** You can change the way information is displayed in the Console view by right-clicking within the view and selecting **Preferences**.

As the server is starting, the Server and Console views may switch focus back and forth. To change this behavior, do one of the following:

- Right-click in the Console view and select **Preferences**. Then deselect **Show when program writes to standard out** and/or **Show when program writes to standard error**.

- Drag the Console view to some other place on Workshop for WebLogic so it can produce its output without stealing focus from other views.

Deploying or Debugging the Application

To deploy or debug the application:

1. Do one of the following:

   – To deploy, right-click your web application project and select **Run As > Run on Server**.

   – To debug, right-click your web application project and select **Debug As > Debug on Server**.

2. In the Define a New Server dialog box, select your server. (There should be only one.)

3. Select **Set server as project default**. (Your project will always use this server.)

4. Click **Finish**.

The web application will now run on the server. If the project is not built, Workshop for WebLogic builds it and then publishes (deploys) it. In the Servers view, the State column displays Publishing. When the application is deployed, a browser opens in Workshop for WebLogic and loads the default page for the web application. The default page is index.jsp, which comes from the default package (Controller JPF).

> **Tip:** Workshop for WebLogic uses its internal browser to display web pages. You can switch Workshop for WebLogic to use your normal (external) browser by going to the menu bar, and selecting **Window > Web Browser > <Your Browser>**.
>
> You can also choose to always use an external web browser to view your portal. To do so, select **Window > Preferences** and select **General > Web Browser** in the property

tree. Next, select the **Use external Web browser** and pick a browser type from the list. If no browsers appear in the list, you can search for available browsers and add them to the list.

Anytime you select **Run on Server** or **Debug on Server** for a resource, such as a `.portal` file, the project is built and redeployed as necessary. Additionally the selected resource opens in the browser. However, the one exception is if you have a page flow controller (JPF file) associated with that folder (package). In this case, the Page Flow runs rather than the individual file you have created. Subsequently, once you have actual content in your web application, you may want to delete the default `controller.java` and `index.jsp` files that were generated when you created the project.

Related Information

The following topics provide more detail for running or debugging your application on the server.

- Oracle WebLogic Server (Oracle Workshop for WebLogic document)

- Defining the Server Inside the IDE in Oracle WebLogic Server

Sharing Configuration Objects

At this point you might need to choose a strategy for sharing the configuration of your domain, database, and other objects with the rest of your development team.

This chapter includes the following topics:

- Strategies for Sharing

- Related Information

Strategies for Sharing

You may need to account for these two types of configuration, and how you will identify them or keep them separate in your development environment. The two configuration types are:

- Configuration of the staging or production servers—including desktops, content management data and structures, entitlements, and other configuration objects. It can also include any database schema definitions that are unique to your project.

- Configuration of the development and test servers—including test data and test users, as well as the configurations listed in the previous bullet item.

The main ways to share configuration objects are:

- Manually using a set of instructions.

- Using scripts such as WLST, DDL, Ant, and so on.

- Using JSPs, servlets, and so on

- Propagation—moving the database contents of a portal application from one server environment to another.

Each way has its advantages and drawbacks. Most likely you will use a mixture of these techniques.

Related Information

The following topics provide more detail for sharing configuration objects:

- Managing a Team Development Environment in the *Production Operations Guide*

- Developing a Propagation Strategy in the *Production Operations Guide*

Portal Web Application Organization

The following topics will help you organize your portal web application:

- Directory Structure

- Portal Organization

- Support Proxy Caching

- Plan for Upgrading WebLogic Portal

- Related Information

Directory Structure

The structure described here is suggested to help keep your portal web application stay organized and easy to navigate as it grows larger. The following list demonstrates a structure for portlets. Use a similar organizational technique for portals, skins, look-and-feel files, and so on.

- Create a top-level folder, under the WebContent folder named portlets.

- Create a folder (or folder structure) for each portlet.

- Put the `.portlet` file, JSPs, and other files associated with the portlet in the portlet folder.

  WebContent
    /portlets
      /utility
        /weather
         weather.portlet
         weather_summary.jsp

- Match the portlet folder structure for any associated utility classes, backing files, and other files in a package. For example, the backing and utility classes might go in something like `org.foo.projectx.portlets.utility.weather` package.

Portal Organization

To organize your portals, create `.book` and `.page` files separately from the `.portal` file rather than inline. Although a simple portal can use an inline structure, keeping these files separate is useful for managing and organizing your portal as it grows.

For more information, see "Creating a Standalone Book or Page" and "Extracting an Existing Page or Book to Re-Use" in Developing Portals Using Workshop for WebLogic in the *Portal Development Guide*.

Support Proxy Caching

Create static content under separate top-level folders. This makes them easier to deploy from a front-end like Apache or a caching/edge proxy. For example, using the sample in "Directory Structure" on page 9-1:

```
WebContent
   /images
       /weather

   /icons
       /weather

   /content (static content)
     /weather
```

Plan for Upgrading WebLogic Portal

In WebLogic Portal, certain files work together as a logical unit. This means that a change in one file may affect another. During an upgrade or patch, it's possible that a file you changed will not be upgraded, while a file that it depends on may be upgraded. As a best practice, especially when using Workshop for WebLogic's **Copy to Project** feature, do one of the following to avoid problems with the way a logical group of files interact with each other:

● Create your own versions of files, with their own names and paths, from the J2EE libraries. Do not override the library modules at all.

● If you do override a file, make sure to override a logical unit of files, rather than overriding individual files within that group.

Tip: This suggestion is valid for any set of files you might override, but it is especially important when tweaking skins, skeletons, and layouts. It is best to create your own look-and-feel files, using the original library modules as a starting point, rather than overriding the default look-and-feel files.

For additional information, see Installing Maintenance Updates, Service Packs, and Maintenance Packs.

Related Information

The following topics provide more detail for organizing your portal application:

● User Interface Development with Look And Feel Features in the *Portal Development Guide*

● Using the Merged Projects View in the *Portal Development Guide*

Portal Web Application Organization

Developer Productivity

This chapter includes the following tips to increase developer productivity:

- Use the Team Plug-in

- Use JRockit

- Run Without Debug when Possible

- Make Sure You Have Enough Heap

- Prefer Quicker Redeploys

Use the Team Plug-in

See "Team Plug-ins" on page 4-1.

Use JRockit

Based on the test described in the next paragraph with a basic portal application, the JRockit JVM is faster than the Hotspot JVM for iterative development.

Using a scenario that is based on a typical developer's tasks—starting a server, deploying an application, then redeploying the application ten times—JRockit saves several minutes of wait time on a reasonably fast Linux machine. You'll save even more time on a slower computer.

For more information using JRockit, see:

- Oracle JRockit Online Documentation

- "Creating a Server" on page 6-1

- "JVM" on page 11-3

Run Without Debug when Possible

Running the Debug Server slows down both JRockit and the Sun JVM. In the developer's scenario described in the previous section, running debug add several minutes of wait time.

Obviously, using debugging will save you much more time in productivity when you need it compared to sprinkling your code with `printlns`. However, if you are not going to use debug, start the server without it.

For additional information, see:

- Chapter 7, "Running and Debugging an Application"

Make Sure You Have Enough Heap

As your portal gets larger, you may need to increase the heap (-Xmx) and initial heap sizes (-Xms). It is generally best to make the initial and maximum heap sizes equal.

For more information, see:

- Tuning Oracle JRockit JVM in *Tuning the JRockit JVM*

- Tuning the Memory Management System in the *JRockit Diagnostics Guide*

Prefer Quicker Redeploys

Because of the size of a portal EAR, redeployment of a portal application can be slow—several minutes, depending on the machine. However, you can organize your work to spend less time waiting and avoid slow redeployment in the following ways:

- Some Resources Do Not Need to Redeploy

- Web Applications Redeploy Faster than their Enclosing EARs

- EJB Projects in a Sub-classloader Deploy Faster

Some Resources Do Not Need to Redeploy

JSPs, `.portal` files, and similar objects do not need to redeploy. For example, if you change a JSP and then republish in Workshop for WebLogic, basically nothing happens. Use coding in a JSP for light investigation or testing.

For information about using Workshop for WebLogic for JSP development, see Oracle Workshop for WebLogic for JSP (Oracle Workshop for WebLogic document).

Web Applications Redeploy Faster than their Enclosing EARs

The redeployment of a web application, including a portal web application, is significantly faster than deploying the entire EAR.

If your only change a web application, Workshop for WebLogic republishes only that web application. Therefore, it is faster to code in a web application as classes in the web project's `Java Resources/src` folder. They deploy in the web application's `WEB-INF/classes`, and Workshop for WebLogic directs the server redeploy only the WAR (unless you change something that requires the application to redeploy).

For additional information, see Streamlining Deployment/Testing (Oracle Workshop for WebLogic document).

EJB Projects in a Sub-classloader Deploy Faster

Because the majority of application redeployment time is spent processing resources like portal EJBs, you can save time by avoiding redeployment of these resources. For instance, if you have code that is application-scoped, such as EJBs, you may be able to deploy the code in a submodule. However, use of this method is limited, due to the ClassLoader arrangement that it requires. This technique will not work if the application calls code in the EJB project, such as implementing a content management SPI or registering P13n EventListeners and UUP.

To make an EJB project a submodule of the EAR, add a classloader-structure stanza to your EAR's `weblogic-application.xml` file. This allows the EJB and WAR projects to redeploy without redeploying the EAR.

To add a classloader-structure stanza to your EAR's `weblogic-application.xml` file:

1. Create an EJB project, as discussed in "EJB Projects" on page 3-6, and associate that EJB project with your application project.

2. In the Projects view, open the Weblogic Deployment Descriptor for the EAR, and select the Source view.

3. Add a stanza like this, changing the `module-uri` names to those of your project:

```
<wls:classloader-structure>
   <wls:classloader-structure>
      <wls:module-ref>
        <wls:module-uri>projectXejb.jar</wls:module-uri>
      </wls:module-ref>
      <wls:classloader-structure>
        <wls:module-ref>
           <wls:module-uri>projectX.war</wls:module-uri>
        </wls:module-ref>
      </wls:classloader-structure>
   </wls:classloader-structure>
</wls:classloader-structure>
```

Because this stanza makes your EJB project (`projectXejb`) a submodule of the EAR, it allows WebLogic Server to redeploy the EJB JAR without requiring the entire EAR to redeploy. Notice that the WAR project (`projectX`) is made a child of the EJB, which allows the code in the WAR to access the code in the EJB project.

For more information, see:

● Tutorial: Building Enterprise JavaBeans (Oracle Workshop for WebLogic document).

● Understanding WebLogic Server Application Classloading in *Developing Applications with WebLogic Server*

Server Memory and Other Settings

This chapter includes the following topics:

- Start-up Scripts

- JVM

- Memory Settings

- Classpath

- Other Java Options

- PointBase

- Autonomy

- Using the nodebug Argument

- More Complex Settings

Start-up Scripts

Settings for your server should go in the command line or start-up scripts. To localize the default overrides and make it easier to share the changes with your team, add them to the startWebLogic scripts:

- `DOMAIN_HOME/startWebLogic.cmd` (Windows)

- `DOMAIN_HOME/startWebLogic.sh` (UNIX)

To edit the startWebLogic script in Workshop for WebLogic:

1. In the Server view, double-click the server.

2. In the Server Overview view, click **Edit** to the right of the **Startup script** entry.

3. Edit the script as necessary. Figure 11-1 shows an example for both Windows and Linux.

   Common environment variables you might want to set are:

   – JVM
   – Memory Settings
   – Classpath
   – Other Java Options
   – PointBase
   – Autonomy
   – Using the nodebug Argument
   – More Complex Settings

Figure 11-1 Example Start WebLogic Scripts

```
startWebLogic.sh
#!/bin/sh
DOMAIN_HOME="/HOME/domains/projectX"

# Use JRockit
export JAVA_VENDOR="BEA"

# Set initial and max heap to 1G
export USER_MEM_ARGS="Xms1g -Xmx1g"

# Turn off autonomy
export CONTENT_SEARCH_OPTION="none"

# Add my system property
export JAVA_OPTIONS="-DmySystemProperty=foo"

${DOMAIN_HOME}/bin/startWebLogic.sh $*
```

```
*workshop.startWebLogic.cmd  ✕
@ECHO OFF
SETLOCAL
set DOMAIN_HOME="C:\Domains\projectX"

REM Use JRockit
set JAVA_VENDOR="BEA"

REM Set initial and max heap to 1G
set USER_MEM_ARGS="Xms1g -Xmx1g"

REM Turn off autonomy
set CONTENT_SEARCH_OPTION="none"

REM Add my system property
set JAVA_OPTIONS="-DmySystemProperty=foo"

call "%DOMAIN_HOME%\bin\startWebLogic.cmd" %*
ENDLOCAL
```

JVM

When creating a domain in the Domain Configuration Wizard, you set the default JVM, as described in Chapter 6, "Creating a Domain and Server." If you need to change or override the default JVM, in the startWebLogic script, set JAVA_VENDOR to either BEA for JRockit, or Sun for Hotspot. This triggers the domain to use the specified JVM as supplied by the WebLogic Portal installer and any JVM-specific settings for memory and other options.

Memory Settings

The default memory settings for WebLogic Portal are:

JRockit	`-Xms256m -Xmx768m`
Sun Hotspot	`-Xms256m -Xmx768m -XX:CompileThreshold=8000 -XX:PermSize=48m -XX:MaxPermSize=128m`

To override these, set `USER_MEM_ARGS` in the startWebLogic script. This is useful for gaining more heap or PermSize.

In general, JRockit requires more heap than the same server run on Hotspot. This is probably because Hotspot separates some objects out into Permanent Generation, while JRockit uses heap for these objects.

Servers running either JVM startup and perform initial application deploys faster if you provide enough initial memory (`Xms` and `PermSize`) so that server startup does not require extra allocations and the associated garbage collection passes. Generally, set the initial and maximum sizes the same.

For more information, see Tuning the Memory Management System in the *JRockit Diagnostics Guide*

Classpath

The two ways to deploy into the Java classpath are:

- Copy JARs into your domain's `DOMAIN_HOME/lib` directory. They are automatically picked up by the server; no other configuration is needed.

- Set the Java classpath in the startWebLogic script, use these environment variables:

  – `EXT_PRE_CLASSPATH` – adds to the beginning the server's classpath.

  – `EXT_POST_CLASSPATH` – adds to the end of the server's classpath.

Other Java Options

You can set other Java options in the startWebLogic script using the `JAVA_OPTIONS` setting. This setting is additive; it does not override anything else. This is a good place to put things like system properties.

PointBase

The startup of the PointBase database server is controlled in the startWebLogic script by the command line argument `nopointbase`. There is not an environment variable to control this setting.

To start the server without PointBase, run the command:

```
startWebLogic nopointbase
```

To permanently set `nopointbase` for a domain, add this argument to where `DOMAIN_HOME/startWebLogic.sh` or `DOMAIN_HOME/startWebLogic.cmd` calls the `DOMAIN_HOME/bin/startWebLogic` script.

This is useful when you are using an external database, such as Oracle or SQL Server.

For information about using databases, see the *Database Administration Guide*.

Autonomy

The startup of Autonomy is controlled by the `CONTENT_SEARCH_OPTION` in the startWebLogic script. To turn off Autonomy startup, set this option to `none`. Use this option when you do not need to run Autonomy or when it is already running as an external daemon.

For information about using Autonomy, see *Integrating Search*.

Using the nodebug Argument

For production mode servers, the default is `nodebug`. However, for development environments, although the startWebLogic script accepts a `nodebug` command line argument, it is not advisable to try to override this setting with your scripts because it confuses Workshop for WebLogic. Instead use the **Server** or **Debug** buttons in the Server view.

If you are not using Workshop for WebLogic, you can use the `nodebug` argument. The debug mode sets the following arguments in the JVM:

```
-Xdebug -Xnoagent
-Xrunjdwp:transport=dt_socket,address=8453,server=y,suspend=n
-Djava.compiler=NONE
```

More Complex Settings

If you have complex requirements, such as starting external processes or significant changes to server settings, you should create your own domain template to capture and replay these changes. A domain template facilitates sharing complex domain configurations with the rest of your team. For information about creating your own domain templates, see:

- *Creating Templates Using the Domain Template Builder*

- *Creating WebLogic Domains Using the Configuration Wizard*

Application Location

Workshop for WebLogic deploys your application to the server as a split-source application.

This chapter includes the following topics:

- Split-source Directories

- Related Information

Split-source Directories

The deployed application is an exploded directory inside your Workspace. This directory only contains copies of the files from META-INF directory in your application and a .beabuild.txt file, which lays out mappings for the actual application content.

The deployed split-source application is located in:

```
WORKSPACE/.metadata/.plugins/org.eclipse.core.resources/.projects/
APP_NAME/beadep/DOMAIN_NAME/APP_NAME
```

where WORKSPACE is your Eclipse workspace directory, APP_NAME is the name of your EAR project, and DOMAIN_NAME is the server domain name.

Figure 12-1 is an example of a .beabuild.txt file.

Listing 12-1   .beabuild.txt file

```
/home/projectX/projectX/build/weboutput = projectX.war
/home/projectX/projectXapp/EarContent/APP-INF/classes = APP-INF/classes
/home/projectX/projectX/WebContent = projectX.war
/home/projectX/projectX/build/classes = projectX.war/WEB-INF/classes
/home/projectX/projectXejb/build/classes = projectXejb.jar
/home/projectXutility/build/classes = APP-INF/lib/projectXutility.jar
/home/projectX/projectXdatasync/src = META-INF/data
```

The `.beabuild.txt` file pulls in the actual directory locations (on the left of the equals sign) to their virtual locations within the application (on the right of the equals sign). The deployed application picks up the actual files from your project's Workshop for WebLogic build output, WebContent directories, and so forth. This means that if you change JSPs, HTML files, or other files in your application, the server sees it immediately.

Related Information

The following topics provide more detail about split-source applications:

- Managing Folder Mappings (Oracle Workshop for WebLogic document).
- Creating a Split Development Directory Environment in *Developing Applications with WebLogic Server*