



BEA WebLogic Portal™

Team Development Guide

Copyright

Copyright © 2004-2005 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE 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 SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Trademarks or Service Marks

BEA, BEA WebLogic Server, Jolt, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Liquid Data for WebLogic, BEA Manager, BEA WebLogic Commerce Server, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Enterprise Security, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic JRockit, BEA WebLogic Personalization Server, BEA WebLogic Platform, BEA WebLogic Portal, BEA WebLogic Server Process Edition, BEA WebLogic Workshop and How Business Becomes E-Business are trademarks of BEA Systems, Inc.

All other trademarks are the property of their respective companies.

About This Document

This guide shows you how to configure, store, and manage a common development domain, database data, and portal applications in source control, letting you quickly and consistently develop, build, and update your portal applications.

Product Documentation on the dev2dev Web Site

BEA product documentation, along with other information about BEA software, is available from the BEA dev2dev Web site:

<http://dev2dev.bea.com>

To view the documentation for a particular product, select that product from the list on the dev2dev page; the home page for the specified product is displayed. From the menu on the left side of the screen, select Documentation for the appropriate release. The home page for the complete documentation set for the product and release you have selected is displayed.

Contact Us!

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

In your e-mail message, please indicate that you are using the documentation for BEA WebLogic Portal ProductVersion.

If you have any questions about this version of BEA WebLogic Portal, or if you have problems installing and running BEA WebLogic Portal, contact BEA Customer Support at

<http://support.bea.com>. You can also contact Customer Support by using the contact information provided on the quick reference sheet titled “BEA Customer Support,” which is included in the product package.

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

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

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
<i>italics</i>	Indicates emphasis or book titles.
monospace text	<div>Indicates <i>user input</i>, as shown in the following examples:</div> <ul style="list-style-type: none">• Filenames: <code>config.xml</code>• Pathnames: <code>BEAHOME/config/examples</code>• Commands: <code>java -Dbea.home=BEA_HOME</code>• Code: <code>public TextMsg createTextMsg(</code> <div>Indicates <i>computer output</i>, such as error messages, as shown in the following example:</div> <pre>Exception occurred during event dispatching:java.lang.ArrayIndexOutOfBoundsException: No such child: 0</pre>
monospace boldface text	<div>Identifies significant words in code.</div> <div><i>Example:</i></div> <pre>void commit ()</pre>

Convention	Item
<i>monospace</i> <i>italic</i> <i>text</i>	Identifies variables in code. <i>Example:</i> String <i>expr</i>
{ }	Indicates a set of choices in a syntax line. The braces themselves should never be typed.
[]	Indicates optional items in a syntax line. The brackets themselves should never be typed. <i>Example:</i> java utils.MulticastTest -n <i>name</i> [-p <i>portnumber</i>]
	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed. <i>Example:</i> java weblogic.deploy [<i>list</i> <i>deploy</i> <i>update</i>]
...	Indicates one of the following in a command line: <ul style="list-style-type: none"> • That an argument can be repeated several times in a command line • That the statement omits additional optional arguments • That you can enter additional parameters, values, or other information The ellipsis itself should never be typed. <i>Example:</i> buildobjclient [-v] [-o <i>name</i>] [-f "file1.cpp file2.cpp file3.cpp . . ."]
.	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

About This Document

Contents

Team Development Guide

Choosing a Source Control Vendor	1
Creating a Shared Portal Domain.	2
The BEA Home Directory	2
Why a Common BEA Home Matters	2
Managing Multiple BEA Home Locations for Your Team	5
Creating and Sharing the Portal Domain	7
Best Practices for Creating a Portal Domain to Share with a Team	7
Excluding Domain Files From Source Control Management	9
Binary Files in Source Control Management	9
Developing Against an Enterprise-Quality Database	12
Creating and Sharing the Portal Application	13
Where to Create the Workshop Application	14
Checking in the Workshop Application	14
Excluding Portal Application Files From Source Control Management.	14
Managing Checkouts of the Workshop Application.	15
Portal Coding Practices.	16
Java Projects	16
Cross-Platform Support	17
Definition Labels for Portal Components	17
Cluster Configuration	17
Setting up Config Templating (config-template.xml)	17

Using Multiple Enterprise Applications in a Single Domain 19

Team Development Guide

Team development of a portal Web site revolves around good source control. Proper use of a source control management system has many benefits, such as close integration between team members, the ability to quickly scale the size of a development team, and protection against data loss.

This guide shows you how to configure, store, and manage a common development domain, database data, and portal applications in source control, letting you quickly and consistently develop, build, and update your portal applications.

Use this guide in conjunction with the guide to “Deploying Portal Applications” at <http://e-docs.bea.com/wlp/docs81/deploy/index.html> for full coverage of deployment issues.

This document contains the following sections:

- [Choosing a Source Control Vendor](#)
- [Creating a Shared Portal Domain](#)
- [Creating and Sharing the Portal Application](#)

Choosing a Source Control Vendor

There are a number of source control providers, such as CVS, Perforce, StarTeam, Visual Source Safe (VSS), and PVCS. This guide should assist you with using any of those vendors. However, each vendor has different characteristics when it comes to storing code. An important consideration when choosing your source control management system for team development of portal applications is that it must support an unreserved checkout model for files. There are

numerous files in the domain and application that need to be checked into source control management but must be writable by the server.

Note: Even if your source control management tool does not directly integrate with WebLogic Workshop (see <http://dev2dev.bea.com> for information on “Visual Studio Source Control Provider for WebLogic Workshop”), you can still use it to manage your WebLogic domains and applications.

Creating a Shared Portal Domain

The first activity to be done with a source control management system is to have the development team lead create the appropriate portal domain for the group. The important consideration before creating and storing domain assets is determining the BEA home directory—where the team will install WebLogic.

The BEA Home Directory

There are a number of implications around where you create your BEA home that affect any WebLogic Workshop applications and domains that reference that location. When installing WebLogic, each developer can determine which drive and directory to install to.

When creating a new portal domain with the domain Configuration Wizard, you choose which BEA home directory you want to reference for that domain. The physical path to this directory is contained in any portal domain's `config.xml` file on each development machine and in domain batch scripts such as `startWeblogic.cmd`.

Why a Common BEA Home Matters

Team members will share these domain files using source control, so that all modifications to existing deployed applications, the addition of new applications, and other settings stored in `config.xml` and the start scripts can be shared.

Note: When development environments require different domain configuration settings, and you want to use config templating as a solution, do not store `config.xml` in source control. For more information see “[Setting up Config Templating \(config-template.xml\)](#)” on page 17.

Following are snippets from `config.xml` and `startWeblogic.cmd` that show hard-coded paths:

config.xml

```
<Application Name="JWSQueueTransport" Deployed="true"
  LoadOrder="1000" Path="C:\bea812\weblogic81\server\lib\" TwoPhase="true">
  <EJBComponent Name="QueueTransportEJB" Targets="portalServer"
    URI="QueueTransportEJB.jar"/>
</Application>
```

startWebLogic.cmd

```
set DOMAIN_HOME=C:\bea812\weblogic81\samples\domains\portal
```

If the `config.xml` and `startWebLogic.cmd` files are shared in source control, all team members must have installed WebLogic to the path shown in those files. However, there are circumstances where developers cannot install WebLogic to a particular drive, such as when they have multiple partitions with not enough space left on their C drive.

In addition, `pointbase.ini` in the domain root directory is a configuration file that may contain a `documentation.home` property that points to a hard-coded location on the file system.

[Table 1](#) shows which files in a domain contain hard-coded paths.

Table 1 Domain files with hard-coded paths

backup_config.xml
config.xml
create_db.*
installService.cmd
set-dbenv.*
setDomainEnv.*
setDomainEnvQS.*
startManagedWebLogic.*
startManagedWebLogicQS.*
startPointBaseConsole.*
startPointBaseConsoleQS.*
startWebLogic.*
startWebLogicQS.*
stopManagedWebLogic.*
stopManagedWebLogicQS.*
stopWebLogic.*
stopWebLogicQS.*
uninstallService.*
webappCompile.*
webappCompileQS.*
domain-info.xml (in /_cfgwiz_donotdelete)
startscript.xml (in /_cfgwiz_donotdelete)

The next section contains strategies to employ when not all team members can use the same BEA home.

If all team members *can* use the same BEA home, you may skip [“Creating and Sharing the Portal Domain” on page 7](#).

Managing Multiple BEA Home Locations for Your Team

There are a number of different techniques for sharing a common domain with team members with different BEA home directories, described in the following sections.

Option 1: Config Templating BEA Home

Config templating avoids many of the problems associated with other solutions, but it requires you to implement search and replace activities on your `config.xml` and other files. By creating a template for your `config.xml` which contains tokens that represent your BEA home directory, you can create a process to create a `config.xml` from a combination of the template and developer specified token values.

Config templating can be used for much more than setting up machines with different BEA home directories: It can provide a way for each developer to work with a separate database instance that shares a common data source configuration.

With config templating, `config.xml` is not stored in source control. For information on setting up config templating, see [“Setting up Config Templating \(config-template.xml\)” on page 17](#).

Option 2: (Windows) Using a Common Virtual Drive for BEA Home

Windows developers may consider setting up a substitute drive letter to map to their BEA home directory.

From the command prompt, the Windows OS lets you create a virtual drive and map it to an existing drive and path using the `subst` command. Your team can configure a common virtual drive letter not currently in use and use that drive for application and domain activities.

For example, you can create a drive letter P: that maps to a directory on the C drive, such as `C:\bea812`, by executing the following command from a DOS prompt:

```
subst P: C:\bea812
```

Now, after creating a new domain, you can change all references to `C:\bea812` to `P:` in the domain files listed in [Table 1](#).

The previously listed `config.xml` and `startWebLogic.cmd` entries would now look like the following:

config.xml

```
<Application Name="JWSQueueTransport" Deployed="true"
  LoadOrder="1000" Path="P:\weblogic81\server\lib\" TwoPhase="true">
  <EJBComponent Name="QueueTransportEJB" Targets="portalServer"
    URI="QueueTransportEJB.jar"/>
</Application>
```

startWebLogic.cmd

```
set DOMAIN_HOME=P:\weblogic81\samples\domains\portal
```

Note: The new hard-coded paths do not contain \bea812, because C:\bea812 was mapped to P: using the subst command.

When you want to use the domain, switch to the P drive and go into the domain directory. If another developer installs WebLogic to D:\bea, he can simply substitute that directory for P:\ by executing subst P: d:\bea and share the same config.xml and start scripts with ease.

This virtual drive option has a number of drawbacks:

- Users must run the subst command upon each reboot, though they can type the command in a text file, save the text file with a .cmd extension, and put it in their program \Startup folder so the command runs automatically at system startup.
- The created domain and application must be run from the new virtual drive. Running the domain from the “true” install drive and path will result in errors.
- Does not support UNIX developers, though UNIX developers can use the link command.

Option 3: Using Relative Paths

If team members need to install different paths on the same drive, and the domain and application are located in a common relative path to the WebLogic directory, it is possible to change all file paths in config.xml and your start scripts to be relative. However, this solution is limited in its scope.

Assuming the domain is installed to C:\bea812\user_projects\mydomain, the previously listed config.xml and startWebLogic.cmd entries would now look like the following:

config.xml

```
<Application Name="JWSQueueTransport" Deployed="true"
  LoadOrder="1000" Path="..\..\weblogic81\server\lib\" TwoPhase="true">
  <EJBComponent Name="QueueTransportEJB" Targets="portalServer"
    URI="QueueTransportEJB.jar"/>
</Application>
```

startWebLogic.cmd

```
set DOMAIN_HOME=..\..\weblogic81\samples\domains\portal
```

Problems with this solution include:

- No ability to span multiple drives.
- Project domain directory must always be in the exact same relative location to the server, even in a deployed production environment.

Creating and Sharing the Portal Domain

The first step the team lead needs to take is the creation of the new portal domain. There are several phases of domain creation, including the possible creation of your own domain template, the domain Configuration Wizard process, initial check-in to source control, and domain configuration tasks.

Best Practices for Creating a Portal Domain to Share with a Team

- Before creating a domain with the domain Configuration Wizard, you can create a custom template that determines what is in the domain you create, and you can store that template in source control. The domain Configuration Wizard builds the domain using your template. For instructions on using the Configuration Template Builder to create a domain template, see “Creating Configuration Templates Using the WebLogic Configuration Template Builder” at <http://e-docs.bea.com/platform/docs81/configwiz/tempbuild.html>.
- When creating a new domain for team development using the Configuration Wizard, be sure to select “development mode.” For instructions on running the Configuration Wizard to create a new portal domain, see “Creating WebLogic Configurations Using the Configuration Wizard” at <http://e-docs.bea.com/platform/docs81/configwiz/index.html>.
- To avoid path length exceptions, store the domain directory in a short path. For example, <drive>:\ourDomain.
- Creating the domain and the application so that they are peers to each and share a common parent directory makes sharing them with source control management systems easier to manage.

For example, install your domain to

```
<BEA_HOME>\<WEBLOGIC_HOME>\user_projects\<PROJECT>\domain\<DOMAIN>
```

And your application to

```
<BEA_HOME>\<WEBLOGIC_HOME>\user_projects\<PROJECT>\application\<APP>
```

Note: Your paths can be shorter and outside the BEA installation hierarchy.

This approach lets you have a common root directory (%PROJECTNAME) in your source control system's project for both the domain and application.

- After you create the domain, but *before you start the server*, check the domain into source control. WebLogic server creates a number of temporary files and directories in the domain directory at server startup that you are unlikely to want in source control. [Table 2](#) lists the post-startup files to exclude.
- If you have a shared parent directory for your domains and applications, use that as your root in source control.
- To enable file logging so that WebLogic Portal messages are sent to a file instead of just the screen, add an environment variable entry to the start script. Open the `startWebLogic` script in a text editor and add a path for the `WLS_REDIRECT_LOG` file before the `if "%WLS_REDIRECT_LOG%"` line. For example:

```
WLS_REDIRECT_LOG=D:\logs\admin
```

```
if "%WLS_REDIRECT_LOG%"==" " (
```

- After establishing your initial baseline for the domain in source control, check the domain back out of source control and start the domain using the `startWebLogic` command. With the server running, you may want to configure the domain so it is ready to use for your project. Using the WebLogic Server Administration Console (<http://<server>:<port>/console>), you can set up the domain to support the development team, including the addition of needed data sources.

Common tuning activities for a development domain include setting the server logging mode to 'Info' from 'Warn' (for more verbose console output and outputting JVM messages to the console). Additionally, you may want to limit the file size of the logging. Most developers do not want endless growth of their log files.

For information on server configuration, see WebLogic Server "System Administration" at <http://e-docs.bea.com/wls/docs81/admin.html>.

After you make changes to the server configuration, check in `config.xml` (or a hand-modified version of a templated `config-template.xml`).

- Before configuring your portal application, have another developer validate the changes by checking out the domain and starting the server without error.

Excluding Domain Files From Source Control Management

Exclude the following domain files from source control:

Table 2 Domain files to exclude from source control

Path	Wildcard
/ (domain root)	config.xml (ONLY if you are using config templating. See “Setting up Config Templating (config-template.xml)” on page 17.)
/	config.xml.booted
/	config.xml.original
/	*.log
/logs	*
/portalServer/pstore/ (persistent file store for session beans)	*
/<servername>/	*.log
/<servername>/	.app_poller_lastrun
/<servername>/wlnotdelete/	*
/<servername>/internal/	*
/<servername>/ldap/	*LDAPBackup*.zip
/<servername>/ldap/log/	*
/<servername>/logs/	*

Binary Files in Source Control Management

There are a number of binary files in the WebLogic domain that need to be checked into source control management for the domain to function properly. These binary files may change over time for user-initiated reasons, automatic growth of index files, and so on. For this reason, it is important that developers have a good understanding of what these files are, why they change, and when to check them in and out. The emphasis of this section is explaining how to determine when you will need to update those files in source control management.

Examples of binary files that are updated are LDAP, security, and database configuration files.

Working with Binary Files

With all binary files, there is a consistent process to follow when you make changes to them so they can be shared in source control. Changes to binaries should be initiated by a single user, typically the team lead. This reduces the chances of merge conflicts over the project lifecycle. To modify domain binary files in source control:

1. Stop the server.
2. Do a clean checkout of the binary files from source control to ensure you are working from a common base.
3. Start the server.
4. Make your changes.
5. Stop the server.
6. Check-in any modified binary files to source control management.
7. Test a clean checkout from another machine.

Users, Groups, Roles, and Entitlements – Updating LDAP

A common activity in development is the creation of a base set of users that are used to test the system. By default, WebLogic stores user, group, role, and entitlement information in an LDAP server provided by BEA. This LDAP server persists its data store to the file system in the `<domain>/ldap` directory.

For information on BEA's LDAP server, see "Managing the Embedded LDAP Server" at <http://edocs.bea.com/wls/docs81/secmanage/ldap.html>.

As the LDAP server contains information that needs to be shared by team members, check the files in the LDAP directory into source control, excluding backup and log files (see [Table 2](#)). During project development, there may be occasion to modify the existing users, groups, roles, and entitlements. You can configure users, groups, roles, and entitlements with the WebLogic Administration Portal and check in the updated LDAP files to source control.

For instructions on using the WebLogic Administration Portal, see "Getting Started with Portal Administration" at <http://e-docs.bea.com/wlp/docs81/startadm/index.html>.

Other Security Information

Other important security files located in the domain are the `SerializedSystemIni.dat`, `DefaultAuthenticatorInit.ldif`, `DefaultAuthorizerInit.ldif`, and `DefaultRoleMapperInit.ldif` files, which contain essential security information needed to start the domain. While not typically modified during the course of development, these files must exist for the server to be started. The `boot.properties` file in the domain root contains encrypted username and password information for starting the domain. That file is not mandatory, but it is typically used in development environments to allow server startup without requiring authentication.

Databases

Portal stores much of its configuration information in the database, and there are occasions where development teams need to share access to this configuration. However, WebLogic Portal does not support running multiple instances of a portal server against the same single database or database schema. Although the default database for a WebLogic Portal domain is PointBase, it is recommended that an enterprise-quality database be used for most development efforts. This is key because moving data between heterogeneous databases can be a labor-intensive manual effort. See [“Developing Against an Enterprise-Quality Database” on page 12](#).

PointBase

When creating a new portal domain, an instance of a PointBase database is created that is persisted in the root directory of the domain. New domains can also create the database objects necessary for an enterprise-quality database. For details on configuring an Oracle, SQL Server, DB2 or Sybase database, see the Database Administration Guide at <http://edocs.bea.com/wlp/docs81/db>. For details on creating new domains, see *Creating WebLogic Configurations Using the Configuration Wizard* at <http://edocs.beasys.com/platform/docs81/configwiz/index.html>.

The database contains a number of tables that store base WebLogic Portal and WebLogic Workshop data. For a description of the database objects for each component of WebLogic Portal, see the “Data Dictionary” at <http://edocs.bea.com/wlp/docs81/db/4Schemas.html>. In addition, WebLogic Workshop stores some of its internal state in the database. The topic “How Do I: Configure WebLogic Workshop to Use a Different Database for Internal State?” at <http://edocs.bea.com/workshop/docs81/doc/en/workshop/guide/howdoi/howConfigWLWToUseADifferentDBForInternalState.html> in the WebLogic Workshop help system addresses how to move that internal state store to another database.

PointBase Development Considerations

PointBase uses two files to persist the database to the file system: `workshop$1.wal` and `workshop.dbn`. Since the database is persisted to the file system, sharing copies of the database can easily be accomplished using source control management. However, PointBase files grows incrementally over time when PointBase is used, which means that the files always appear to have been modified by the user. Over time, the PointBase files can grow from about 3 MB to 10 or more. Developers need to be aware that they should not check in the database unless they explicitly are making changes to the underlying data directly through the WebLogic Administration Portal or the PointBase console. When a change does need to be made, there is a process to follow to keep the size of the updates to a minimum, which is outlined in the following paragraph steps:

To make changes to the database:

1. Stop the servers (WebLogic and PointBase).
2. Do a clean checkout of the binary files from source control to ensure you are working from a common base.

This is especially important as your PointBase files may have grown significantly since the last checkout, so a new checkout will reduce the size of those files before making your additions.

3. To modify those files in source control, follow the procedure in “[Working with Binary Files](#)” on page 10.

Knowing When You Are Making Changes to PointBase – In general, most activities that are accomplished using the WebLogic Administration Portal are persisted to the database, with the exception of user, group, and entitlements, which are persisted to LDAP. As the guide to *Deploying Portal Applications* points out at <http://e-docs.bea.com/wlp/docs81/deploy/index.html>, most of the work with the WebLogic Server Administration Console is done in a staging or production environment. However, there may be times when you want to develop against a portal desktop or have test users with user property values (the values are stored in the database).

Developing Against an Enterprise-Quality Database

Rather than share the PointBase database between developers as a binary file, it is common for each developer to work against their own unique instance of the portal database using Oracle, SQL Server, or another enterprise-quality database.

The same enterprise quality DBMS used in production should be used for development. In other words, if you plan to deploy your application on Oracle you should develop your application on Oracle as well.

Note: For Oracle and DB2, a separate database schema for each developer on a development database instance is recommended. For Sybase and SQL Server, a separate database and database log file for each developer on the development database instance is recommended.

There are several advantages to using this methodology: greater performance, easier maintenance of a baseline of data (with proper support from a database administrator and scripts), and the ability to propagate database data from a production or staging environment to a development environment using the WebLogic Portal Propagation Utility, assuming the production or staging environments use the same type of database as the development environment.

Each development machine is configured to use a specific database, contained in `config.xml`, which is a shared file in source control management. “[Setting up Config Templating \(config-template.xml\)](#)” on page 17 can help provide some mechanisms for allowing developers to share `config.xml` while still pointing to their unique database instance.

Sharing Information Using Unique Enterprise Quality Database Instances

To share information, a database administrator sets up a process where a developer can snapshot his database schema. This snapshot can then be applied to other developer schemas as part of a process that those developers can initiate. Snapshots of partial pieces of the database, or the storing of a common set of DDL scripts, are also common practices.

For a description of the database objects for each component of WebLogic Portal, see the “Data Dictionary” at <http://edocs.bea.com/wlp/docs81/db/4Schemas.html>.

In addition, WebLogic Workshop stores some of its internal state in the database. The following topic in the WebLogic Workshop help system, “How Do I: Configure WebLogic Workshop to Use a Different Database for Internal State?” addresses how to move that internal state store to another database:

<http://edocs.bea.com/workshop/docs81/doc/en/workshop/guide/howdoi/howConfigWLWToUseADifferentDBForInternalState.html>.

Creating and Sharing the Portal Application

After configuring the portal domain, the team lead needs create a new portal application. There are several phases of application creation, including creating the application and any number of portal Web projects with WebLogic Workshop and initial check-in to source control.

For instructions on creating a new portal application and Web project, see “Creating a Portal Application and Portal Web Project” at <http://edocs.bea.com/workshop/docs81/doc/en/portal/buildportals/addPortalApp.html>.

Be sure to install any services necessary to your application, such as Commerce and Pipeline, as well as any necessary tag libraries in each portal Web project, such as Commerce and Webflow (for compatibility with legacy portal Web applications).

Where to Create the Workshop Application

As mentioned in the domain configuration section, creating the domain and the application so that they are peers to each and share a common parent directory makes sharing them in source control management systems easier to manage.

For example, install your domain to: `<drive>:\ourDomain`
and your application to: `<drive>:\ourApp`

This approach lets you have a common root directory (%PROJECTNAME) in your source control system’s project for both the domain and application.

Note: To avoid path length exceptions, store the domain and application directories in a short path. For example, `<drive>:\ourDomain`.

Checking in the Workshop Application

Once the Workshop application has been constructed, the team lead should check the application into source control. This should be done before doing a build of the application, because there are a number of files which are created during a build which should not be checked into source control.

Excluding Portal Application Files From Source Control Management

Exclude the following application files from source control:

Table 3 Application files to exclude from source control

Path	Wildcard
/ (portal application root)	Each “EJB Project” contains a <code>.jar</code> file. These should be excluded.
/	<code>.beabuild.txt</code>
/APP-INF/lib/	Each “Java Project” contains a <code>.jar</code> file. These should be excluded.

Table 3 Application files to exclude from source control (Continued)

Path	Wildcard
/<project>/WEB-INF/ .pageflow-struts-generated/	*
/ .workshop/	*

Files that are excluded from source control include compiled JARs, temporary configuration files, and the output directory for workshop.

Managing Checkouts of the Workshop Application

The fundamental idea when working with source control management and a WebLogic Workshop application is that developers should be able to check out the application, initiate a build, and start the server without error.

A background on what is created when a build is executed is important to review. When a build is initiated in Workshop, a number of JAR files and temporary directories are created inside the WebLogic Workshop application itself. These files are listed in [Table 3](#). In addition, WebLogic Workshop creates some additional files such as stateless session beans for controls in the `.workshop` directory of the application. These stateless beans often have names like `TimerControl_-1n8kn2z7skxv`.

When the application is deployed to the domain by WebLogic Workshop, it is registered in `config.xml`. This deployment happens automatically when the server is started and the application is built. At this point, the application is added to `config.xml` in a new XML block. The following example shows the block added to `config.xml` for an application named `portalpm`.

```
<Application Name="portalpm" Path="P:\user_projects\applications"
  StagingMode="nostage" TwoPhase="true">
  <WebAppComponent Name="portalpmAdmin" Targets="portalServer"
    URI="adminPortal.war"/>
  <EJBComponent Name="content.jar" Targets="portalServer"
    URI="content.jar"/>
  <EJBComponent Name="content_repo.jar" Targets="portalServer"
    URI="content_repo.jar"/>
  <WebAppComponent Name="portalpmDatasync" Targets="portalServer"
    URI="datasync.war"/>
  <EJBComponent Name="netuix.jar" Targets="portalServer"
    URI="netuix.jar"/>
  <EJBComponent Name="p13n_ejb.jar" Targets="portalServer"
    URI="p13n_ejb.jar"/>
```

```

<EJBComponent Name="prefs.jar" Targets="portalServer"
  URI="prefs.jar"/>
<WebServiceComponent Name="portalpmTool" Targets="portalServer"
  URI="wps-toolSupport.war"/>
<EJBComponent Name="wps.jar" Targets="portalServer" URI="wps.jar"/>
<ApplicationConfiguration Name="portalpm" Targets="portalServer"/>
</Application>

```

As WebLogic Workshop will update the `config.xml` for the domain automatically, it is not necessary to check in a `config.xml` that contains the application name XML block. Instead, a developer checks out the application, does a build, and starts the server against a domain without this application reference. His application is then deployed to `config.xml` with all the required references to the newly built application components. If the application name XML block is checked in with `config.xml`, WebLogic Workshop will automatically update it if necessary to add or remove components.

Note about config templating – If you are using config templating ([page 17](#)), the basic `config.xml` has already been created on each development machine using the config template, and application modifications or additions are correctly added to each developer's `config.xml` file.

There are two other files that store the components for the application: `application.xml` and `weblogic-application.xml`, which are found in the application's `META-INF` directory. These files need to be shared in source control, and when new components are added to the application (such as a new EJB), the updated `application.xml` and `weblogic-application.xml` must be checked back in to source control.

Portal Coding Practices

This section provides best practices guidance for portal application source code storage.

Java Projects

If you have a number of general-purpose Java libraries that will be used by your portals, it is recommended that they be stored in a Java project inside the portal enterprise archive. This enables portability of your Java libraries across multiple instances of the server and is a convenient mechanism for packaging libraries for reuse.

Cross-Platform Support

When coding to develop and deploy in a cross-platform environment, you observe the following best practices:

- Do not use spaces in filenames.
- Keep filenames short (older versions of TAR do not support long filenames).
- Use forward slashes in paths when possible.
- Be aware of the difference between case-sensitive operating systems (Unix) and case-preservative operating systems (Windows). For example, you could create a file called `myPortletContent.jsp` and specify the file `MyPortletContent.jsp` as the Content URI on windows without problems. However, when this same application is deployed on UNIX, an error that the file `MyPortletContent.jsp` cannot be found will be generated.

Definition Labels for Portal Components

As you add books, pages, and portlets to portal desktops in WebLogic Workshop, a Definition Label (unique ID) is automatically generated for each component (which appears in the Property Editor window). With multiple developers creating new portal components, it is possible that different components can have the same automatically generated Definition Labels. To avoid duplicate Definition Labels, manually change the Definition Label for each new component using your own naming conventions.

Cluster Configuration

Any code you write should be tested often in a clustered environment. Also important is keeping session data to a manageable size and configuring your Web applications to support session sharing across the cluster. For clustering information, see “Deploying Portal Applications” at <http://e-docs.bea.com/wlp/docs81/deploy/index.html>.

Setting up Config Templating (config-template.xml)

When working in a team development environment, your team members need to work with the same `config.xml` file so that all modifications to existing deployed applications, the addition of new applications, and other settings stored in `config.xml` file can be shared. However, there are configuration settings that may need to vary from user to user. The most common variations are when developers are using different BEA home directories or their own database instances. Developers typically have different database logins and need different settings for their JDBC connection pools.

Config templating addresses this problem by letting you distribute a templated domain configuration file, `config-template.xml`. Developers share this `config-template.xml` file through source control.

Exclude config.xml From Source Control

When you need unique domain configurations in each developer environment, *exclude the config.xml file from source control*. Check in `config-template.xml` instead.

You can use ant or other script language to set up a process that copies the `config-template.xml` over the `config.xml` file. Next, certain strings in the `config.xml` file are replaced with strings that the user defines in a properties file.

For example, we can start with a `config-template.xml` file that contains the following:

```
<JDBCConnectionPool DriverName="weblogic.jdbc.oci.Driver"
    MaxCapacity="10" Name="Arcadia"
    Properties="user=ARCADIAUSER;password=ARCADIAPASSWORD;server=arcadia"
    RefreshMinutes="10" Targets="myserver"
    URL="jdbc:weblogic:oracle"/>
```

and a particular user has a `local.configtemplate.properties` file with the following two entries:

```
ARCADIAUSER=john
```

```
ARCADIAPASSWORD=mypassword
```

After running the replacement process, that user ends up with a `config.xml` file that reads:

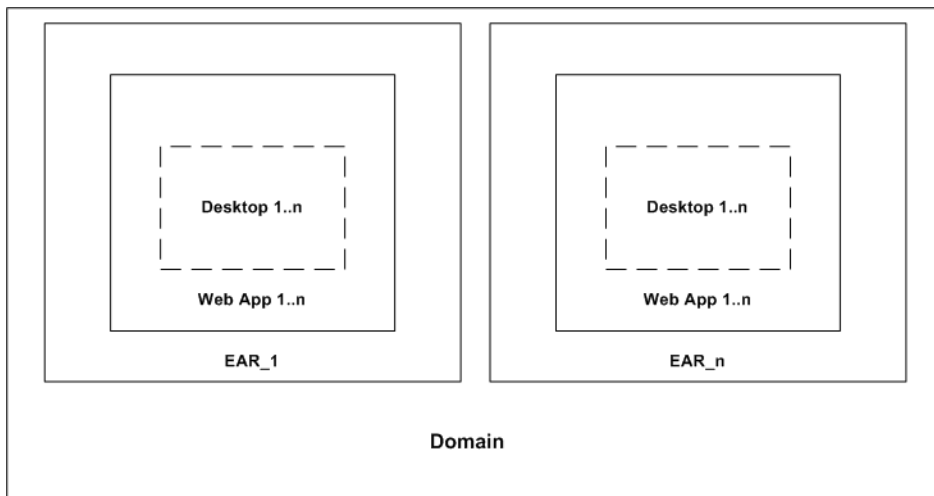
```
<JDBCConnectionPool DriverName="weblogic.jdbc.oci.Driver"
    MaxCapacity="10" Name="Arcadia"
    Properties="user=john;password=mypassword;server=arcadia"
    RefreshMinutes="10" Targets="myserver"
    URL="jdbc:weblogic:oracle"/>
```

Each user need only keep his own copy of the `local.configtemplate.properties` file, which should *not* be checked into source control. A file named `configtemplate.properties` should be distributed in source control to serve as an example of a valid `local.configtemplate.properties` file.

Using Multiple Enterprise Applications in a Single Domain

You can create and run multiple enterprise applications in a single-cluster domain. As shown in [Figure 0-4](#), a single domain can host multiple enterprise applications (EARs). Each EAR deployment can host multiple web applications, and any number of desktops can be created based on the web applications. The web applications and desktops associated with one enterprise application (EAR) are not dependent on those in another enterprise application (they are decoupled).

Figure 0-4 Multiple Enterprise Applications in a Single Domain



The following restrictions apply to this configuration of multiple enterprise applications in a single domain:

- **Database** – If you place multiple enterprise applications in a single domain, all applications must share the same database. The use of multiple databases in this configuration is not supported.
- **Resource names** – Names cannot conflict. For example, for each deployment, web application names must be unique. This applies as well to `context-root` names and their associated `CookieName` names. For each enterprise application, the WebLogic Administration Portal shows all of the portal web applications that are deployed to the server.

Note: Each enterprise application must be managed through its respective Administration Portal. Some domain-level resources, such as content, users, and groups, can be viewed and managed across enterprise applications from a single Administration Portal; however, be aware that data in one application may be cached, and updates to the same data made from another application's Administration Portal may not be immediately visible.

- **Content** – If you are deploying multiple enterprise applications within the same domain, and plan to use content management's library services for each application, you must configure each repository datasource (one per repository) to be XA-enabled. For more information regarding XA connections, see <http://edocs.bea.com/platform/docs81/configwiz/examples.html#1074297>.

Note: Content for each enterprise application is managed through its respective Portal Administration tool and Virtual Content Repository. Virtual Content Repositories (as well as Portal Administration tools) are unique to each application cannot be shared.

- **Personalization** – The same local property sets cannot be shared between multiple enterprise applications. If common properties must be shared among different enterprise applications, then use Unified User Profile (UUP). Another alternative is to copy and deploy the same property sets to multiple applications.