JUST GET IT WRITTEN: DEPLOYING APPLICATIONS TO WEBLOGIC SERVER USING JDEVELOPER AND WLS CONSOLE

Peter Koletzke, Quovera and Duncan Mills, Oracle

> *Don't get it right, just get it written.*
>
> —James Thurber (1893–1961),
> The Sheep in Wolf's Clothing

> *However beautiful the strategy,*
> *you should occasionally look at the results.*
>
> —Winston Churchill (1874–1965)

After you create, debug, and test the code for your Java Platform, Enterprise Edition (Java EE) web application, you will probably need to install it on a test server. The term *deployment* refers to the process of copying and installing the necessary application, library, and configuration files into a specific server runtime environment.

Although your organization may require that a group other than developers is in charge of deployments, the developer is ultimately responsible for ensuring that the deployment package is complete and bug free. Therefore, once you have completed development and unit testing for the application on a local (desktop) server, you (or someone else) will deploy the application to an environment that matches a production situation for further testing; this deployment will test the ability of your application to run on an external server. It will also test the completeness of the deployment; for example, it will help you understand if all files necessary to run the application are available and if the application's library versions work in the runtime environment.

Deployment is not a one time task in an application's life cycle. After the test deployment just mentioned, your organization may require other levels of testing (for user acceptance tests or volume tests), and these tests will require additional preproduction deployments. After all testing is complete, the application will be deployed into the production environment. But even this is not the end. When an application is enhanced or its bugs are fixed, the deployment process will need to be repeated to make the new application code available.

One objective of this white paper is to introduce deployment principles, so that you can be effective in creating (and possibly deploying) deployment files. It assumes you now have, or will soon have, a working familiarity with the Java EE runtime environment and application files.

The other objective of this white paper is to show the techniques used to deploy an application to Oracle WebLogic Server (WLS) using JDeveloper or the WLS console. Since this objective is best met with an actual demonstration, the last part of this white paper contains a hands-on practice where we show—and, hopefully have you try out—the steps required for these two deployment methods.

Deployment Principles

Deploying an application to a Java EE server requires these two stages:

1. **Prepare the deployment file** This stage consists of collecting application files and supporting libraries into a single Java archive (JAR) file sometimes called "the deployment file." So that it can be properly handled by a Java

EE-compliant server, such as WLS, the archive file must contain a standard structure with appropriate configuration files.

2. **Copy the deployment file to the application server** In this stage, you use deployment tools (JDeveloper or the WLS console) to upload the archive file to the server and then allow the server to install the application in the proper directory. The application will be available to users at the end of this process.

Java Archive Files

Java EE defines standards for deployment file packages and locations as well as for standard configuration files expected by an application server that is Java EE–compliant. Using Java EE configuration and archive files, any Java EE–compliant server can find the proper files and process them in a standard way.

> **Note**
> This white paper uses as an example an Application Development Framework (ADF) application created in JDeveloper 11g as a subject for deployment. Although ADF applications are deployable to any Java EE-compliant server, the server must support JavaServer Faces (JSF) 1.2. WLS 10.3 and beyond fulfills this requirement.

The first step in deploying an application is gathering together potentially thousands of application and library files that are located in various directories. Java EE defines standard *archive files*, single files that contain more than one file and directory. Tools such as JDeveloper help you assemble the files into these archives. Each file contains one or more files in one or more directories. A single archive file can contain the equivalent of one or more file system directories (with nested subdirectories) and files. The JVM can access files inside an archive file in the same way as if they were located in a real file system directory. Any archive file can be viewed with an archive viewer (such as the Archive Viewer in JDeveloper) and manipulated with an archive file program (such as WinZip or 7-Zip). The Java archive files used for Java EE deployment are web application archive and enterprise application archive files. In addition, another archive—metadata application archive—is required for Metadata Services (MDS) customizations in Fusion applications.

Web Application Archive (WAR)

Also called "web archive," a *web application archive*, or *WAR*, file is an archive file that contains all files required for the application's runtime. The WAR file contains all JSF files in your application with the directories set up in the ViewController project. The WAR file also contains a number of files and directories inside a WEB-INF directory. These files are a combination of standard Java EE XML descriptor files (such as web.xml) and possibly library JAR files that support the application. The deployment process expands the WAR file into its component files and directories. A copy of the WAR file is kept in the project root directory. WAR files are named with a "war" extension.

> **Tip**
> ADF applications depend on many individual JAR files which are installed as shared resources on the target application server. These libraries of shared JAR files are referenced from a common location and are not packaged within the WAR file. A list of these shared JAR files and the libraries that they are included in is available in the *Oracle Fusion Middleware Administrator's Guide 11g Release 1* (www.oracle.com/pls/as111120/portal.portal_db).

Enterprise Application Archive (EAR)

Also called "enterprise archive," an *enterprise application archive*, or *EAR*, file is an archive file used for standard Java EE deployments. It provides a single archive that contains all other archive and other files needed for an entire enterprise (many applications). The EAR file can contain one or more WAR files, JAR files, and EJB JAR files, as well as several deployment descriptor files. EAR files are named with an "ear" extension.

Metadata Application Archive (MAR)

The MAR file is an optional format that is not part of the Java EE standard and is specific to ADF based applications that contain metadata customizations. The MAR is essentially the same as an EAR file except that it contains all of the seeded

MDS customizations for an application as well. The WebLogic container has special logic to correctly install applications in a MAR file and will both install the Java EE portion of the application and publish the MDS metadata to the metadata repository. In addition to WAR, EAR, and MAR files, a deployment will likely include normal JAR files, which contain supporting library code (and use a .jar file extension).

> **Note**
> Although some Java EE applications can be deployed as a simple WAR files without being embedded in an EAR, ADF applications will generally deploy the EAR or MAR files.

Hands-On Practice: Deploy a Java EE Web Application

This hands-on practice provides the steps for deploying an existing Java EE application written with Oracle ADF technologies. The practice steps refer to an application —the second version of The Ultimate Human Resources Application (TUHRA or TUHRA2)—developed for the authors' book *Oracle JDeveloper 11*g *Handbook*. Although the practice refers to specific features and files in this application, you can use an application of your own and adjust appropriately.

Should you want to follow the steps exactly as written, you can start with the same sample application referred to in this practice. Start by pointing your web browser to tuhra2.java.net. Then click Downloads and download the file for Chapter 21, part 4. Use this sample application as a starting point for this hands-on practice.

> **Note**
> This practice was developed with JDeveloper, version 11.1.1.1.0 (build 5407). If you use a later release, you may need to adjust steps to changes introduced with that later release. However, the principles and steps are the same.

For the purposes of this hands-on practice you will use the existing standalone WebLogic software that is part of the JDeveloper installation. Although the process followed in this hands-on practice uses an existing WebLogic does not employ a full application server, it very closely emulates the process used for deployment to a full application server environment. As mentioned, although in many organizations it is not the responsibility of the developer to finally deploy applications into production, testing the deployment is nevertheless an essential step in the testing of the application before you hand it over to those who will perform the deployment.

This practice follows these phases:

I. Set up a standalone WebLogic server for testing

- Create and run a WebLogic domain
- Run the Oracle WebLogic Server Administration Console

II. Configure application-specific settings

- Define database connection information
- Set up users and groups

III. Deploy the application

- Deploy the application from JDeveloper
- Deploy and install the application from an EAR file

I. Set up a Standalone WebLogic Server for Testing

Within your JDeveloper 11*g* installation, you already have all of the pieces required to set up a standalone WebLogic server for testing. When you run the TUHRA application from within JDeveloper, the IDE takes various shortcuts to speed up the runtime and so does not exactly emulate the final deployment environment with the application server. Therefore, this

additional step to deploy to a standalone server is a valuable one because, if you've tested an EAR file by deploying it to a standalone server, the hand-off process for the final production environment will be smoother.

> **Note**
>
> You can read more information about the WebLogic Server in the online manual "Oracle Fusion Middleware Deploying Applications to Oracle WebLogic Server 11g Release 1 (10.3.1)," currently available at download.oracle.com/docs/cd/E12839_01/web.1111/e13702/toc.htm.

> **Tip**
>
> Links to specific on-line documents may change from time to time, however, you can always start from otn.oracle.com and click **Documentation | Fusion Middleware** on the shortcuts menu bar to get to the very latest documentation on WebLogic.

Create and Run a WebLogic Domain

When you set up a server environment in WebLogic, you need to create a *WebLogic domain*, one or more servers, all configured with a certain set of shared libraries. As shown in Figure 1, on the left, a WebLogic Server (WLS) software installation on a single physical server machine may support multiple WLS domains configured for ADF, or, on the right, a single ADF WLS domain may be distributed across many physical servers, each having a WLS installation.

As a further twist, the architecture supports two kinds of servers: *administration servers*, used to control and monitor other servers but not intended to run applications, and *managed servers*, used to run applications. In a production domain there would generally be one administration server and many managed servers, to which applications would be deployed. For local testing purposes, it's fine to just create a single administration (admin) server and use that server for deployment.

The instructions in this section are based on the use of the command line rather than the Windows Start menu, although instead you can use the Windows Start menu options (in the **Start** | **Programs** | **Oracle Fusion Middleware 11.1.1.x.x** | **WebLogic Server 11gR1** | **Tools** program group). In all cases the scripts mentioned will have two versions, a ".cmd" version for Windows and a ".sh" version for Unix/Linux. Use the version that applies to your platform and the appropriate path separator in your commands (";" for Windows and ":" for Unix/Linux).

1. If JDeveloper is open, be sure to stop all server instances (default and application) using the Terminate (red box) button in the toolbar. (Stopping the default server will automatically stop any running application instances.)
2. Open a command shell (command-line) window and change (CD) to the directory into which you installed Oracle Fusion Middleware. In these examples, we use "FMW_HOME" to represent the directory into which you installed JDeveloper (for example, C:\Oracle\Middleware).

   Additional Information: In Windows, you can open a command-line window by selecting **Start | Run,** entering "CMD," and clicking OK.
3. Change to the <FMW_HOME>\jdeveloper\common\bin directory and run the config.cmd script. The Oracle WebLogic Configuration Wizard will start. (You can alternatively run this utility using the Configuration Wizard selection in the Start menu program group mentioned earlier.) The dialog in Figure 2 will appear.
4. If no existing standalone domain exists, leave the *Create a new WebLogic domain* option selected and click Next.
5. The Select Domain Source page will appear with a list of possible products (libraries) to configure into the domain. Select the *Oracle JRF* checkbox and click Next.

   Additional Information: The Oracle JRF (Java Runtime Foundation) contains all of the shared JAR files used by ADF BC and ADFm; it also includes the ADF Faces components.
6. On the Specify Domain Name and Location page, set *Domain name* as "adf_test" and leave the *Domain location* as the default (for example, "<FMW_HOME>\user_projects\domains"). Click Next.

7. The Configure Administrator User and Password page will appear where you define the account used to manage the server. By convention the default user name used for the administrator is "weblogic," and that default will work. Set *User password* and *Confirm user password* to "weblogic1" (the password must contain a number). Click Next.
8. The Configure Server Start Mode and JDK page will appear. Ensure that the *WebLogic Domain Startup Mode* is set to "Development Mode," and accept the default for *JDK Selection*, which should be the same JDK version with which you are running JDeveloper (for example, "1.6.0_11"). Click Next.
9. The Select Optional Configuration page will appear. You will not need any of these options, so just click Next.
10. The Configuration Summary page will appear as in Figure 3. Review the settings. You can return to earlier pages using the Previous buttons.
11. Once you have reviewed and confirmed the settings, click Create. The Creating Domain progress page will appear.
12. When the creation process completes, select the *Start Admin Server* checkbox in the lower-right corner of the dialog and click Done. The Configuration Wizard will exit, and another command-line window will open and display messages about the server startup. Wait for the message that ends with "<Server started in RUNNING mode>."

> **Caution**
>
> For now, leave the server command-line window. Closing this window will stop the server.

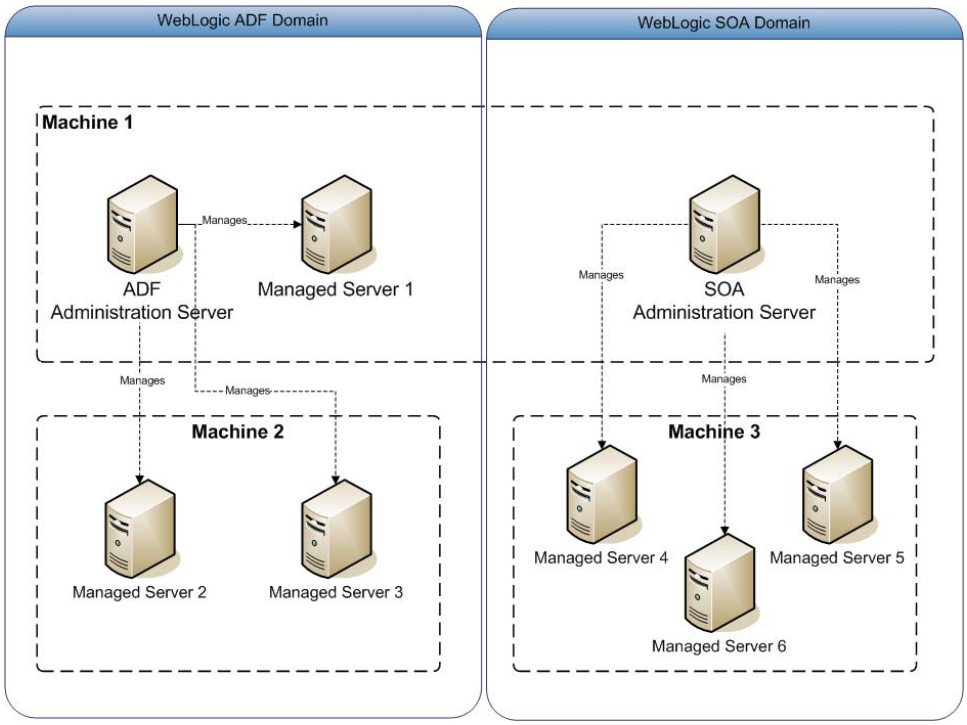


Figure 1. Domains in a WebLogic server

13. The original command-line window from which you started the Configuration Wizard will return to a command-line prompt. Close this window (but not the server window).

> **Tip**
>
> If you accidentally close the server window, or forgot to start the server at the end of the configuration wizard, jump ahead to the sidebar "Starting the WebLogic Domain Server from the Command Line" for instructions on starting the server.

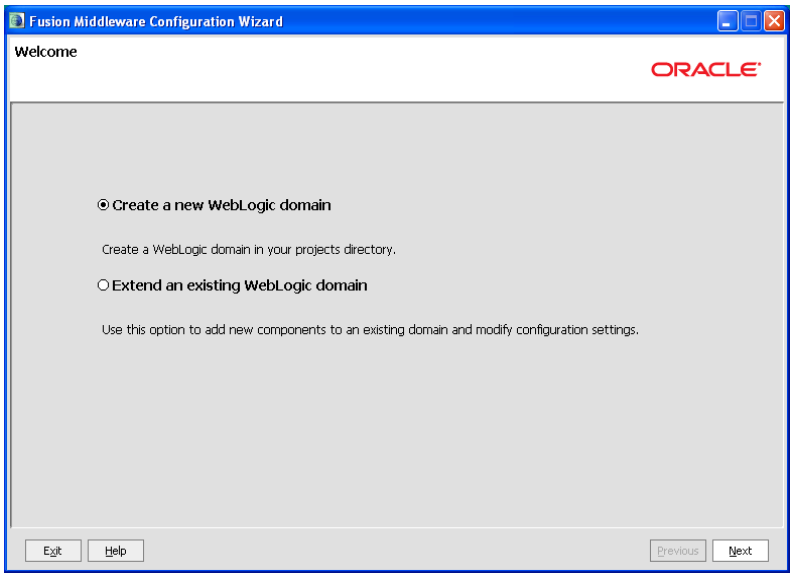


Figure 2. Welcome page of the Fusion Middleware Configuration Wizard

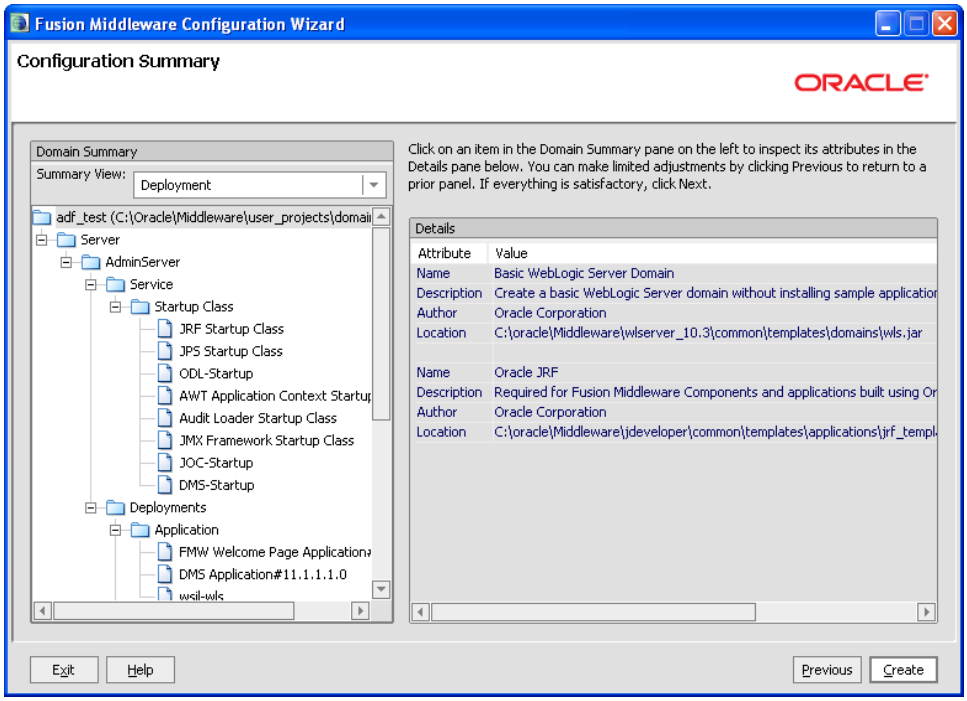


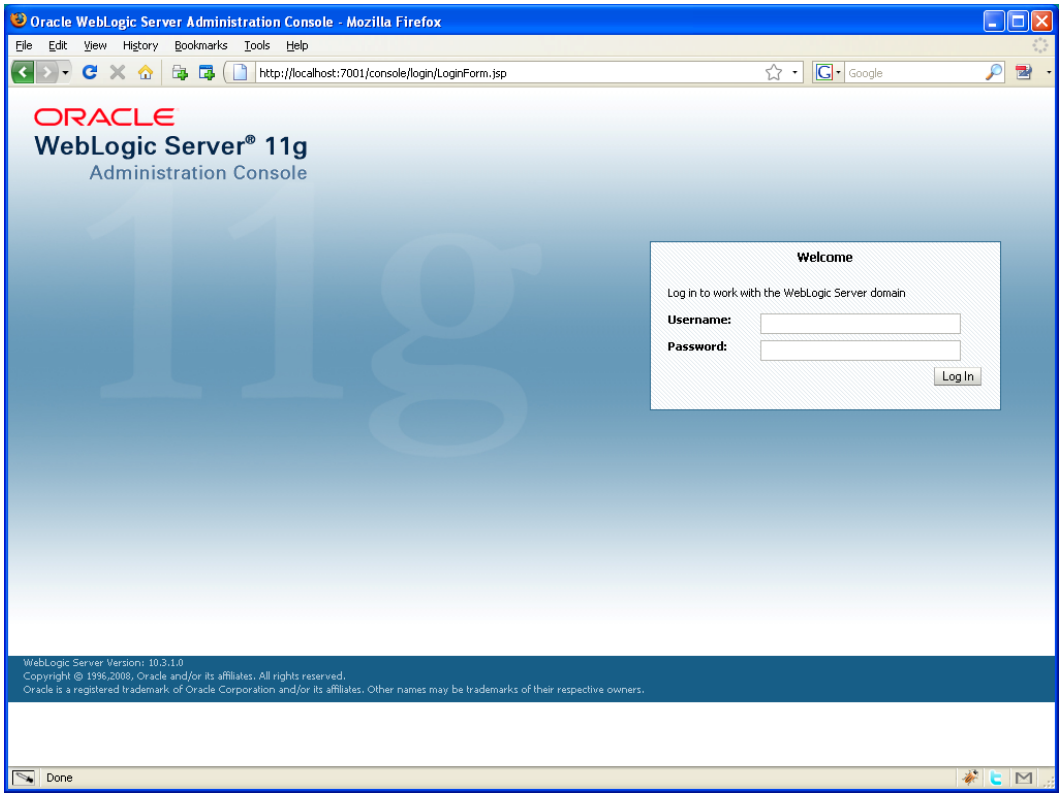
Figure 3. Configuration Summary page of the configuration wizard

Run the Oracle WebLogic Server Administration Console

Next you need to configure some key attributes to allow the TUHRA2 application to successfully deploy and run—specifically, connection information for the database and a list of valid users for the security was added to the application . (Refer to the authors' white paper "WebLogic Server Application Security— Implementing the Superstition in JDeveloper" for more information about application security.) This configuration is performed using the Oracle WebLogic Server Application Console ("the console").

> **Tip**
>
> A single WebLogic admin server can be used for multiple concurrently running server instances. If you want to set up multiple servers, you would fill in some of the Configuration Wizard screens to assign a unique port number to each server on a physical machine. The default port is 7001.

1. Open a web browser and navigate to the URL: http://localhost:7001/console. This will launch the WebLogic console application, which is installed into every administration server. You will see a message saying "Deploying application for /console," then the following login screen will display.



2. Log in to the console using the user name and password that you specified earlier (weblogic/weblogic1). The administration console page will then display as shown in Figure 4. Notice the domain name in the top right-hand corner ("Connected to") area.

WHAT DID YOU JUST DO?

The configuration wizard that you ran in this phase creates a set of configuration files and scripts, which will start a WebLogic Server instance. This server is configured to use the default port: 7001, which is different from the port that is used when running within JDeveloper (port 7101), so you can safely use both the internal server and this new external testing server at the same time. The domain that you have created is set up with all of the libraries required to run an ADF application.

In this phase, you also ran the console, the tool you will use for the rest of the configuration tasks. Before you run the console, you need to start the server. The steps in this phase ran the server from the Configuration Wizard. The next time you need to start the server, you will be using the command line batch file or shell script (or shortcut) as described in the sidebar "Starting the WebLogic Domain Server from the Command Line."

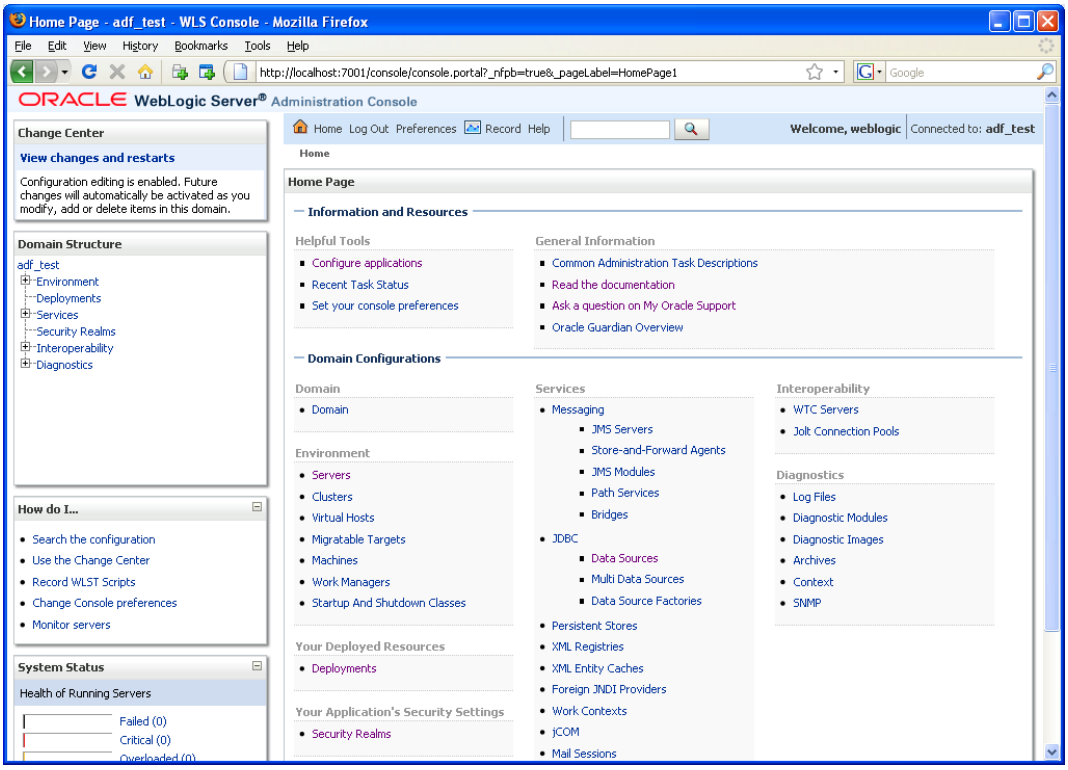


Figure 4. Oracle WebLogic Server Administration Console for the adf_test domain

What Could You Do Next?

You can also access the administration console for the default server in JDeveloper. Once JDeveloper starts the default server (after you run an application, for example), use the default server URL (http://localhost:7101/console) to start the console. The user name and default password are the same as those used in this practice (weblogic / weblogic1).

Starting the WebLogic Domain Server from the Command Line

The last step of the Oracle WebLogic Configuration Wizard allowed you to start the server (domain). If you are not running the wizard, you can manually start up the server using the following steps. If the server is running and you want to try these steps, press CTRL-C, then enter "Y" at the prompt to stop the server, and then close the command-line window.

1. From a new command-line window, navigate to the FMW_HOME directory and then navigate to the user_projects\domains\adf_test. Notice that "adf_test" corresponds to the name and location of the domain that you created earlier.
2. At the command line, run the script startWebLogic.cmd (or .sh) script to start the server instance. As before, wait for the "Server started in RUNNING mode." message. Leave this window open. Reconnect to the console as described earlier.
3. To stop the server you can run the corresponding stopWeblogic.cmd (or .sh) script in the adf_test\bin directory rather than pressing ctrl-c in the server window.

Tip

As with any frequently used program, you will probably want to create a shortcut on your desktop for startWebLogic.cmd (Windows) or startWebLogic.sh (Unix or Linux).

II. Configure Application-Specific Settings

The steps in the preceding phase centered around tasks you need to complete regardless of the application. This phase focuses on tasks that are more specific to an application, for example, database connection information and users and groups.

Define Database Connection Information

Running the TUHRA2 application from within JDeveloper requires a database connection called HR_DB. The sample application altered its ADF BC configuration "TUHRAServiceLocal" to make the application use the HR_DB connection information in a JDBC data source rather than as a hard-coded user name and password. This *JDBC data source* mechanism allows you to abstract the connection information and hold it separately from the application definition. *Data sources* are a named connection resource that the application requests from the application server for rather than holding the connection information directly. This feature allows the application to remain unchanged as it is moved from server to server, for example, test to production. In this section you will define this data source in the adf_test WebLogic domain.

1. Return to the WLS console (http://localhost:7001/console). In the Domain Structure (left-hand side) region expand the Services\JDBC node and click the Data Sources node. The screen will change to look like Figure 5.

   Additional Information: The Multi Data Sources node in the services section is used when configuring a data source to work with an Oracle Real Application Clusters (RAC) database. The "Data Source Factories" page is used only for backward compatibility.

2. Click New (above the Data Sources table) to launch the Create a New JDBC Data Sources wizard, which you will use to create a data source. On the first page of the wizard, set the values as follows:

Field	Value	Purpose
Name	TuhraDS	A name that identifies this resource in the console.
JNDI Name	jdbc/HR_DBDS	The name that is used to look up the connection. This value matches the value defined in the ADF BC configuration without the java:comp/env prefix.
Database Type	Oracle	This is the default; it assumes you're accessing an Oracle database.
Database Driver	Oracle's Driver (Thin) for Service Connections	The correct driver for a local XE or local Oracle 11*g* database installation. Do not select the "Thin XA" driver.

3. Click Next. On the next page leave all options as their defaults and click Next.
4. Fill in the fields on this page as in the following table to define the core connection information. The following values are based on a default Oracle XE database installed on the local machine; adapt these if you are using a different database type.

Field	Value	Purpose
Database Name	XE	The SID or service name of the database ("XE" for Oracle XE and "ORCL" for an 11*g* install, by default).
Host Name	localhost	This setting will loop back to your local machine; alternatively use the database machine's host name, if any.
Port	1521	The default port for an Oracle database.
Database User Name	hr	The schema owner (HR).
Password	hr	Use the password for hr on your system.
Confirm Password	hr	The same as the password.

5. Click Next. The screen will change slightly adding fields at the bottom, one of which, *Test Table Name*, allows you to enter a sample SQL statement (one is entered by default) for testing purposes.

6. Be sure your database is started and click the Test Configuration button to check that the database is accessible. A green "Connection test succeeded" message will appear near the top of the screen if the connection is correctly defined and the database is reachable.

   Additional Information: If the connection fails, cross-check the settings on all pages with the settings in the preceding steps and with the HR_DB database connection information in JDeveloper.

7. Click Next to display the Select Targets page. At this point, you need to activate the data source for the administration server. Select the *AdminServer* checkbox.

   Additional Information: If you miss this step or need to deactivate the data source later, you can click the data source link after displaying the list of data sources, select the Targets tab, select (or deselect) the target server, and click Save.

8. Click Finish. The wizard will set up the data source and return you to the Data Sources list, which now contains TuhraDS.

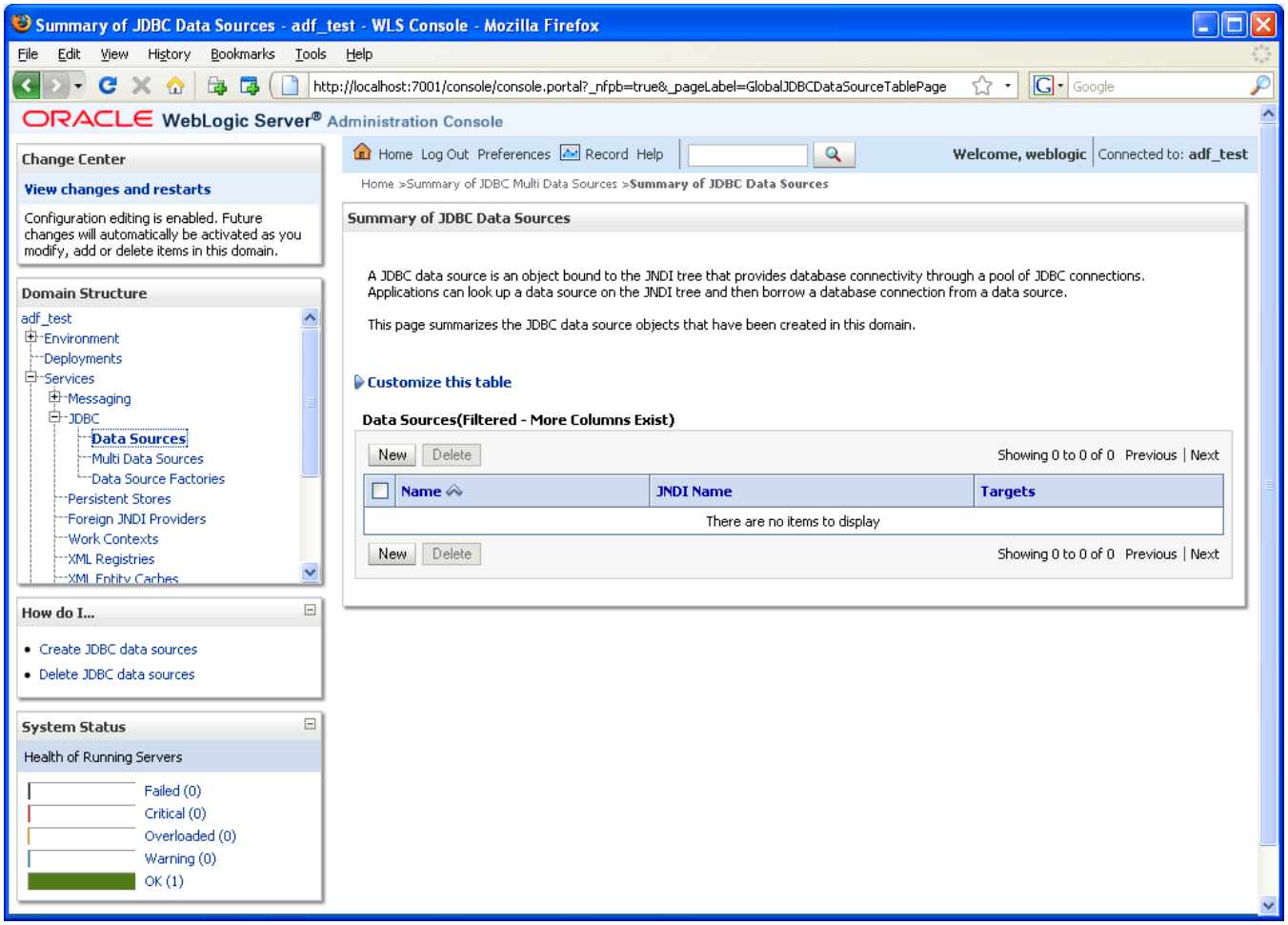


Figure 5. Summary of JDBC Data Sources page of the console

Set Up Users and Groups

The next configuration task is to configure the WebLogic server's LDAP service for the users and roles that TUHRA2 needs. When you run the application from within the IDE, JDeveloper takes care of this for you, but running on a standalone server requires some setup steps.

1. In the WebLogic console's Domain Structure navigator area, click *Security Realms* to display a single predefined realm called "myrealm." Click myrealm.

> **Note**
>
> Remember that a realm in security is just a namespace. It does not matter that the realm you are using here is different from the default (jazn.com).

2. Click the Users and Groups tab in the "Settings for myrealm" page and click New to define each of the application users (CDAVIES, NKOCHHAR, and TFOX, all with passwords of "welcome2tuhra"). The screen should appear as in the console snippet in Figure 6.

   Additional Information: Feel free to add descriptions (as shown in Figure 6) and create extra users as well, as long as you assign them to the correct groups, as you will do for the main users in the next steps.

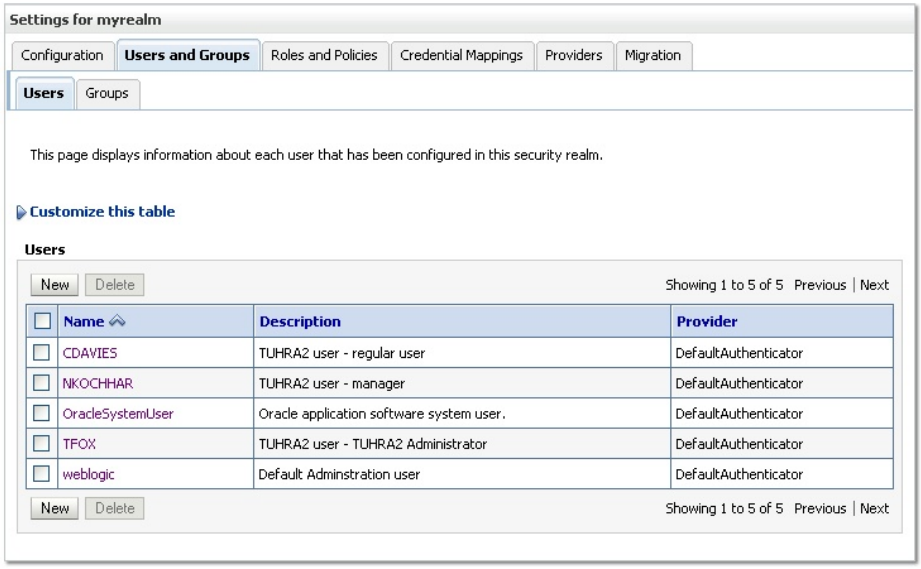


Figure 6. Users and Groups tab of the Security Realm page

3. Click the Groups subtab.
4. Click New on the Groups table to add a group called "cs_admin" that will represent the application's admin group. Provide a suitable description. Click OK.

   Additional Information: Naming the server group differently from the application's logical roles helps illustrate the kind of mapping that will often have to take place when using existing credential stores such as Oracle Internet Directory (OID). Each of the server's LDAP credential store groups will correspond to one of the local server groups you set up when testing security from within JDeveloper.
5. Repeat the preceding step to define groups for cs_manager and cs_user. Provide a suitable description for each group.
6. Click the Users subtab.
7. Click CDAVIES to display the "Settings for CDAVIES" page. Click the Groups subtab. Shuttle "cs_user" from Available to Chosen. The screen will contain the areas shown in Figure 7.
8. Click Save. Click the "Users and Groups" link in the breadcrumb area (above the title "Settings for myrealm").
9. Repeat the preceding two steps to assign NKOCHHAR to the cs_manager group and TFOX to the cs_admin group.
10. Click Home (under the WLS Server banner at the top of the page) to display the main console page.

What Did You Just Do?

In this phase you have carried out some essential configuration work. First, you ran the Oracle WebLogic Server Application Console, where you can perform configuration and administrative tasks for the server instance. You defined a JDBC data source for the application to use for its database connections. Then you created a set of groups and test users in the internal LDAP server provided by Oracle WebLogic so that security in the deployed application can be tested. You also assigned users to groups.

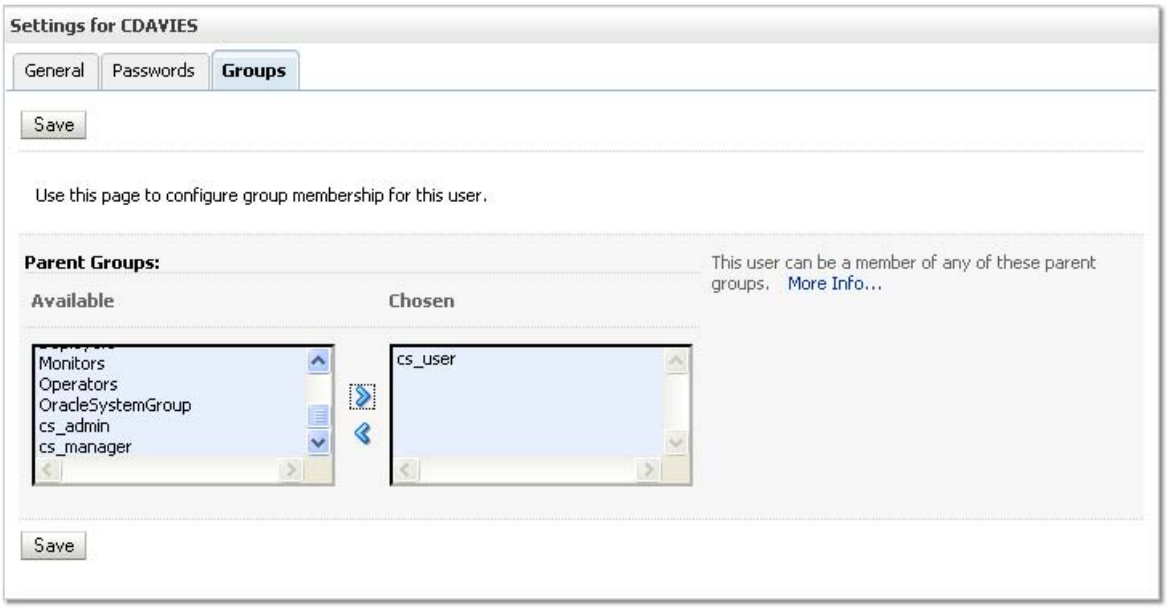


Figure 7. Assigning groups to a user

III. Deploy the Application

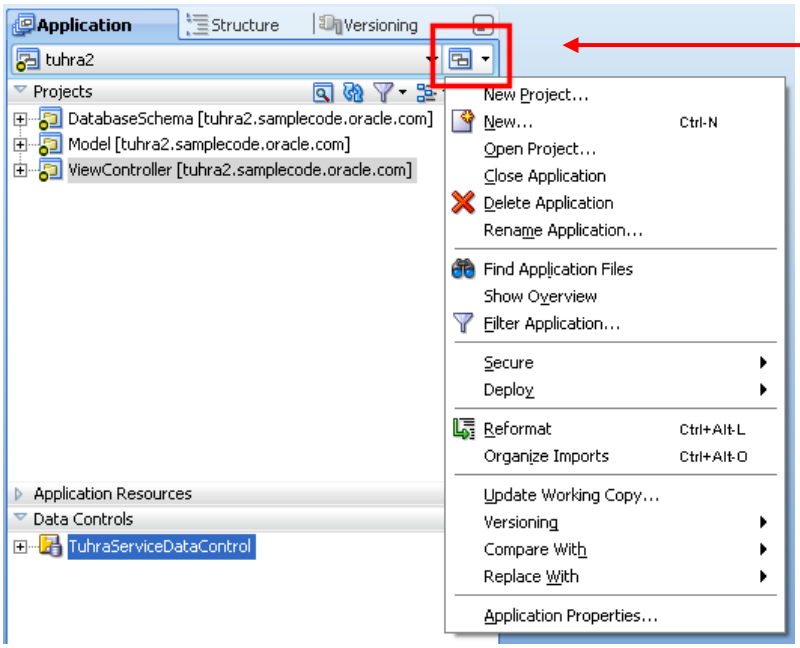
Now that the server is set up, you can deploy the TUHRA2 application to it. This section will actually explain two alternative forms of deployment: deployment directly from JDeveloper, which you would use to test the deployment on a local WebLogic Server instance, and deployment using an EAR file, which is the technique that you or an administrator would usually use to roll out an application into a full application server environment.

> **Note**
>
> If you have been using an earlier version of ADF security, you may be familiar with the concept of having to manually migrate security permissions from the application jazn-data.xml file into the application server. One of the big improvements you will see in this version of the framework is that this permission deployment is now handled by the direct deployment from JDeveloper.

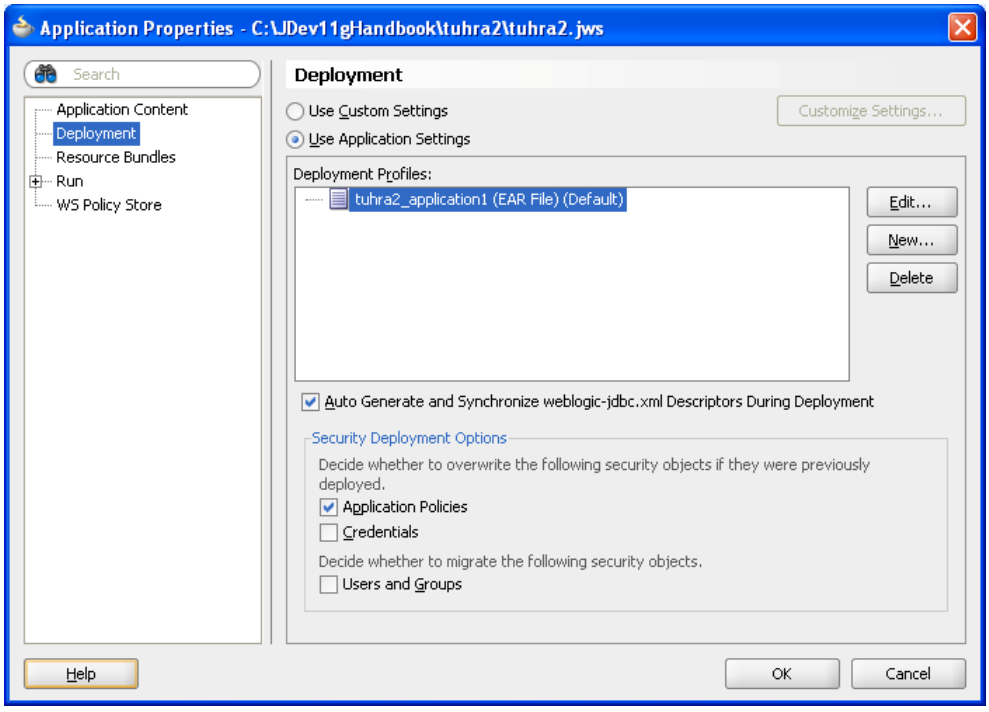
Deploy the Application from JDeveloper

JDeveloper makes the process of deploying to a local or a remote WebLogic server, a relatively easy, point-and-click task.

1. In JDeveloper, with the TUHRA2 application open in the Application Navigator, display the project properties for the ViewController project (by double clicking the ViewController node). On the Java EE Application page, ensure that *Java EE Web Application Name* and *Java EE Web Context Root* are both set to "tuhra2."
2. Click OK.
3. Click the application menu for TUHRA2 shown in the following illustration:



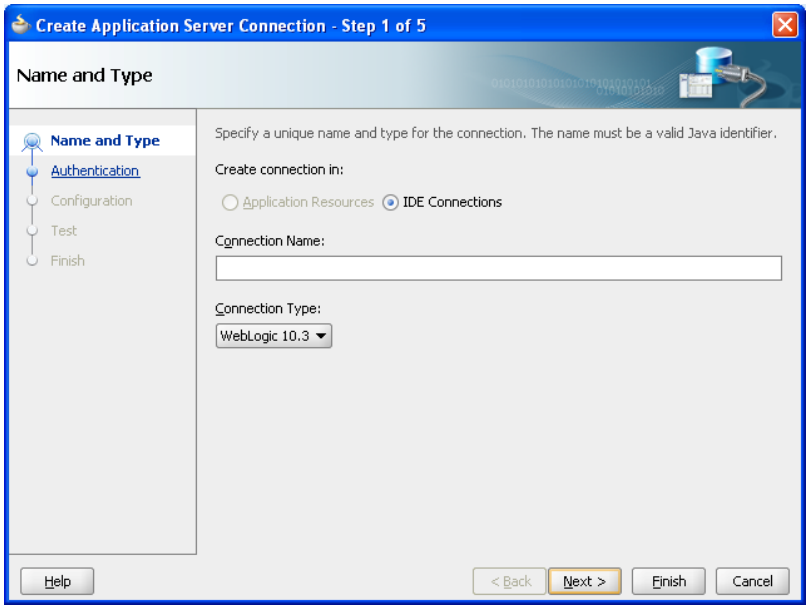
4. Select Application Properties (also available from **Application** | **Application Properties**), and in the Deployment page ensure that the *Application Policies* checkbox is selected and the *Auto Generate and Synchronize weblogic-jdbc.xml Descriptors During Deployment*, *Credentials*, and *Users* and *Groups* checkbox are unselected as shown here:



Additional Information: You have already explicitly defined the database connection, credentials, users, and groups on the server side and do not need to move those items.

> **Caution**
>
> If you fail to deselect the *Auto Generate and Synchronize weblogic-jdbc.xml Descriptors During Deployment* option the final deployment from an EAR file in Phase III will fail. This is because this option embeds a separate data source definition for the application with the EAR file but cannot embed the password for it (the password has to be encrypted after installation onto the target WebLogic server). It is possible to complete this data source definition from the WebLogic console after installation; however, because you have already defined a global data source it is much simpler to avoid the whole problem by not generating the extra application level data source. Note that you will need to re-select the Auto Generate and Synchronize weblogic-jdbc.xml Descriptors During Deployment option if you want to run the application again in the embedded server.

5. Click OK to dismiss the dialog.
6. Be sure the WebLogic server is running from the preceding phase. If it is not, return to the sidebar "Starting the WebLogic Domain Server from the Command Line" in Phase I and start the server.
7. From the Application menu, select **Deploy** | **tuhra2_application1** | **to** | **New Connection**. The Create Application Server Connection dialog will appear as shown next:



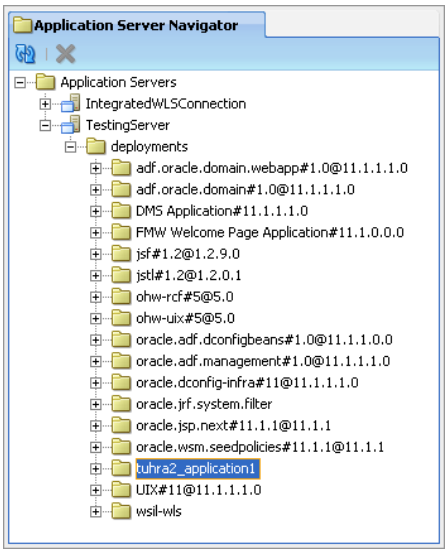
8. Set the *Connection Name* as "TestingServer." Leave the *Connection Type* as "WebLogic 10.3." Click Next.
9. On the Authentication page, provide the user name and password, which you used to log in to the WebLogic console for the adf_test domain (weblogic/weblogic1). Click Next.
10. On the Configuration page, leave all of the defaults except for setting the WLS Domain value to the name of your domain, "adf_test." Click Next.
11. Click Test Connection and wait for the test to complete. All of the tests should succeed. Click Finish.

    Additional Information: In the future, you will be able to select this connection from the Deploy menu option without having to step through this connection wizard.
12. The deployment will proceed, and you will see messages appear in the Log window.

Additional Information: If more than one server is defined for the domain, an additional popup dialog allows you to select the target server.

13. When the process is complete, the Log window will display "Deployment finished."
14. Open the Application Server Navigator (**View | Application Server Navigator**). Expand the Application Servers\TestingServer\ deployments node to view all deployed applications as shown on the right with the TUHRA2 deployment highlighted.

    Additional Information: The right-click menu on a deployed application node allows you to undeploy the application.



15. Test the deployed application by opening a browser window and entering the URL http://localhost:7001/tuhra2/faces/employeeSearch into the browser location field.
    Additional Information: The Employee Search page will appear. Alternatively, you could test the login.html page to confirm that the login works in this server arrangement.
16. Test various application functions and close the browser when you are finished testing.

EXPLORING THE EAR FILE

As mentioned earlier in this white paper, an EAR file is the standard deployment unit for Java EE applications; it contains all the bits of the application that have to be deployed except for the connection information and security setup, which are held in the server.

The EAR file created by this process was created, copied to the server, and deployed automatically by the JDeveloper deployment process. However, you can open and examine the contents of this file using the following steps:

1. In JDeveloper, look in the Deployment log window for a message about the EAR file that was created ("Wrote Enterprise Application Module to…"). Notice the file system location for the EAR file. Select **File** | **Open** and navigate to that deploy directory under TUHRA_HOME.

   Additional Information: The deploy directory is parallel to the Model and ViewController directories. You will see two nodes for the one TUHRA2_application1.ear file. One node (displayed as a file icon) allows you to open the EAR as a file; the other node (displayed as a directory icon with a "?") allows you to look inside the EAR file and select one or more files.

> **Tip**
>
> You can copy the file system location from the Log window and paste it into the Open dialog instead of navigating through the file system.

2. Select the version of the EAR file that displays as a file (page with lines) icon and click Open. The list of EAR file contents should open in the Archive Viewer. The list of files in the EAR file is mostly configuration files and the WAR file, tuhra2_ViewController_webapp1.war.
3. Double click the WAR file to examine its contents. You should see familiar files as you browse this list, for example, Model project files that are required for the application, library JAR files, and ViewController files (JSF files and images).
4. You can open any of these files in an editor or viewer window by double clicking the file name. Close the Archive Viewer when you are finished browsing.

Deploy and Install the Application from an EAR file

Although the approach of deploying the application directly from JDeveloper will work successfully for both local and remote application servers, many organizations would prefer that their developers not directly update production servers. In these cases the developer will need to create an EAR file of the application to hand over to the administrator for deployment.

Most Java EE application servers (with exceptions such as Apache Tomcat, which is not a complete Java Enterprise Edition application server) understand how to automatically install an EAR file. The process of deployment directly from JDeveloper

that we explained in the previous section is really just automating the process of packaging the EAR file, copying it to the target server machine, and triggering the installation. However, since most administrators will use manual deployment steps, we will run through that technique in this section. Although you've already created the EAR file as part of the JDeveloper deployment steps before, the following creates it again to demonstrate the steps. If you are in charge of deployment to the production application server, you can use these steps to practice the production deployment steps on a standalone local server.

1. Starting from the JDeveloper Application Navigator open the application menu, and select **Deploy** | **tuhra2_Application1** | **to EAR File**. By default this will package the application into an EAR and copy it to the deploy subdirectory underneath the application directory root. You will see "Deployment finished" in the Log window when the process completes.

   Additional Information: You would then pass this EAR file to the application server administrator, who would deploy the application to the server, but the following steps demonstrate the process of deployment using the WebLogic console in your standalone WebLogic server.

2. Next, log in to the WebLogic console (http://localhost:7001/console) using the user name/password you set up (weblogic/weblogic1). From the Domain Structure tree on the left-hand side of the screen, click Deployments.
3. Browse the list of the deployed applications until you find tuhra2_application1 that you deployed from JDeveloper as shown here:

4. Select the checkbox next to this deployment and click Delete to remove it.

> **Note**
>
> This step just demonstrates how to remove a deployment, but it is not really necessary. You could choose instead to update the deployment.

5. Click the Install button in the Deployments table. This will launch the Install Application Assistant, the first screen of which will allow you to navigate to and select the EAR file for the application (tuhra2_application1.ear) from your local disk.
6. Find the file (as mentioned before, the name is provided at the end of the Log window messages in a line starting with "Wrote Enterprise Application Module to"). Select the radio button next to the file name and click Next.
7. On this page, leave the default option of "Install this deployment as an application" selected and click Next.
8. On this page, leave all of the options as default and click Finish. The "Summary of Deployments" page will display.
9. Scroll to the page containing tuhra2_application1 to confirm that the State is "Active" and Health is "OK."
10. Open a new browser session and connect to the application as before (http://localhost:7001/tuhra2/faces/employeeSearch). Test the application.

11. Close the browser and stop the server using the stopWeblogic script or by pressing ctrl-c and closing the server's command-line window.

What Did You Just Do?

In the final phase of this practice, we showed how to deploy the application you developed into a standalone instance of WebLogic Server. In the process you used both automated deployment from JDeveloper as well as the more manual process using the console to deploy an EAR file. As mentioned, you would only need to deploy an application once, but this phase showed both methods to give you experience with the processes.

When deploying directly from JDeveloper, you also migrated the permissions used by ADF to secure bindings and task flows. The manual migration of these permissions is also discussed in some detail in the JDeveloper online help if you need to understand the steps in more detail.

Summary

This white paper has introduced the principles of deploying a Java EE web application to Oracle WebLogic Server. It described how application files are assembled into the Java archive files used for deployment—WAR and EAR (and occasionally, MAR). It also described the main second stage of deployment—copying the archive file to the server—and how the server expands the file into its components files and directory structures. The white paper then concluded with a demonstration of these principles in the form of a hands-on practice that you can follow using JDeveloper and WLS.

> *This is not the end.*
> *It is not even the beginning of the end.*
> *But it is, perhaps,*
> *the end of the beginning.*
>
> —Winston Churchill (1874–*1965*),
> Nov. 10, 1942, Mansion House

About the Authors

Peter Koletzke is a technical director and principal instructor for the Enterprise e-Commerce Solutions practice at Quovera, in Mountain View, California, and has 29 years of industry experience. Peter has presented at various Oracle users group conferences more than 300 times and has won awards such as Pinnacle Publishing's Technical Achievement, Oracle Development Tools Users Group (ODTUG) Editor's Choice (three times), ODTUG Best Speaker, ODTUG Volunteer of the Year, NYOUG Editor's Choice (three times), and ECO/SEOUC Oracle Designer Award. He is an Oracle Certified Master, Oracle ACE Director, and coauthor (variously with Dr. Paul Dorsey, Avrom Roy-Faderman, and Duncan Mills) of eight Oracle Press development tools books including *Oracle JDeveloper 11g Handbook* (on which some of the material in this white paper is based).

Duncan Mills is senior architect for Oracle's ADF Product Development group. He has been working with Oracle in a variety of application development and DBA roles since 1988. For the past 18 years he has been working at Oracle in both customer support and product development, spending the last 10 years in product management for the development tools platform. Duncan is a frequent presenter at industry events and has many publications to his credit including coauthorship of the Oracle Press books: *Oracle JDeveloper 11*g *Handbook* and *Oracle JDeveloper 10*g *for Forms and PL/SQL Developers*.