Creating Web Portals with BEA WebLogic

HOWARD BLOCK, ROB CASTLE, AND DAVID HRITZ

Apress™

Creating Web Portals with BEA WebLogic

ISBN (pbk): 1-59059-069-4

Printed and bound in the United States of America 12345678910

Technical Reviewer: Gregory Smith
Editorial Directors: Dan Appleman, Gary Cornell, Jason Gilmore, Simon Hayes, Martin Streicher, Karen Watterson, John Zukowski
Managing Editor: Grace Wong
Project Manager and Development Editor: Tracy Brown Collins
Copy Editor: Ami Knox
Compositor: Susan Glinert
Artist and Cover Designer: Kurt Krames
Indexer: Valerie Robbins
Production Manager: Kari Brooks
Manufacturing Manager: Tom Debolski

Distributed to the book trade in the United States by Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, NY, 10010 and outside the United States by Springer-Verlag GmbH & Co. KG, Tiergartenstr. 17, 69112 Heidelberg, Germany.

In the United States, phone 1-800-SPRINGER, email orders@springer-ny.com, or visit http://www.springer-ny.com.

Outside the United States, fax +49 6221 345229, email orders@springer.de, or visit http://www.springer.de.

For information on translations, please contact Apress directly at 2560 9th Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax: 510-549-5939, email info@apress.com, or visit http://www.apress.com.

The source code for this book is available to readers at http://www.apress.com in the Downloads section.

CHAPTER 2

Building a Portal from Scratch

"What is an enterprise application? What is a portal? How do I build all the configuration files that are required for enterprise applications? How do I put all the pieces together?" These may be some of the questions that you are asking yourself. If so, rest assured—WebLogic has gone a long way to make this process easier and more manageable.

This chapter is the starting point for putting it all together. By the end of this chapter, you will have a working enterprise application running in the WebLogic Portal Server. Although this application will be an admittedly simple one, it will feature many of the common components that your Web applications require.

This chapter is meant to give you the big picture. Here we explain the architecture of WebLogic Portal applications. We review key portal concepts and the physical implementation of these concepts including directory structures and configuration files used in a WebLogic Portal application. We also discuss the steps to creating a portal application.

WebLogic Portal Architecture

In this section, we discuss several key concepts that will enable you to understand the context in which a WebLogic Portal application will run. Each portal Web application may have one or more components used to implement the functionality for the portal. These components may include portlets, Java classes, Enterprise JavaBeans (EJBs), etc.

An enterprise application is the highest-level component that can be contained within a portal domain. A portal Web application must be deployed within an enterprise application. One or more portal Web applications may be deployed in the context of an enterprise application. In turn, one or more enterprise applications may be deployed in a portal domain.

As we mentioned in the previous chapter, a *portlet* is the component of a portal Web application that is used to build functionality into your application. In this chapter, we show you how to build your first portlet and deploy it in the sample portal application.

Figure 2-1 illustrates the relationships between portal domains, enterprise applications, portal Web applications, and application components.

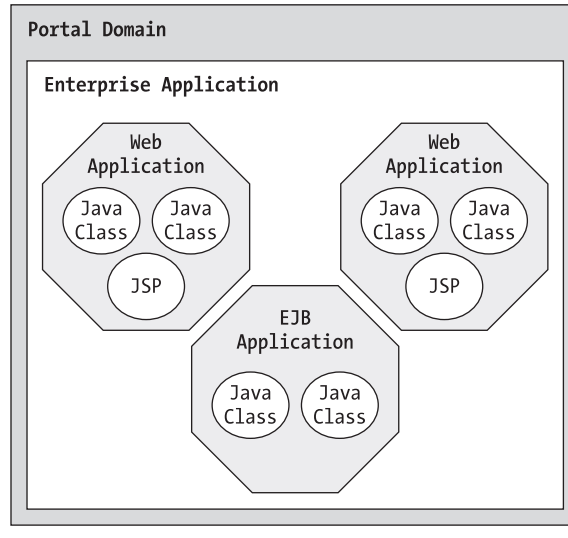

Figure 2-1. The portal domain

WebLogic Domains

Any discussion of the architecture of a WebLogic Portal must include the concept of domains. A *portal domain* is one implementation of a WebLogic Domain. A *WebLogic Domain* is a grouping of applications, application components, JDBC connection pools, servers, and other objects. What items or objects are placed under the umbrella of a portal domain is a matter to be determined by the portal architect; however, certain items must be considered when designing a domain implementation:

- If an application is deployed within a certain domain, all objects or components used by that application must also be deployed in the same domain.

- If you configure a cluster within a portal domain, all servers in the cluster must be configured as a part of the portal domain.

WebLogic Domain configurations are managed and monitored using a *WebLogic Administration Server*. You may run your applications on the Administration Server, and for the purposes of the examples in this book that is what we will do. For production environments, however, it is recommended that you run your Administration Server on a separate machine. WebLogic Administration Servers are the central point for managing and monitoring a WebLogic Domain configuration.

The physical implementation of a WebLogic Portal Domain is constructed in the following way.

WebLogic Portal Domain configuration information and other required files are stored in the root directory for the portal domain on the Administration Server.

Table 2-1 shows the required files for a WebLogic Portal Domain.

Table 2-1. Required Files for a WebLogic Portal Domain

FILE OR DIRECTORY	COMMENT
portal domain root/config.xml	The config.xml file is the persistent configuration repository for the domain. When the Administration Server starts up, it reads the configuration information for the domain from the config.xml file. When a managed server within the domain starts up, it retrieves the domain configuration information from the Administration Server for the domain.
portal domain root/fileRealm.properties	This file stores the security information for the default security realm, as well as storing users, passwords, groups, and Access Control List (ACL) information for the file realm. In particular, this file stores information about the system user, which is the user required to boot the server.
portal domain root/SerializedSystemIni.dat	This file is used in conjunction with the fileRealm.properties file by the file realm. If you copy the fileRealm.properties file from one of the "reference" domains to create a new domain, the SerializedSystemIni.dat file should be copied along with it.
portal domain root/logs directory	This directory is used to output portal server log files, such as weblogic.log, which is a log of server activity.

Enterprise Application

An *enterprise application,* which is a Java 2 Platform, Enterprise Edition (J2EE) concept, consists of a grouping of Web applications and EJBs. In a WebLogic Portal Server, an enterprise application may be deployed in its expanded form or as a J2EE enterprise application archive (.ear file). The expanded directory structure for an enterprise application mimics exactly the structure of an .ear file. We recommend that during development you deploy your applications in expanded form, and when you are ready to go to production, deploy your application as an .ear file.

Multiple enterprise applications may be deployed within a WebLogic Portal Domain. While enterprise applications may be placed anywhere in your server's directory structure by default, enterprise applications that you create are deployed in the BEA_HOME/user_projects directory. Figure 2-2 illustrates the default installation along with the sample applications that are installed by the WebLogic Portal installation.

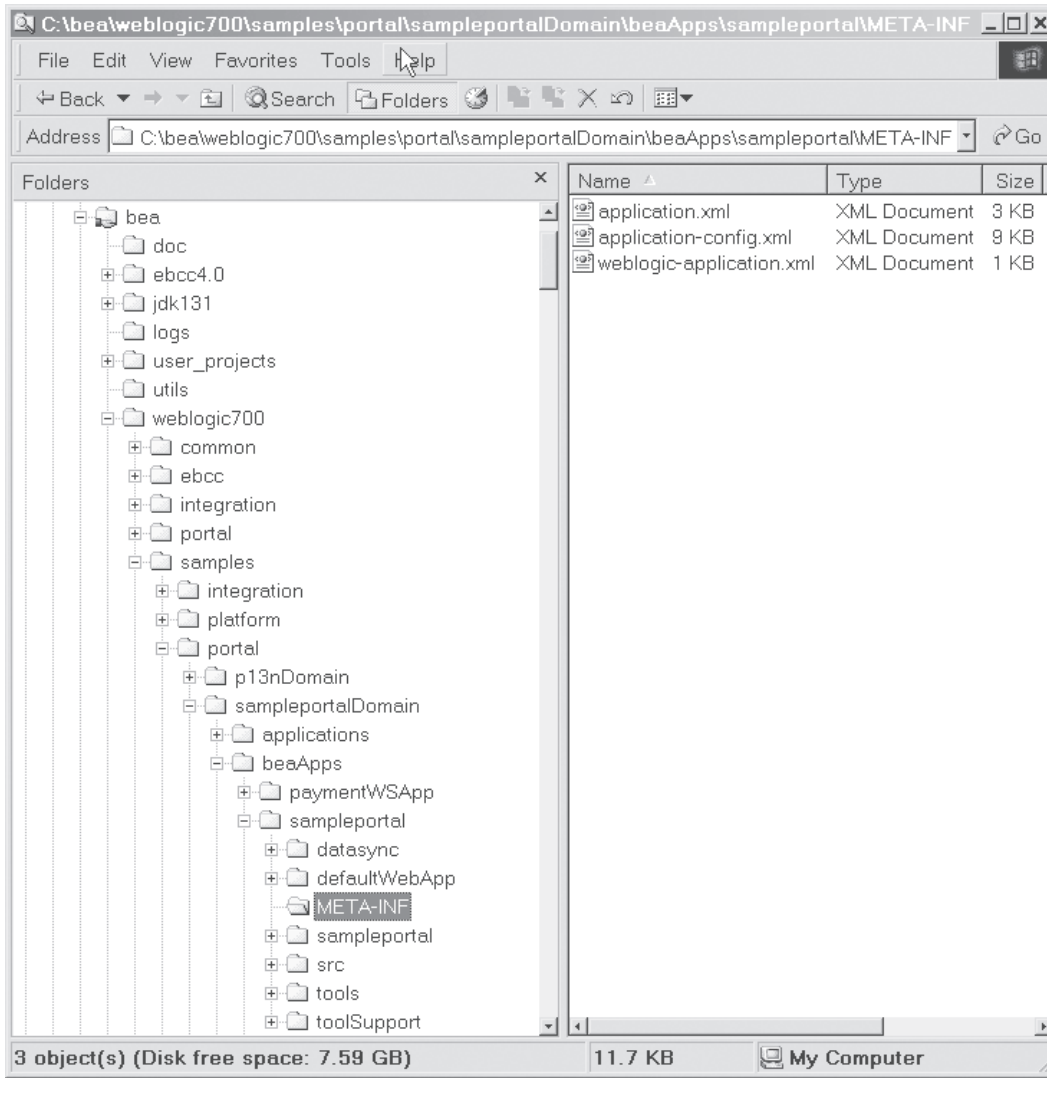

Figure 2-2. Default enterprise application installation

Notice that each enterprise application directory contains a META-INF directory. This directory contains the enterprise application deployment descriptor files, application.xml and application-config.xml. The application.xml enterprise application deployment descriptor specifies the Web applications and EJB applications that are deployed within the enterprise application and defines security roles for the enterprise application. The application-config.xml enterprise application deployment descriptor file specifies general configuration

information for the enterprise application that is used by the Administration Server. Most of the settings in these files can be configured through the server console, therefore it is not recommended that you edit these files manually. However, an understanding of these files will give you better comprehension of how an enterprise application is deployed and configured.

Listing 2-1 shows the default application.xml configuration when WebLogic Portal is installed.

Listing 2-1. Default application.xml Configuration

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

<!DOCTYPE application PUBLIC
'-//Sun Microsystems, Inc.//DTD J2EE Application 1.3//EN'
'http://java.sun.com/dtd/application_1_3.dtd'>

<application>
  <display-name>Sample Portal</display-name>
  <description>BEA Sample Portal Application</description>

  <module>
    <web>
      <web-uri>defaultWebApp</web-uri>
      <context-root>defaultWebApp</context-root>
    </web>
  </module>

  <!-- Portals -->
  <module>
    <web>
      <web-uri>sampleportal</web-uri>
      <context-root>sampleportal</context-root>
    </web>
  </module>

  <!-- Tool Stuff -->
  <module>
    <web>
      <web-uri>tools</web-uri>
      <context-root>sampleportalTools</context-root>
    </web>
  </module>
  <module>
```

```
  <web>
    <web-uri>datasync</web-uri>
    <context-root>sampleportalDataSync</context-root>
  </web>
</module>
<module>
  <web>
    <web-uri>toolSupport</web-uri>
    <context-root>sampleportalTool</context-root>
  </web>
</module>

<!-- P13N Modules -->
<module>
  <ejb>document.jar</ejb>
</module>
<module>
  <ejb>ejbadvisor.jar</ejb>
</module>
<module>
  <ejb>events.jar</ejb>
</module>
<module>
  <ejb>mail.jar</ejb>
</module>
<module>
  <ejb>pipeline.jar</ejb>
</module>
<module>
  <ejb>placeholder.jar</ejb>
</module>
<module>
  <ejb>property.jar</ejb>
</module>
<module>
  <ejb>rules.jar</ejb>
</module>
<module>
  <ejb>usermgmt.jar</ejb>
</module>
```

```
<!-- Commerce Modules -->
<module>
  <ejb>catalogws.jar</ejb>
</module>
<module>
  <ejb>customer.jar</ejb>
</module>
<module>
  <ejb>ebusiness.jar</ejb>
</module>

<!-- Campaign Modules -->
<module>
  <ejb>campaign.jar</ejb>
</module>

<!-- Portal Modules -->
<module>
  <ejb>portal.jar</ejb>
</module>

<!-- Modules used by portals -->
<module>
  <ejb>sampleportal.jar</ejb>
</module>
<module>
  <connector>BlackBoxNoTx.rar</connector>
</module>


<!-- wlcs and ebusiness -->
<security-role>
  <description>Registered customers with role CustomerRole</description>
  <role-name>CustomerRole</role-name>
</security-role>

<!-- tools -->
<security-role>
  <description>Portal Administrators</description>
  <role-name>SystemAdminRole</role-name>
</security-role>
```

```
  <security-role>
    <description>Portal Administrators</description>
    <role-name>DelegatedAdminRole</role-name>
  </security-role>

  <!-- ebusiness -->
  <security-role>
    <description>Anonymous access</description>
    <role-name>AnonymousRole</role-name>
  </security-role>

  <security-role>
    <description>Administrative role for ebusiness</description>
    <role-name>AdministrativeRole</role-name>
  </security-role>

</application>
```

Four types of modules can be specified under the application element: <web>, <ejb>, <connector>, and <java>.

The <connector> descriptor element specifies a J2EE Connector Architecture (JCA) component within the enterprise application. JCA is a J2EE 1.3 standard for allowing enterprise information systems to be integrated into a J2EE environment. The <java> descriptor element specifies a Java client application. The <web> descriptor element specifies a Web application component. The <ejb> descriptor element specifies an Enterprise JavaBean component. For our purposes, we will focus on the Web and EJB enterprise application components.

Module descriptions for an enterprise application require <web-uri> and <context-root> elements to describe them. The <web-uri> element specifies the location of the module relative to the enterprise application root within the server's directory structure. The <context-root> describes the base URL path that the server will accept to access the functionality of the deployed Web application. For example, <context-root>myportal</context-root > would correspond to a URL of `http://<server>:<port>/myportal/...` .

NOTE *In the sample application.xml, the Web modules are described with a <web-uri> of a directory (see <web-uri>sampleportal</web-uri>). As with enterprise applications, WebLogic allows you to deploy enterprise application modules in either exploded form or archived form. To deploy a Web module in archived form, simply specify the archive name for the <context-root> node.*

The <security-role> element defines security roles that are applicable to the enterprise application. Security roles defined here are WebLogic users or groups that are required for authorization in the enterprise application.

Enterprise Java Bean

Enterprise JavaBeans are a J2EE standard for deploying distributed business functionality. Notice the entry for describing EJB modules in the deployment descriptor in the previous section:

```
<module>
  <ejb>document.jar</ejb>
</module>
```

Coding an EJB and creating an EJB jar is a bit involved. For that reason we discuss EJBs further in Chapter 10; however, describing and deploying an EJB that is already built is as easy as deploying any other enterprise application module. For now, it is important to know that an EJB is another module that may be deployed within an enterprise application.

Portal Web Application

A *portal Web application* is what could be considered a child application to an enterprise application. Each enterprise application may have many portal Web applications. A portal Web application is a grouping of related business functionality. The WebLogic Portal provides several supporting Web applications that you can deploy in your enterprise application along with your portal Web application to provide enhanced functionality to your application. We discuss these applications and the functionality that they provide later.

A WebLogic Portal application has a deployment structure that follows the structure identified by the J2EE specification. Like an enterprise application, a WebLogic Portal Web application may be deployed in either exploded form or in archived form. An archive for a Web application is called a Web application archive, or .war file.

Each Web application, whether in exploded form or in archived form, has a WEB-INF directory as shown in Figure 2-3.

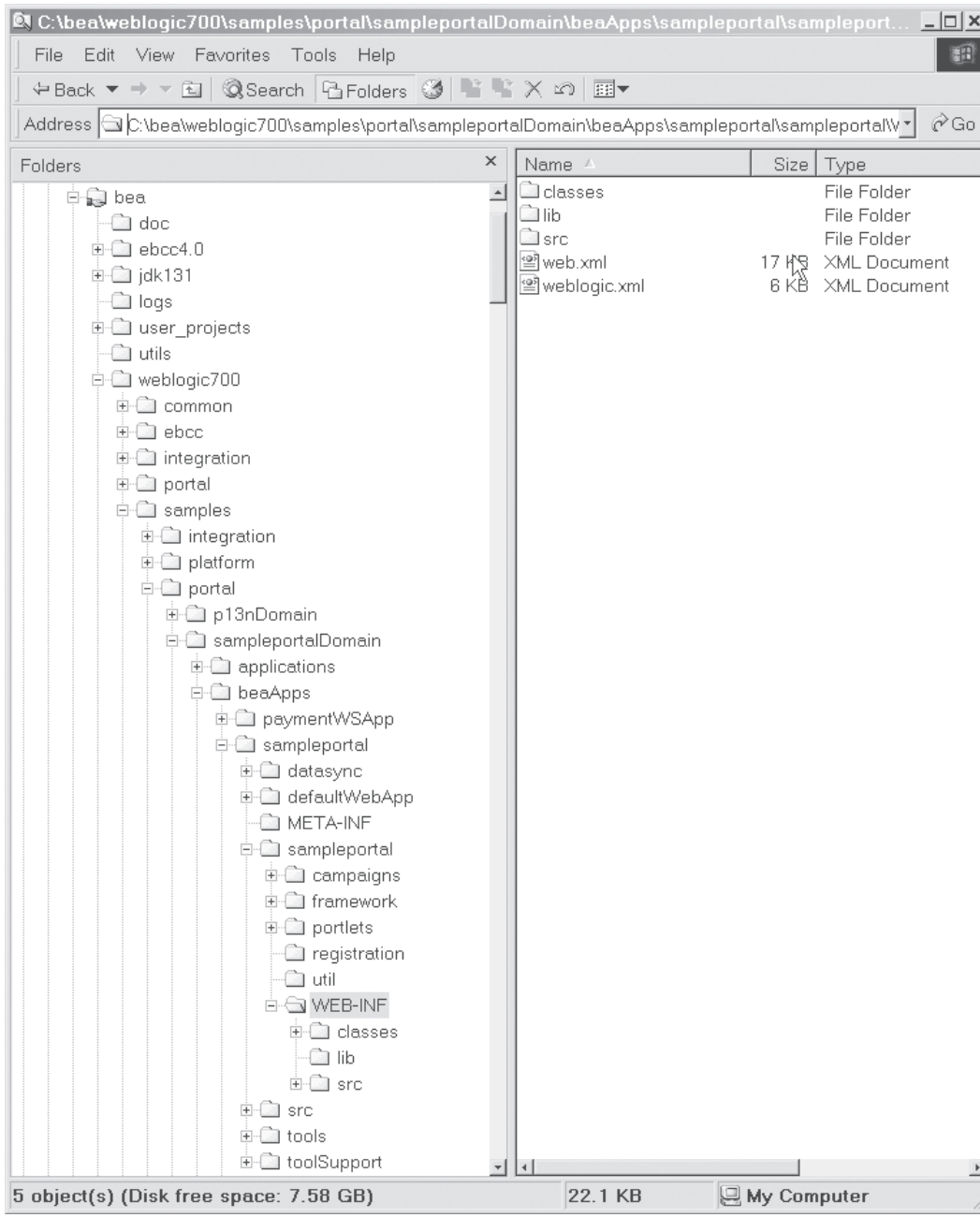

Figure 2-3. WEB-INF directory

The following four directories appear under the WEB-INF directory (these are covered in more detail in the upcoming sections):

- _tmp_war...
- classes
- lib
- src

In addition, WEB-INF contains the application descriptor files.

_tmp_war... Directory

Remember that we said a Web application could be deployed in either exploded format or in archived format. However, when WebLogic runs the Web application, it employs this directory to format and store temporary application files in a structure that it uses for running applications.

classes Directory

The WEB-INF directory also includes a classes directory containing classes required for the Web application. The classes directory is where the compiled Java classes for the Web application are deployed. The class files under this directory must be deployed in a directory structure that corresponds to the package structure for the class.

For example, say your application uses class Foo, as illustrated here:

```
package com.mycompany.myportal;
public class Foo{
      .
      .
}
```

The compiled class file for the Foo class should be deployed in the WEB-INF directory, as illustrated by the following:

```
- WEB-INF
     -  classes
          - com
               - mycompany
                    - myportal
                         - Foo.class
```

lib Directory

The WEB-INF directory also may have a lib directory. In the lib directory, you can place jar files to be used by the Web application. For example, any tag library jars that are used in your Web application should be placed in this directory.

src Directory

The WEB-INF directory also may have a src directory. In the src directory, you can place your Java source code that you compile to classes that are located in the classes directory.

Application Descriptor Files

The WEB-INF directory also contains the web.xml or Web application descriptor file and a WebLogic-specific descriptor file, weblogic.xml.

The following is an empty Web application descriptor file from the WebLogic sampleportal Web application. Notice that as with the enterprise application descriptor file, a Web application descriptor file also has <display-name> and <description> elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>

   <display-name>sampleportal</display-name>
   <description>Sample Portal WebApp</description>
```

<servlet-mapping>

The <servlet-mapping> node allows you to map a servlet described under the <servlet> node with a URL that may be used to access the servlet after deployment. Following is an example <servlet-mapping> descriptor for the ShowDoc servlet. The <url-pattern> node specifies the URL relative to the root of the Web application for the servlet being described.

```
<servlet-mapping>
  <servlet-name>ShowDocServlet</servlet-name>
  <url-pattern>/ShowDoc/*</url-pattern>
</servlet-mapping>
```

<taglib>

The <taglib> node describes any Java Server Page (JSP) tag libraries that will be used in the deployed Web application. Following is an example <taglib> descriptor. The <taglib-uri> node specifies the filename for the tag library that is being described. The <taglib-location> node describes the location of the tag library relative to the root of the Web application.

```
<taglib>
   <taglib-uri>dam.tld</taglib-uri>
   <taglib-location>/WEB-INF/lib/dam_taglib.jar</taglib-location>
</taglib>
```

<ejb-ref>

The <ejb-ref> element describes Enterprise JavaBeans that the Web application references, and is demonstrated here:

```
<ejb-ref>
  <description>
     The EjbAdvisor for this webapp
  </description>
  <ejb-ref-name>ejb/EjbAdvisor</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <home>com.bea.p13n.advisor.EjbAdvisorHome</home>
  <remote>com.bea.p13n.advisor.EjbAdvisor</remote>
</ejb-ref>
```

<context-parm>

The <context-parm> element allows you to define parameters that are available throughout the Web application. These parameters can be accessed via the following methods:

```
javax.servlet.ServletContext.getInitParameter()
javax.servlet.ServletContext.getInitParameterNames()
```

Class Loaders and Scoping

It is important to understand how class loaders work with enterprise applications. A *class loader* is a Java mechanism that turns a class that is referenced in a Java application into an instantiated class in the Java virtual machine. The relationships between class loaders that operate in enterprise applications indicate the visibility among classes and modules in enterprise applications.

All classes, including the enterprise application classes, the Web application classes, and utility classes that the Web applications use, could be loaded from the class path at server startup using the server's class loader. However, this would mean that there would be no ability to load and unload classes without restarting the server.

The WebLogic Server uses a separate class loader to load each enterprise application. The enterprise application class loader also loads any EJBs within the enterprise application. Each Web application within the enterprise application is loaded by a child class loader of the enterprise application class loader. Figure 2-4 illustrates these relationships between class loaders. Using a separate class loader for different modules or components within the server allows the server to control the loading and unloading of classes within the server.

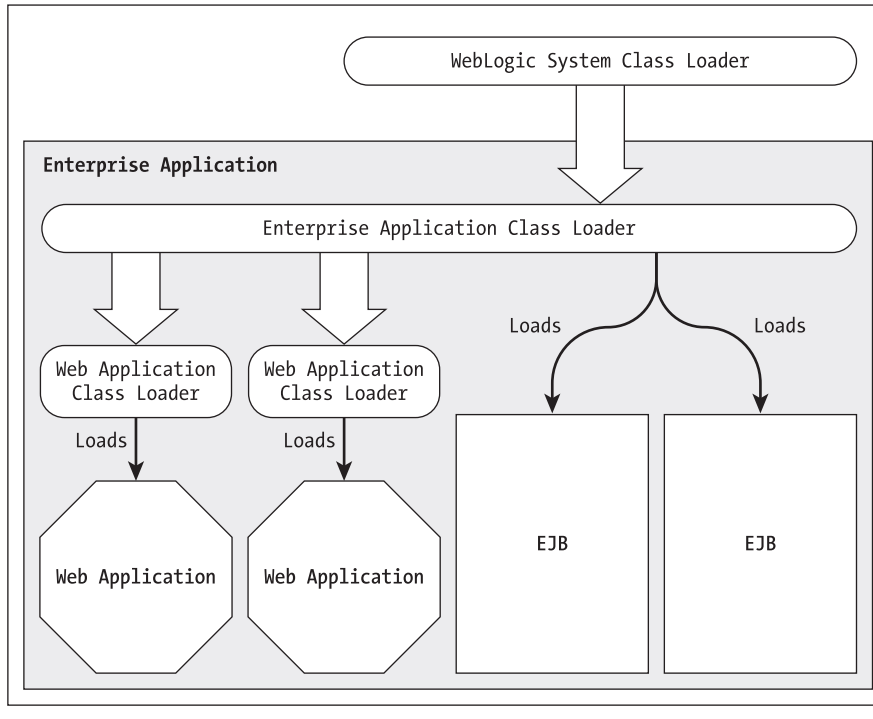

Figure 2-4. Class loader relationships

Though the relationships between class loaders are not class inheritance relationships but more like conceptual parent-child relationships between class loaders, they do indicate visibility among classes loaded by each class loader.

A class will have visibility to other classes loaded by its class loader or its class loader's parent class loader. For example, a Web application will have visibility to any classes loaded by the enterprise application. If you have defined a Web application A and an EJB application B in the enterprise application descriptor, classes running in Web application A will be able to see all EJB classes, home interfaces, and utility classes defined in EJB application B that are loaded by the enterprise application class loader.

In fact, this is a good way to make utility classes that are required by multiple Web applications available to all Web applications within an enterprise application. Even if you don't have any EJBs, you can create a dummy EJB and load it as an EJB application within the enterprise application. All Web applications within the enterprise application will then have access to the utility classes within this dummy EJB application.

Portal Metadata

The WebLogic Portal Server uses metadata stored in a database to support the functionality of the portal framework. The portal metadata stores the definition of portal pages, portlets, and other portal objects and the relationships between these objects. This allows the portal framework to put these objects together dynamically at run time instead of hard coding application data, like application content and navigation.

Though at run time the metadata is stored in the metadata database, this data is initially configured using files in the native file system. Once the metadata is configured and ready for deployment, it is then transferred to the metadata database via a process called *synchronization*.

The WebLogic Portal includes an application called the E-Business Control Center (EBCC) for configuring metadata. We discuss the E-Business Control Center more thoroughly in later chapters. In this section, we simply describe the underlying files used by the E-Business Control Center. Figure 2-5 illustrates the directory and file structure provided by the default installation of the WebLogic Portal.

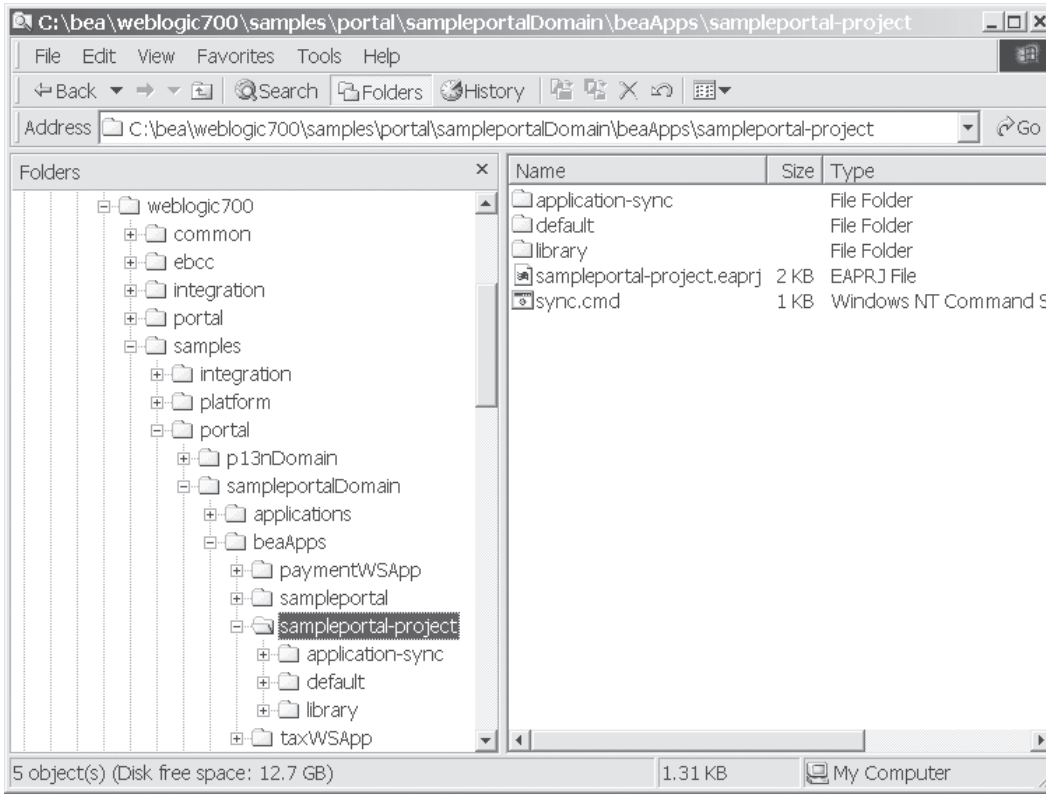

Figure 2-5. WebLogic Portal directory and file structure

As illustrated in Figure 2-5, the EBCC application is installed in the BEA_HOME/weblogic700/ebcc directory. By default, the metadata information for each enterprise application is stored in *domain root directory*\beaApps*application name*-project/application-sync. The EBCC stores all of the metadata configured for the enterprise application in XML files in this directory.

For example, the portlets directory stores metadata for portlets defined for the enterprise application. Listing 2-2 is the XML from the Dictionary.portlet file, which is a sample portlet installed with the WebLogic Portal.

Listing 2-2. Dictionary.portlet File XML

```
<?xml version="1.0" encoding="UTF-8"?>
<portlet
xmlns= http://www.bea.com/servers/portal/xsd/portlet/1.0.1
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=
"http://www.bea.com/servers/portal/xsd/portlet/1.0.1 portlet-1_0_1.xsd">
    <portlet-name>Dictionary</portlet-name>
    <is-complete>true</is-complete>
    <description>A Dictionary portlet</description>
    <content-url>/portlets/dictionary/dictionary.jsp</content-url>
    <header-url/>
    <alternate-header-url/>
    <footer-url/>
    <alternate-footer-url/>
    <titlebar-url>/framework/titlebar.jsp</titlebar-url>
    <banner-url/>
    <editable>false</editable>
    <edit-url/>
    <helpable>false</helpable>
    <help-url/>
    <icon-url>/portlets/dictionary/images/pt_dictionary.gif</icon-url>
    <minimizable>true</minimizable>
    <maximizable>true</maximizable>
    <maximize-url/>
    <mandatory>false</mandatory>
    <movable>true</movable>
    <floatable>false</floatable>
    <default-minimized>false</default-minimized>
    <login-required>false</login-required>
</portlet>
```

Portlets are one of many metadata items that can be configured via the EBCC. Once the metadata for the object is modified on the native file system and synchronized, this configuration change is, in most cases, immediately visible in the deployed application. This powerful feature of the WebLogic Portal allows you to make high-level changes to applications without code changes.

Creating a New Portal Application

The BEA WebLogic Portal provides several reference or sample enterprise applications that may be used as a starting point for new applications. You should review these sample applications to determine which application most closely represents the type of application that you want to create. Once you have determined which reference application you would like to use as a starting point for your new enterprise application, you can begin the process of creating your own application.

The BEA WebLogic Portal provides a large amount of functionality out of the box, from e-commerce functionality to content management to basic portal necessities. In the text that follows, we walk you through starting and modifying a sample portal application as a quick start. This section outlines the process of starting the sample portal and creating a new page and a new portlet as a means of building a new application.

Review/Modify Server Startup Files

You could create a new domain to run your portal enterprise application and your new portal Web application. However, here you learn how to use the sample portal, which already has an enterprise application; this enterprise application is already deployed by default in the sampleportalDomain. You will use the sampleportalDomain configuration and startup files to deploy and run a new Web application.

Earlier in this chapter we discussed WebLogic domains. Now let's look specifically at the sampleportalDomain's files and directory structure. Figure 2-6 shows the sampleportalDomain directory structure that is installed by default with the BEA WebLogic Portal.

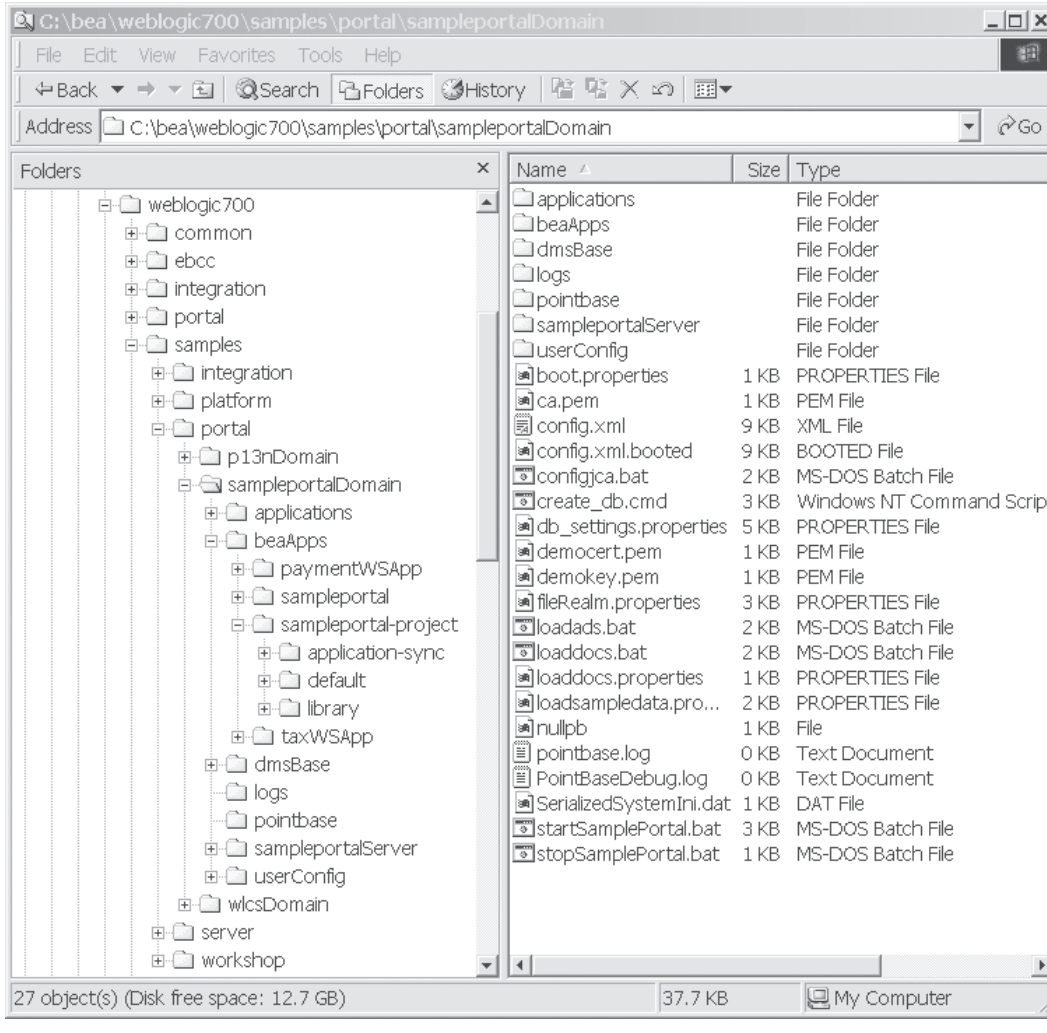

Figure 2-6. The sampleportalDomain directory structure

`startSamplePortal.bat`

We've already discussed many of the files that reside under the domain directory. Now we want to look at the startPortal.bat file. This file is the main batch file that is used to start the WebLogic Portal Server. Listing 2-3 shows the default sampleportalDomain startSamplePortal.bat file. We have included comments in this listing to point out several items that you may need to modify for your application.

Listing 2-3. Default sampleportalDomain startSamplePortal.bat File

```
@ECHO OFF
SETLOCAL

REM ##########################################################################
REM (c) 2002 BEA SYSTEMS INC. All rights reserved
REM
REM BEA WebLogic Portal Server startup script.
REM This script can also install/uninstall a Portal Window Service. Use the
REM -installService or -uninstallService command-line arguments.
REM ##########################################################################

REM ##########################################################################
REM The WLP installation directory
REM ##########################################################################
SET WLP_HOME=C:\bea\weblogic700\portal

REM ##########################################################################
REM Set the WebLogic server name
REM ##########################################################################
REM This line sets the name of the server that we are starting:
SET SERVER_NAME=sampleportalServer

REM ##########################################################################
REM Set the database type
REM Valid values are: POINTBASE, ORACLE_THIN, MSSQL, SYBASE_JCONNECT, DB2_TYPE2
REM Set set-environment.bat for more details
REM ##########################################################################
SET DATABASE=POINTBASE
REM Try to get it from the db_settings.properties file
IF not exist .\db_settings.properties goto _setenv
SET DB_SETTINGS=.\db_settings.properties
FOR /F "eol=# tokens=1,2 delims==" %%i in (%DB_SETTINGS%) do  (
    if %%i == database SET DATABASE=%%j
)
:_setenv

REM ##########################################################################
REM Set the environment
REM See set-environment.bat for more details on the available parameters
REM ##########################################################################
REM Notice the call to set-environment .bat--this file sets environment variables
```

```
REM that are used to set the class path for the server and used by the start
server
REM command. You modify this file to set JDBC drivers and other items that are
REM required at server startup.
CALL "%WLP_HOME%\bin\win32\set-environment.bat"

REM ########################################################################
REM Set any additional CLASSPATH information
REM ########################################################################
REM If you introduced new environment variables in the set-environment.bat file
REM that need to be added to the class path, these variables should be appended
REM to this line:
SET CLASSPATH=%CLASSPATH%;%P13N_DIR%\lib\commerce_system.jar;
%P13N_DIR%\lib\campaign_system.jar
REM ########################################################################
REM Start WebLogic with the above parameters.
REM See startWebLogic.cmd for more details on the available parameters.
REM ########################################################################
set MEM_ARGS=-Xms128m -Xmx128m -XX:MaxPermSize=128m
set JAVA_OPTIONS=-Dcommerce.properties="%WLP_HOME%/weblogiccommerce.properties"

if "%1" == "-installService" goto _installService
if "%1" == "-uninstallService" goto _uninstallService

:_startWebLogic
call "%P13N_DIR%\bin\win32\startWebLogic.cmd"
goto _the_end

:_installService
call "%P13N_DIR%\bin\win32\installWebLogicService.cmd"
goto _the_end

:_uninstallService
call "%P13N_DIR%\bin\win32\uninstallWebLogicService.cmd"
goto _the_end

:_the_end
ENDLOCAL
```

Set-environment.bat

Carefully review the set-environment.bat file shown in Listing 2-4. In the Windows environment this file is located in the PORTAL_HOME/bin/win32 directory. The main item that you need to modify is the specification of the database that will be used for your application. Notice in this file that the commands already exist to specify Cloudscape, Oracle, or Sybase as the database server. It is just a matter of commenting and uncommenting the appropriate lines to set up this file for your application.

However, if you need to use a JDBC driver other than the ones that are already set up in this file (e.g., Oracle thin driver), you must modify this file with commands that set up the class path and other items required for your driver.

Listing 2-4. set-environment.bat

```
@ECHO OFF
REM ###############################################################################
REM #\      (c) 2001-2002 BEA SYSTEMS INC. All rights reserved
REM #
REM #\      BEA Portal Server variable setup script.
REM #
REM ###############################################################################

REM ----------- Set the following variables appropriately -----------
REM ----------- or define them as environment variables  -----------
REM ----------- in your system properties and comment out -----------
REM ----------- the next 4 lines.                           -----------
SET PORTAL_HOME=C:\bea\weblogic700\portal
SET WL_COMMERCE_HOME=C:\bea\weblogic700\portal
SET PORTAL_LIB=%PORTAL_HOME%\lib


SET JDK_HOME=C:\bea\jdk131
SET WLCS_ORACLE_HOME=
SET SYBASE_HOME=not set
SET SYBASE_OCS=@SYBASE_OCS@
SET DB2_HOME=@DB2_HOME_BACK_SLASH@
SET BEA_HOME=C:\bea
SET WEBLOGIC_HOME=C:\bea\weblogic700\server
SET OI_HOME=not set
```

```
REM Location of the P13N platform
SET P13N_DIR=C:\bea\weblogic700\portal

REM ----------- Specify which Database Driver to use -----------
REM Valid values are: POINTBASE, ORACLE_THIN, MSSQL, SYBASE_JCONNECT,
REM DB2_TYPE2
IF "%DATABASE%" == "" SET DATABASE=POINTBASE

REM -- Add WebLogic bin directories to the path --
SET PATH=%WEBLOGIC_HOME%\bin;%JDK_HOME%\bin;%PATH%

if %DATABASE% == POINTBASE GOTO POINTBASE
if %DATABASE% == ORACLE_THIN GOTO ORACLE_THIN
if %DATABASE% == MSSQL GOTO MSSQL
if %DATABASE% == SYBASE_JCONNECT GOTO SYBASE_JCONNECT
if %DATABASE% == DB2_TYPE2 GOTO DB2_TYPE2


:POINTBASE

REM ----------- POINTBASE classes -----------

REM ---------
REM NOTE: POINTBASE_HOME should be set to BEA_HOME when we move to WLS 7.0
REM ---------
SET DB_CLASSPATH=C:\bea\weblogic700\samples\server\eval\pointbase\
lib\pbserver42ECF172.jar
SET DB_CLASSPATH=%DB_CLASSPATH%;
C:\bea\weblogic700\samples\server\eval\pointbase\lib\pbclient42ECF172.jar
SET DB_CLASSPATH=%DB_CLASSPATH%;
C:\bea\weblogic700\samples\server\eval\pointbase\lib\pbtools42ECF172.jar

GOTO continue

:ORACLE_THIN

REM
REM version 8.1.7 of the Oracle thin driver (classes12.zip) in the manifest for
REM weblogic.jar.  weblogic.jar is added to the CLASSPATH below.
REM

GOTO continue
```

```
:SYBASE_JCONNECT

REM
REM The Sybase jConnect driver is a type 4 driver that is supplied with WLS and
REM included in the manifest for weblogic.jar.
REM weblogic.jar is added to the CLASSPATH below.
REM

GOTO continue

:DB2_TYPE2
call @DB2_HOME_BACK_SLASH@\java12\usejdbc2.bat
SET DB_CLASSPATH=%DB2_HOME%\java\db2java.zip
SET PATH=%DB2_HOME%\bin;%PATH%

GOTO continue

:MSSQL

REM
REM The Microsoft SQL Server driver (Weblogic jDriver for SQL Server) is a type 4
REM driver that is supplied with WLS and included in the manifest for weblogic.jar.
REM weblogic.jar is added to the CLASSPATH below.
REM

:continue

REM ----------- JDK classes and executables -----------
SET JDK_TOOLS=%JDK_HOME%\lib\tools.jar
SET JAVA_CLASSPATH=%JDK_TOOLS%

SET WLS_CLASSPATH=%WEBLOGIC_HOME%\lib\weblogic.jar;
%WEBLOGIC_HOME%\lib\webservices.jar

REM ----------- BEA WebLogic Personalization Server classes ------------

SET WLPS_CLASSPATH=%P13N_DIR%\lib\p13n_system.jar;
%PORTAL_LIB%\portal_system.jar;
%P13N_DIR%\lib\ext\jdom.jar;
%P13N_DIR%\lib\ext\HTTPClient.jar;
%P13N_DIR%\lib\ext\wlcsparsers.jar
```

```
REM ----------- WebLogic CLASSPATH -----------
SET CLASSPATH=%WLS_CLASSPATH%;
%WLPS_CLASSPATH%;
%JAVA_CLASSPATH%;
%DB_CLASSPATh%
```

startWebLogic.cmd

Listing 2-5 shows the startWebLogic.cmd batch file. Notice that the environment variables in this file are ones that have been set by the scripts discussed previously.

Listing 2-5. startWebLogic.cmd Batch File

```
@rem ************************************************************************
@rem This script is used to start WebLogic Server
@rem
@rem To create your own start script for your domain, all you need to set is
@rem SERVER_NAME, then call this script from the domain
@rem directory.
@rem If you set DOMAIN_NAME, then this will assume WLS6.1 compability -- it
@rem pass in -Dweblogic.domain and you should invoke it from the parent
@rem directory of config\%DOMAIN_NAME%
@rem
@rem Other variables that startWebLogic takes are:
@rem
@rem DB_SETTINGS  - the db_settings.properties to use pass to
@rem                startPBServer.bat, if DATABASE==POINTBASE
@rem WLS_USER     - admin username for server startup
@rem WLS_PW       - cleartext password for server startup
@rem ADMIN_URL    - if this variable is set, the server started will be a
@rem                managed server, and will look to the url specified (i.e.
@rem                http://localhost:7001) as the admin server.
@rem WLS_PROD_MODE- set to true for production mode servers, false for
@rem                development mode
@rem WLS_MGMT_DISC- set to true for production mode servers, false for
@rem                development mode
@rem JAVA_OPTIONS - Java command-line options for running the server. (These
@rem                will be tagged on to the end of the JAVA_VM and MEM_ARGS)
@rem JAVA_VM      - The java arg specifying the VM to run.  (i.e. -server,
@rem                -client, etc.)
@rem MEM_ARGS     - The variable to override the standard memory arguments
@rem                passed to java
```

```
@rem JAVA_SEC_POLICY - The Java Security policy file to use (defaults to
@rem                   weblogic.policy)
@rem
@rem This assumes that your PATH and CLASSPATH are configured for
@rem the various native libraries and system classes required by the server.
@rem This also supports starting up a Pointbase server
@rem
@rem ***************************************************************************

@echo off
setlocal

@rem Invoke set-environment.bat if we haven't already
if "%WL_COMMERCE_HOME%"==""
call "C:\bea\weblogic700\portal\bin\win32\set-environment.bat"

@rem Now BEA_HOME, WEBLOGIC_HOME, WL_COMMERCE_HOME, and JDK_HOME should be set

@rem Check that the WebLogic classes are where we expect them to be
:checkWLS
if exist "%WEBLOGIC_HOME%\lib\weblogic.jar" goto checkJava
echo The WebLogic Server wasn't found in directory %WEBLOGIC_HOME%.
echo Please edit your set-environment script so that the WEBLOGIC_HOME variable
echo points to the WebLogic Server installation directory.
goto finish

@rem Check that java is where we expect it to be
:checkJava
if exist "%JDK_HOME%\bin\java.exe" goto runWebLogic
echo The JDK wasn't found in directory %JDK_HOME%.
echo Please edit your set-environment script so that the JDK_HOME variable
echo points to the location of your JDK.
goto finish

:runWebLogic

@rem set defaults for these, if they weren't set in either the invoker or
@rem set-environment

@rem By default the server will start with the Java hotspot setting turned on.
@rem If you have code that will not run correctly using the Java hotspot, you
@rem may want to change this by setting JAVA_VM to something different
@rem in the server's start script; however, in most cases you should use the
@rem hotspot engine to enhance the performance of the server.
```

```
if "%JAVA_VM%"=="" set JAVA_VM=-hotspot
if "%MEM_ARGS%"=="" set MEM_ARGS=-Xms200m -Xmx200m
if "%WLS_PROD_MODE%"=="" set WLS_PROD_MODE=true
if "%WLS_MGMT_DISC%"=="" set WLS_MGMT_DISC=false
if "%JAVA_SEC_POLICY%"=="" set JAVA_SEC_POLICY=%WEBLOGIC_HOME%\lib\weblogic.policy

if not "%DOMAIN_NAME%"=="" set JAVA_OPTIONS=%JAVA_OPTIONS% -
Dweblogic.Domain=%DOMAIN_NAME%

set SAVED_CLASSPATH=%CLASSPATH%
set CLASSPATH=%CLASSPATH%;%WL_COMMERCE_HOME%\lib\p13n\ejb\p13n_util.jar

@rem Start PointBase, if needed
set PBOWNER=
if not "%DATABASE%"=="POINTBASE" goto NOT_PB1

set PB_HOST=localhost
set PB_PORT=9092
set PB_DB=wlportal

@rem Get the PointBase host, port, and database name from db_settings.properties
if "%DB_SETTINGS%"=="" goto _NO_DB_SETTINGS
FOR /F "eol=# tokens=1,2 delims==" %%i in (%DB_SETTINGS%) do  (
    if %%i == host SET PB_HOST=%%j
)
FOR /F "eol=# tokens=1,2 delims==" %%i in (%DB_SETTINGS%) do  (
    if %%i == port SET PB_PORT=%%j
)
FOR /F "eol=# tokens=1,2 delims==" %%i in (%DB_SETTINGS%) do  (
    if %%i == db_name SET PB_DB=%%j
)

:_NO_DB_SETTINGS
@rem Check if PointBase is up (sets errorlevel)
"%JDK_HOME%\bin\java"
com.bea.p13n.db.internal.PointBasePing
-host %PB_HOST%
-port %PB_PORT%
-database %PB_DB%

@rem PointBase already running
IF NOT ERRORLEVEL 1 GOTO NOT_PB1
```

```
IF "%DB_SETTINGS%"=="" START /I /B "PointBase Server"
"%WL_COMMERCE_HOME%\bin\win32\startPBServer.bat"
IF NOT "%DB_SETTINGS%"=="" START /I /B "PointBase Server"
"%WL_COMMERCE_HOME%\bin\win32\startPBServer.bat"
%DB_SETTINGS%

@rem set PBOWNER so that we will shutdown down PB at the end
set PBOWNER=THIS_SCRIPT

@rem  if not using PointBase or already started, continue on...
:NOT_PB1

set CLASSPATH=%SAVED_CLASSPATH%

@rem Start Server

echo ************************************************
echo *  To start WebLogic Server, use a username and   *
echo *  password assigned to an admin-level user.  By  *
echo *  default, this is user: weblogic                 *
echo *  and password: weblogic                           *
echo *  These should both be changed using the          *
echo *  WebLogic Server console at                       *
echo *       http://[hostname]:[port]/console           *
echo ************************************************
@rem This is the line of the file that actually starts the WebLogic Portal Server.
@rem Notice that based on the disposition of the ADMIN_URL environment
@rem variable the server will either start as an Administration Server or a
@rem Managed Server.

@rem If you are restarting an Administration Server and you already have Managed
@rem Servers running, set the -Dweblogic.management.discover argument to true by
@rem setting the WLS_MGMT_DISC environment variable to true in the server's start
@rem script. The default startup mode for the server, if this parameter is not
@rem set, is to automatically try to discover Managed Servers that are running.
@rem So if you only have one server in your domain or if you are starting
@rem a Managed Server, you should explicitly set this argument to false to
@rem improve startup speed.
```

```
echo on
@if "%ADMIN_URL%" == "" goto runAdmin
"%JDK_HOME%\bin\java"
%JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
-Dweblogic.Name=%SERVER_NAME%
-Dbea.home="%BEA_HOME%"
-Dweblogic.management.username=%WLS_USER%
-Dweblogic.management.password=%WLS_PW%
-Dweblogic.management.server=%ADMIN_URL%
-Dweblogic.ProductionModeEnabled=%WLS_PROD_MODE%
 -Djava.security.policy=="%JAVA_SEC_POLICY%" weblogic.Server
@goto finish

@:runAdmin
"%JDK_HOME%\bin\java"
%JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
-Dweblogic.Name=%SERVER_NAME%
-Dbea.home="%BEA_HOME%"
-Dweblogic.management.username=%WLS_USER%
-Dweblogic.management.password=%WLS_PW%
-Dweblogic.ProductionModeEnabled=%WLS_PROD_MODE%
-Dweblogic.management.discover=%WLS_MGMT_DISC%
-Djava.security.policy=="%JAVA_SEC_POLICY%" weblogic.Server
@goto finish

:finish

@echo off
@rem ##########################################################################
@rem if PointBase was launched by this script then shut it down
@rem ##########################################################################
if not "%PBOWNER%"=="THIS_SCRIPT" goto _the_end

if "%DB_SETTINGS%"=="" call
"%WL_COMMERCE_HOME%\bin\win32\stopPBServer.bat"
if not "%DB_SETTINGS%"=="" call
"%WL_COMMERCE_HOME%\bin\win32\stopPBServer.bat"
%DB_SETTINGS%

:_the_end
endlocal
```

Start the Sample Portal

To start the sample portal, click Start ➢ Programs ➢ BEA WebLogic Platform 7.0 ➢ WebLogic Portal 7.0 ➢ Portal Examples ➢ Portal Example ➢ Launch Portal Server.

The server will start in a command window. Figure 2-7 shows the messages you will see as the server is starting. You will know that the server has finished the startup process when you see the following message:

```
<Server started in RUNNING mode>
```

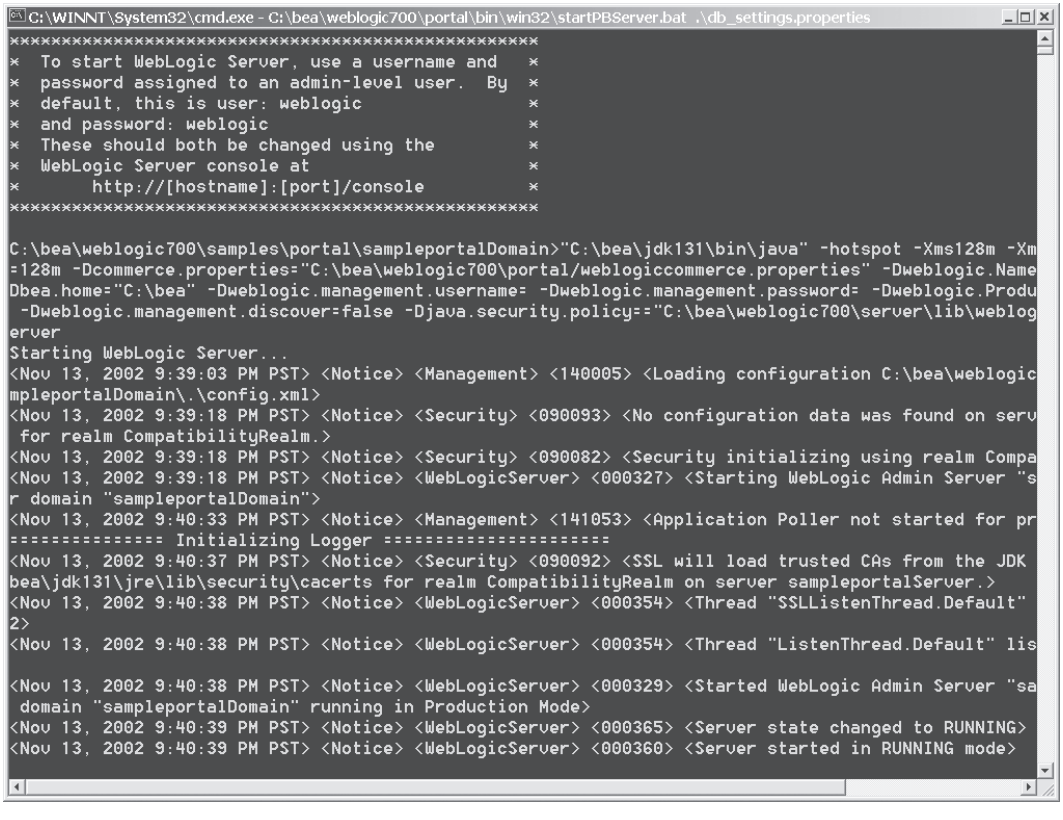

Figure 2-7. Server console

Open the Sample Portal Project

You are now ready to create and deploy the metadata for your Web application and run a test. To do this you use the E-Business Control Center. We go into more detail about the EBCC in a later chapter. For now, we walk you through some simple steps that allow you to deploy some very simple metadata for your application.

1. Open the EBCC by clicking Start ➢ Programs ➢ BEA WebLogic Platform 7.0 ➢ WebLogic Portal 7.0 ➢ E-Business Control Center.

2. Open the sample portal project by clicking File ➢ Open Project (see Figure 2-8).

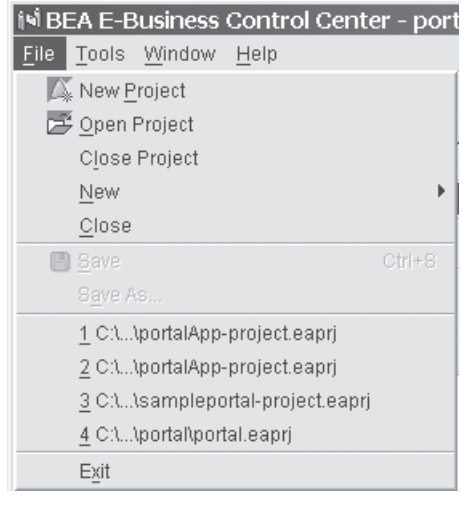

Figure 2-8. Selecting the Open Project menu item

3. Using the Open Project dialog box (see Figure 2-9), navigate to the sample portal project file (sampleportal-project.eaprj), located at BEA_HOME\weblogic700\samples\portal\sampleportalDomain\beaApps\sampleportal-project.

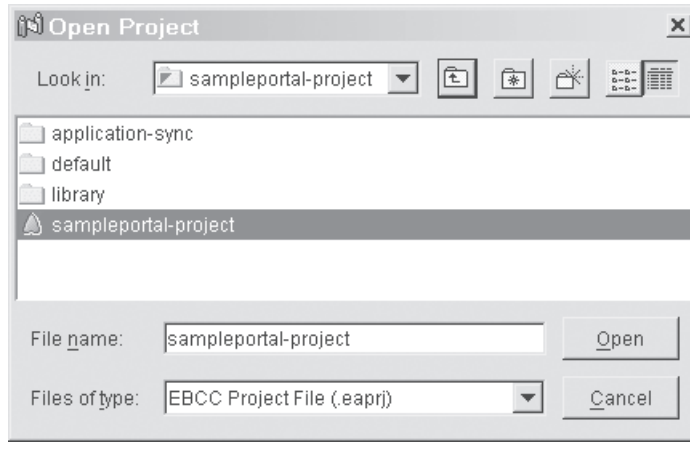

Figure 2-9. Open Project dialog box

Create the Portlet Functionality

The following steps show how to create the functionality for your portlet:

1. Create a new portlet directory called helloworld in the sample portal (see Figure 2-10).

2. Create a new JSP file in the helloworld directory that you just created.

3. Enter the following code into the helloworld.jsp file:

```
<table>
<tr>
<td class=contentheading>
Hello World this is my first portlet!
</td>
</tr>
</table>
```

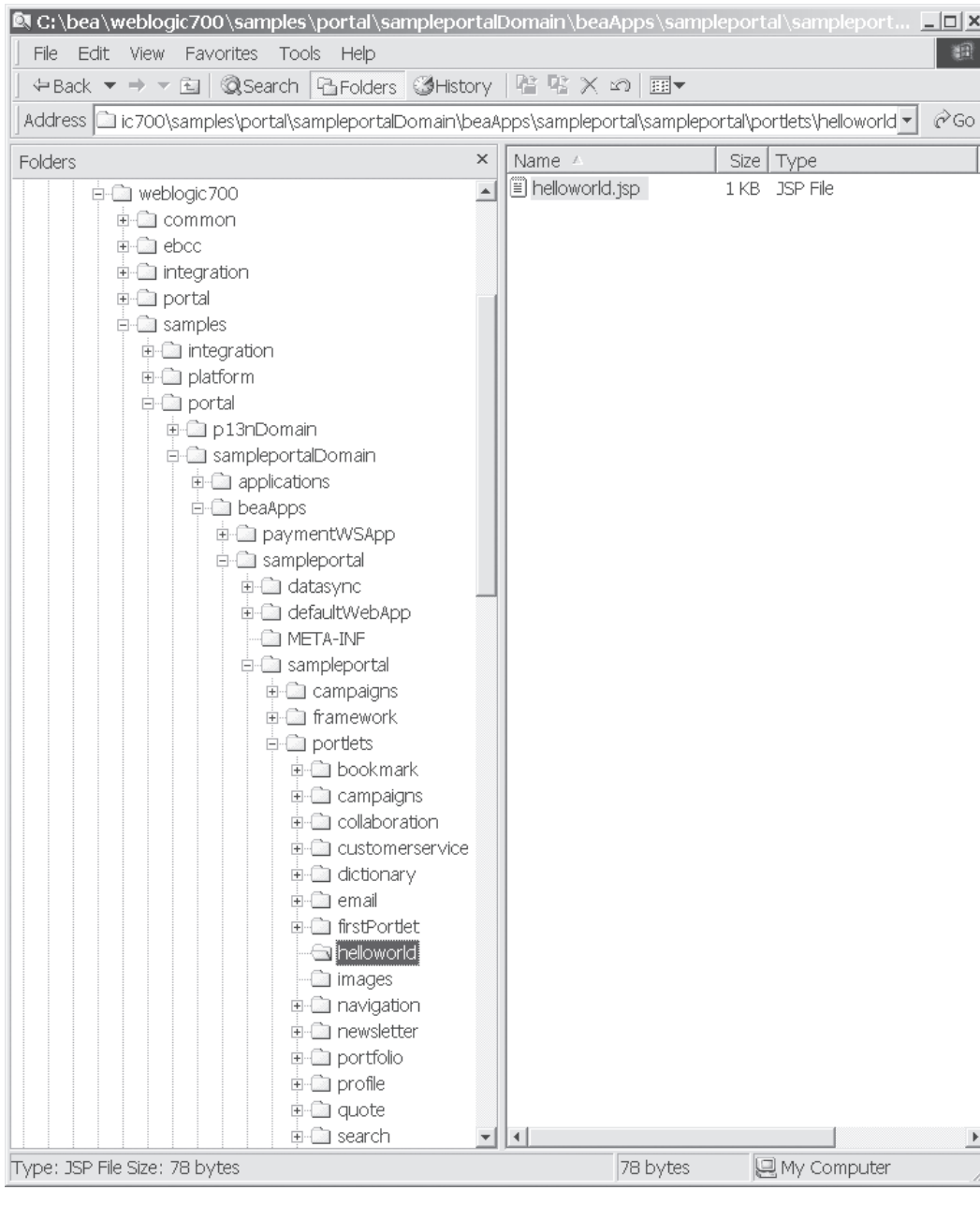

Figure 2-10. Creating the helloworld portlet directory

Create the Portlet Definition (Metadata)

The following steps show how to create the portal definition or metadata for your portlet:

1. Select the Portlet menu item in the toolbar as shown in Figure 2-11.

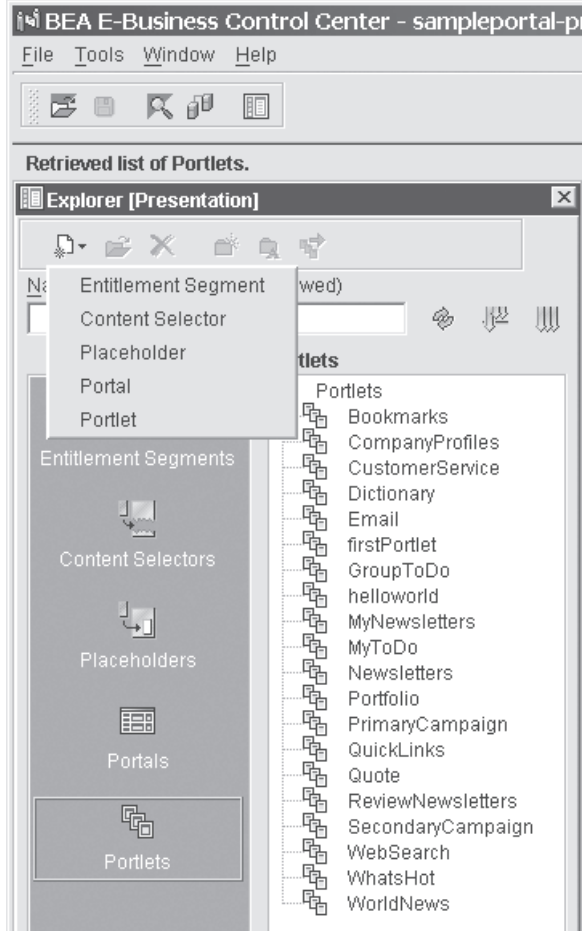

Figure 2-11. Selecting the Portlet menu item

2. Choose the "Use the Portlet Editor to create a new portlet with existing resources" radio button option in the New Portlet dialog box (see Figure 2-12).

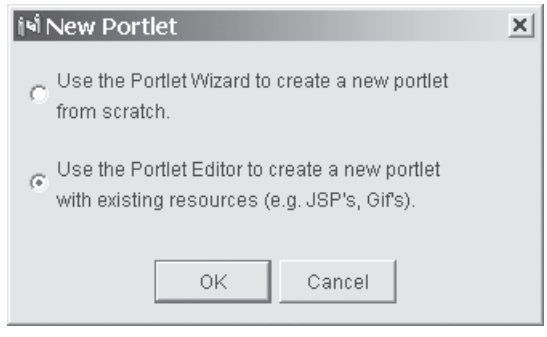

Figure 2-12. New Portlet dialog box

3. Enter the following items in the Portlet Editor (see Figure 2-13):

   Description: Hello World

   Content URL: /portlets/helloworld/helloworld.jsp

   Icon URL: (blank)

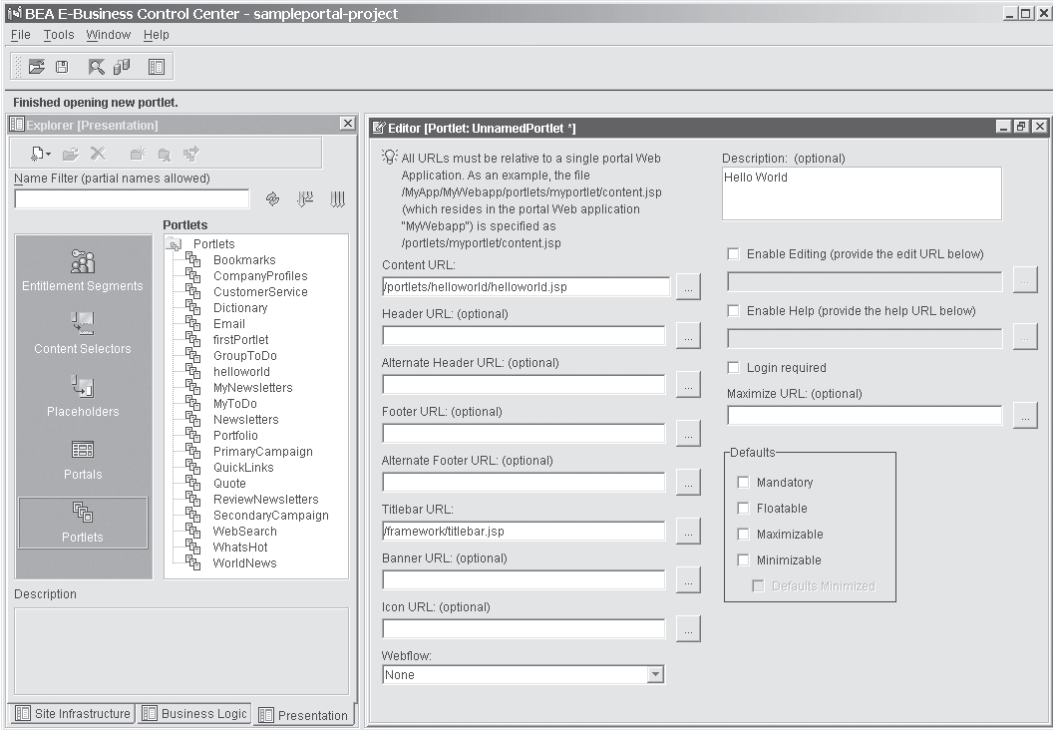

Figure 2-13. Portlet Editor

NOTE *The Icon URL is initially populated with a value of "/portlets/images". Remove this value and leave this field blank.*

4. Save the new portlet definition by clicking the File ➢ Save menu item.

5. Enter helloworld in the File name field in the Save dialog box (see Figure 2-14).

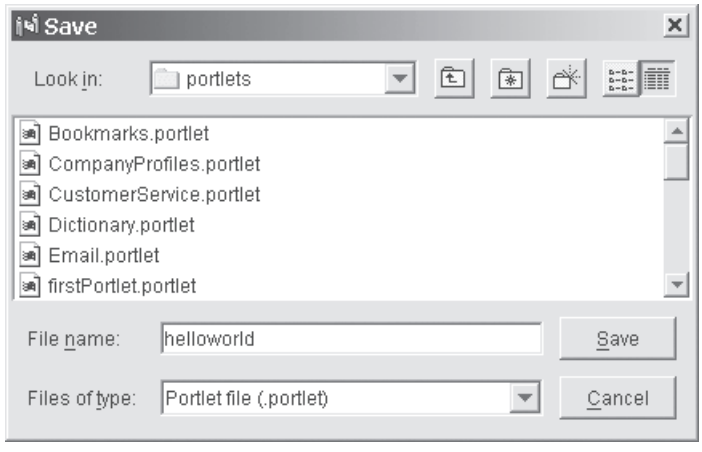

Figure 2-14. Saving your new portlet

6. Select the Portals icon from the EBCC explorer panel.

7. Double-click sampleportal in the list of portals. The Portal Editor will display in the right panel.

8. Open the General section of the Portal Editor.

9. From the Portlets tab in the Portal Editor, add the helloworld portlet to the selected portlets for this portal (see Figure 2-15).

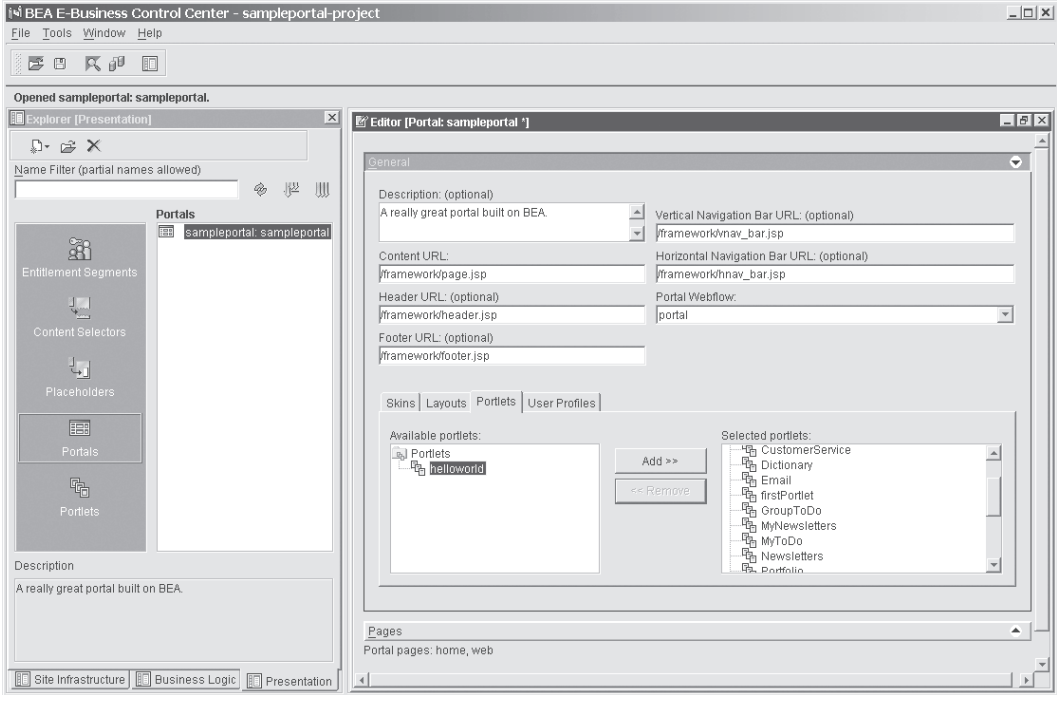

Figure 2-15. Portlets tab in the General section of the Portal Editor

10. Open the Pages section of the Portal Editor.

11. Select the home page from the list and click the Edit button.

12. Add the helloworld portlet to the Selected portlets list in the Page definition dialog box, and then click the OK button to close the dialog box (see Figure 2-16).

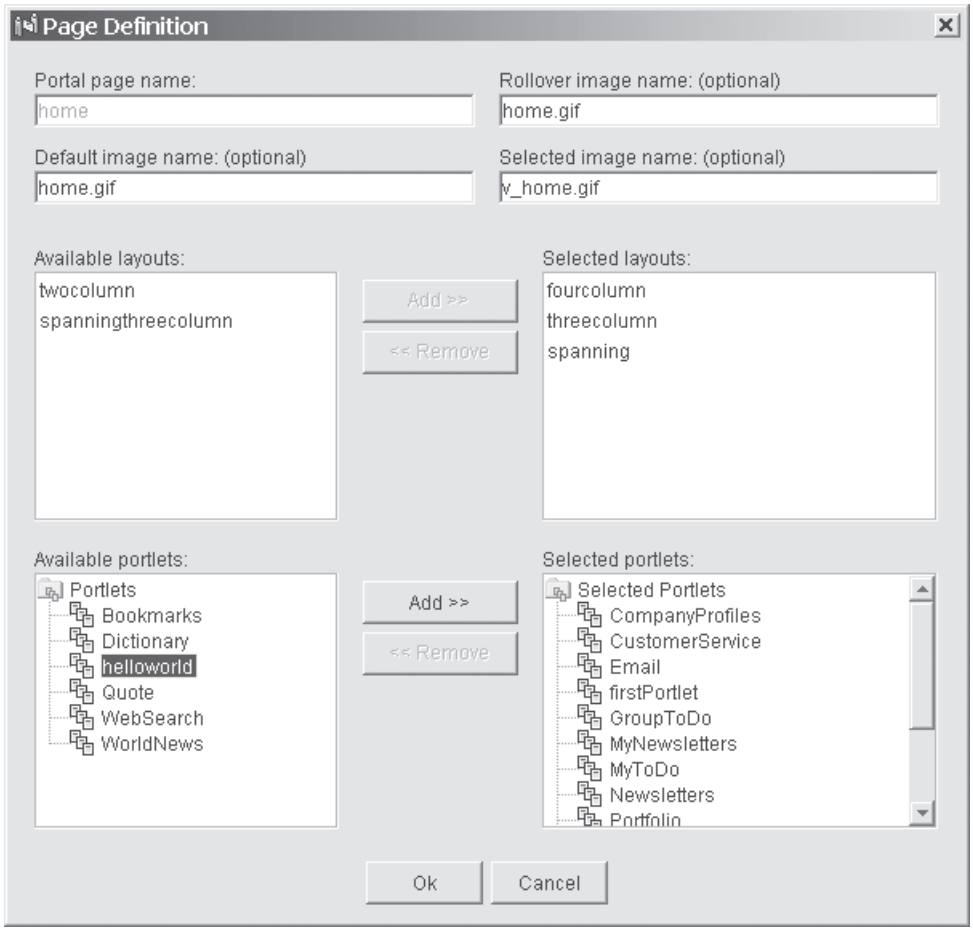

Figure 2-16. Page Definition dialog box

13. Save the portal definition by choosing Save from the File menu.

14. Select Tools ➢ Synchronize. When the synchronization process is completed, close the dialog box (see Figure 2-17). In the password dialog box, enter a valid datasync user name/password combo; from a fresh installation, the valid combos are system/weblogic, weblogic/weblogic, administrator/password.

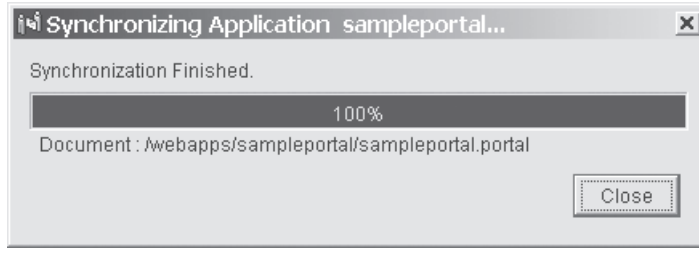

Figure 2-17. Synchronizing the sampleportal application

15. Click Cancel to close the Reset Campaigns State dialog box, which appears after the synchronization process is complete.

Make the New Portlet Available to the Home Page

Follow these steps to make your new portlet accessible from the home page:

1. Open the Portal Tools application in your browser by navigating to the following URL: `http://server:port/sampleportalTools/index.jsp`. Log in using administrator/password as the user/password combination.

2. Click the Portal Management link as shown in Figure 2-18.

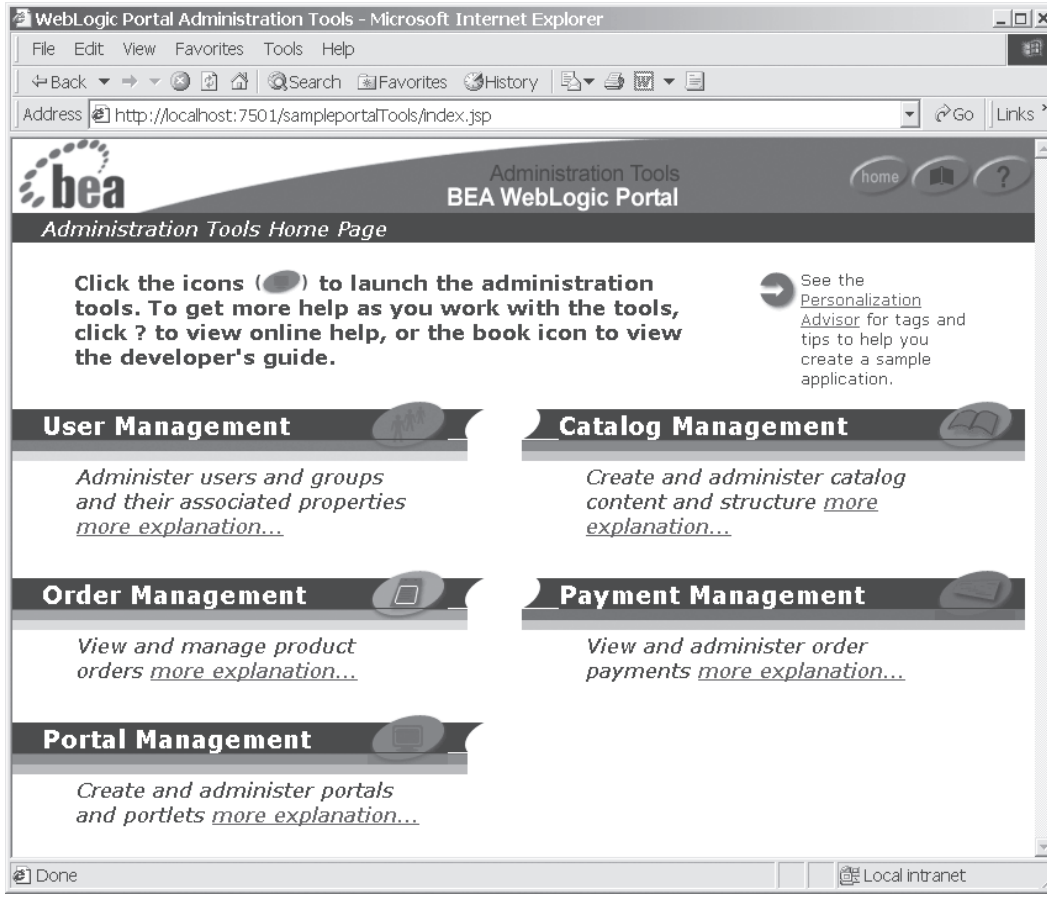

Figure 2-18. Portal Management link on the Administration Tools home page

3. Click the Avitek Portal link from the Portal Management window (see Figure 2-19).

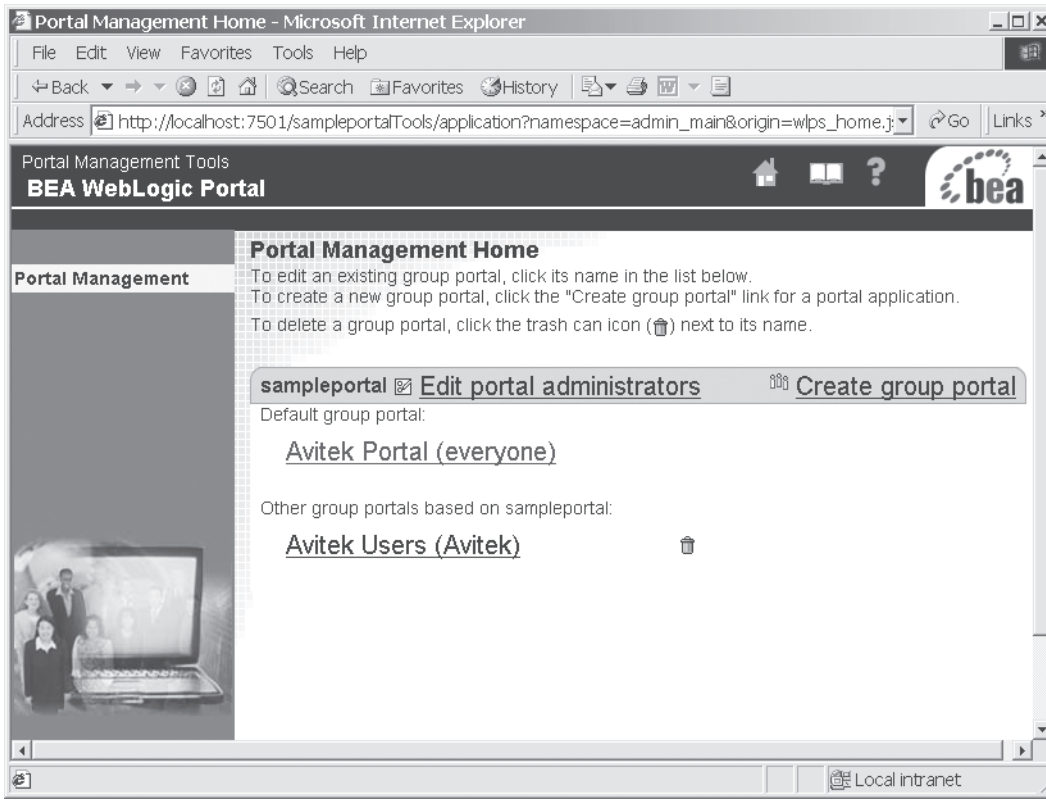

Figure 2-19. Avitek Portal link on the Portal Management page

4. Click the Manage Pages and Portlets link from the Group Portal Management page as shown in Figure 2-20.

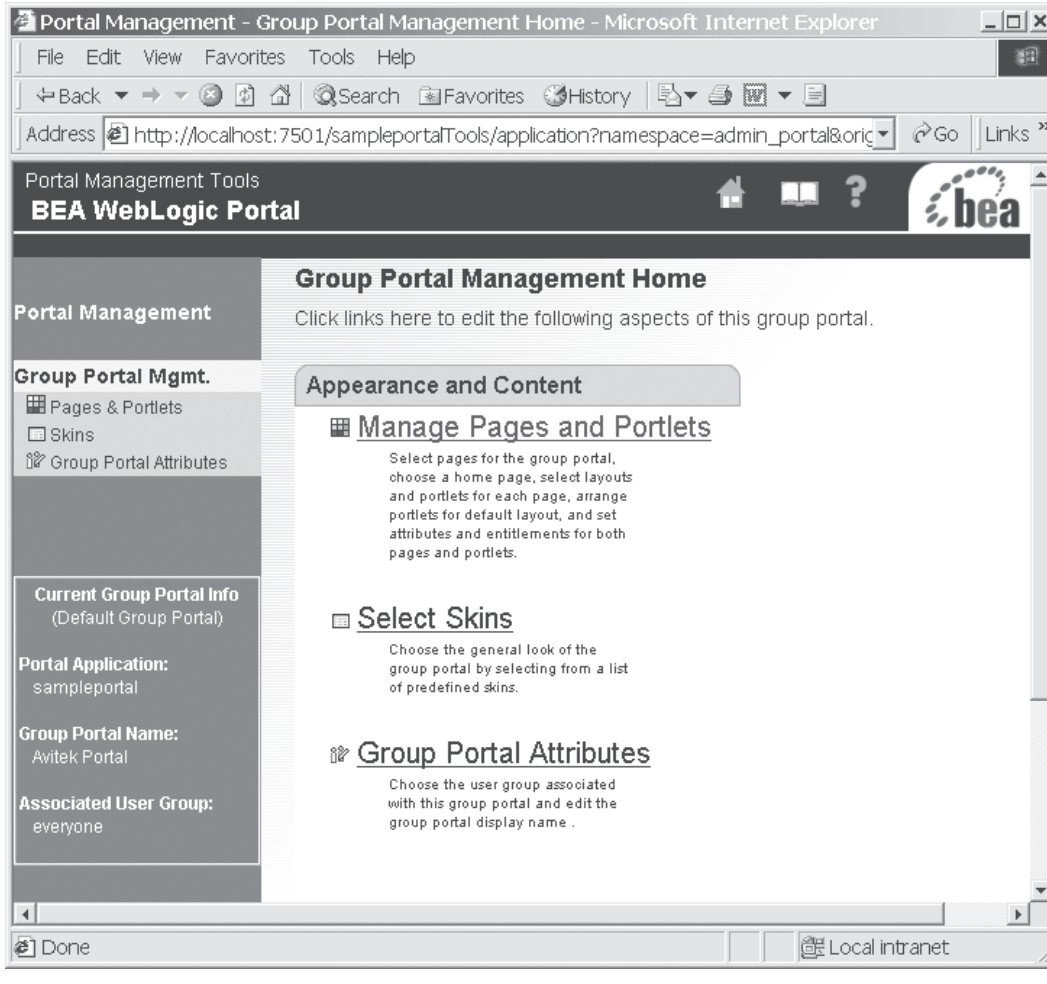

Figure 2-20. Manage Pages and Portlets link on the Group Portal Management page

5. Click the Edit Portlets button next to the home page on the Pages and Portlets page as shown in Figure 2-21.

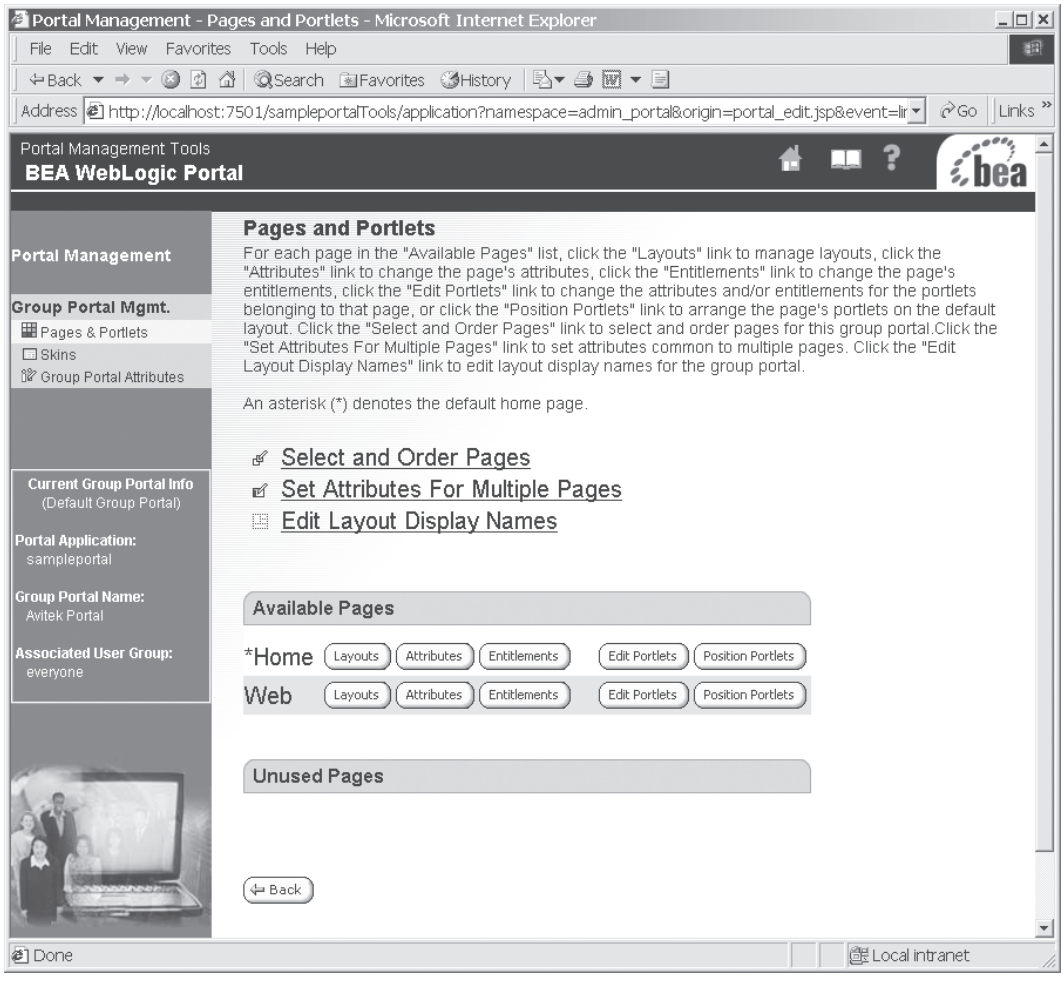

Figure 2-21. Edit Portlets button on the Pages and Portlets page

6. Select the helloworld portlet and click Set Attributes.

7. Select the Available and Visible check boxes and click the Save button as shown in Figure 2-22.

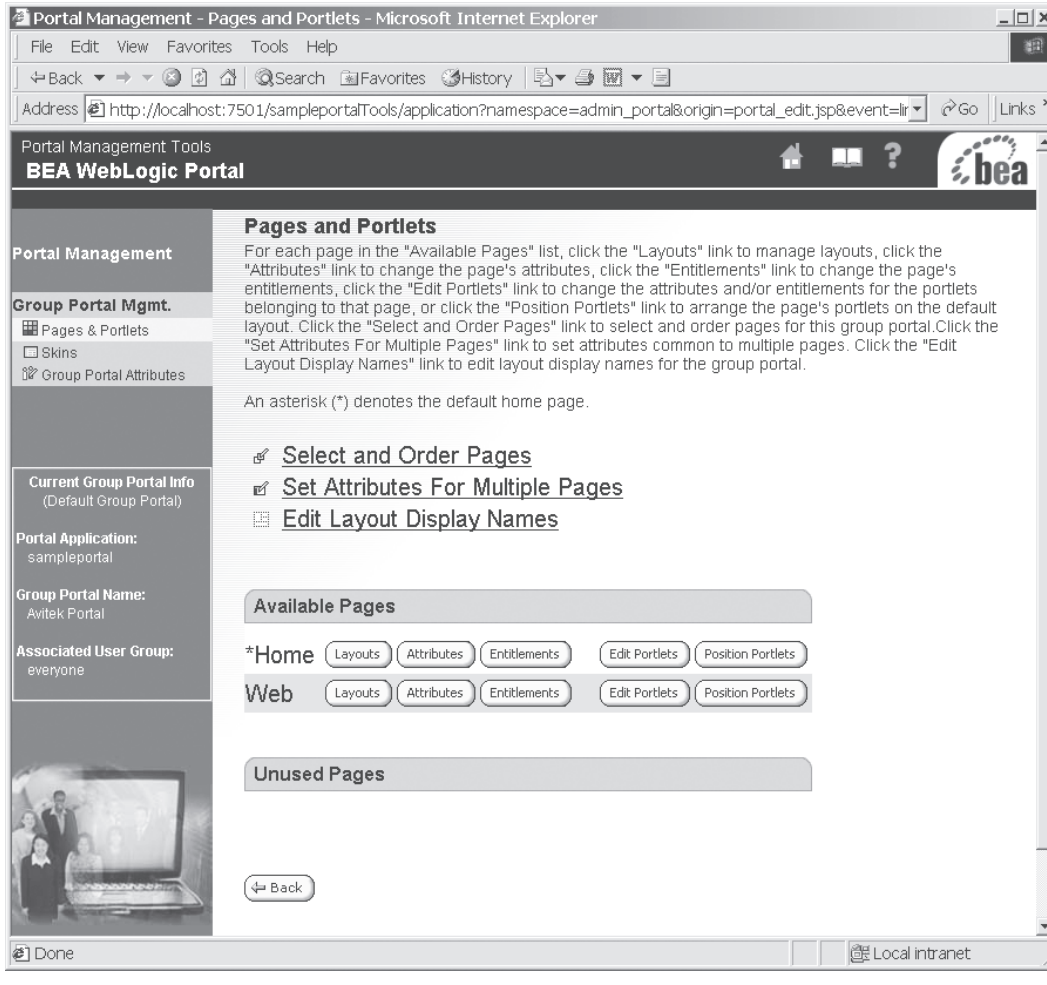

Figure 2-22. Setting portlet attributes

Test the New Functionality

You have now completed the steps required to add new functionality to the sample portal. You can test your application by browsing to `http://server:port/sampleportal` in your Web browser as shown in Figure 2-23. You should now have a very simple working portal Web application.

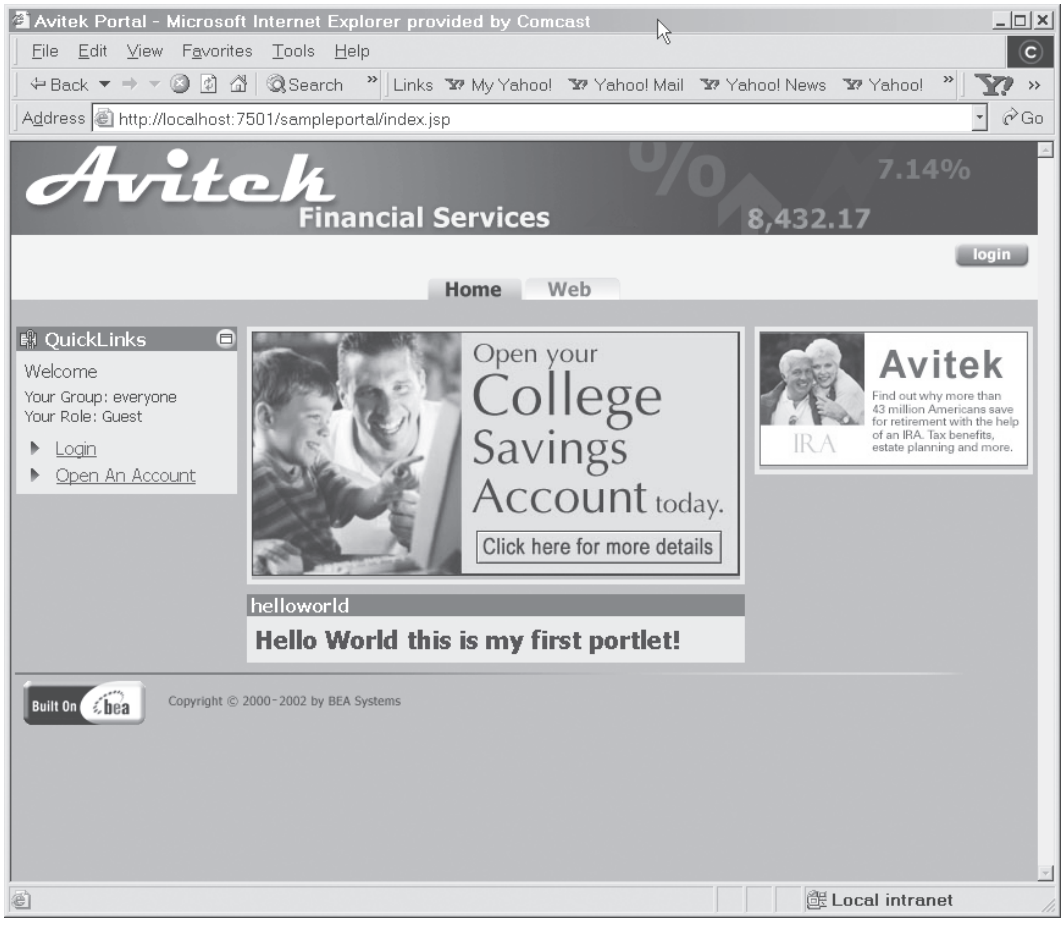

Figure 2-23. Testing your portal

Summary

This chapter explained some important portal concepts that you need to understand before moving forward. It also gave you step-by-step instructions for creating a new portal application based on the existing sample portal application supplied by the WebLogic Portal. We cover creating portal applications in more detail in later chapters, but you should now have a foundation on which you can build. At this point you should have created your own sample portal functionality following the instructions in this chapter. You now have the base knowledge that is required to create more advanced functionality and build your own portal application from scratch.