ORACLE
UNIVERSITY

Oracle WebLogic Server 12*c*: Administration II

Activity Guide

D80153GC20

Edition 2.0 | December 2016 | D96008

Learn more from Oracle University at **oracle.com/education/**

ORACLE

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS
The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Authors

Mark Lindros, TJ Palazzolo, Al Saganich, Tom Eliason

Technical Contributors and Reviewers

Bill Bell, Elio Bonazzi, Tom McGinn, Eduardo Moranchel Rosales, Will Lyons, David Cabelus, Greg Stachnick, Donna Micozzi, Jon Patt, Matthew Slingsby, Bill Albert, Rich Whalen, Kevin Tate, Serge Moiseev, Takyiu Liu, Angelika Krupp, Viktor Tchemodanov, Diganta Choudhury, Jose Alvarez, Alexander Ryndin

This book was published using: Oracle Tutor

Table of Contents

Course Practice Environment: Security Credentials

Chapter I

Course Practice Environment: Security Credentials

For OS usernames and passwords, see the following:

- If you are attending a classroom-based or live virtual class, ask your instructor or LVC producer for OS credential information.
- If you are using a self-study format, refer to the communication that you received from Oracle University for this course.

For product-specific credentials used in this course, see the following table:

Product-Specific Credentials		
Product/Application	**Username**	**Password**
Oracle WebLogic Server	weblogic	Welcome1
Oracle Database	oracle	Welcome1
Oracle HTTP Server	weblogic	Welcome1

Practices for Lesson 1: Course Introduction

Chapter 1

Practices for Lesson 1

Practices Overview

There are no practices for Lesson 1.

Practices for Lesson 2: WebLogic Server Review

Chapter 2

Practices for Lesson 2

Practices Overview

There are no practices for Lesson 2.

Practices for Lesson 3: Upgrading WebLogic Server

Chapter 3

Practices for Lesson 3: Overview

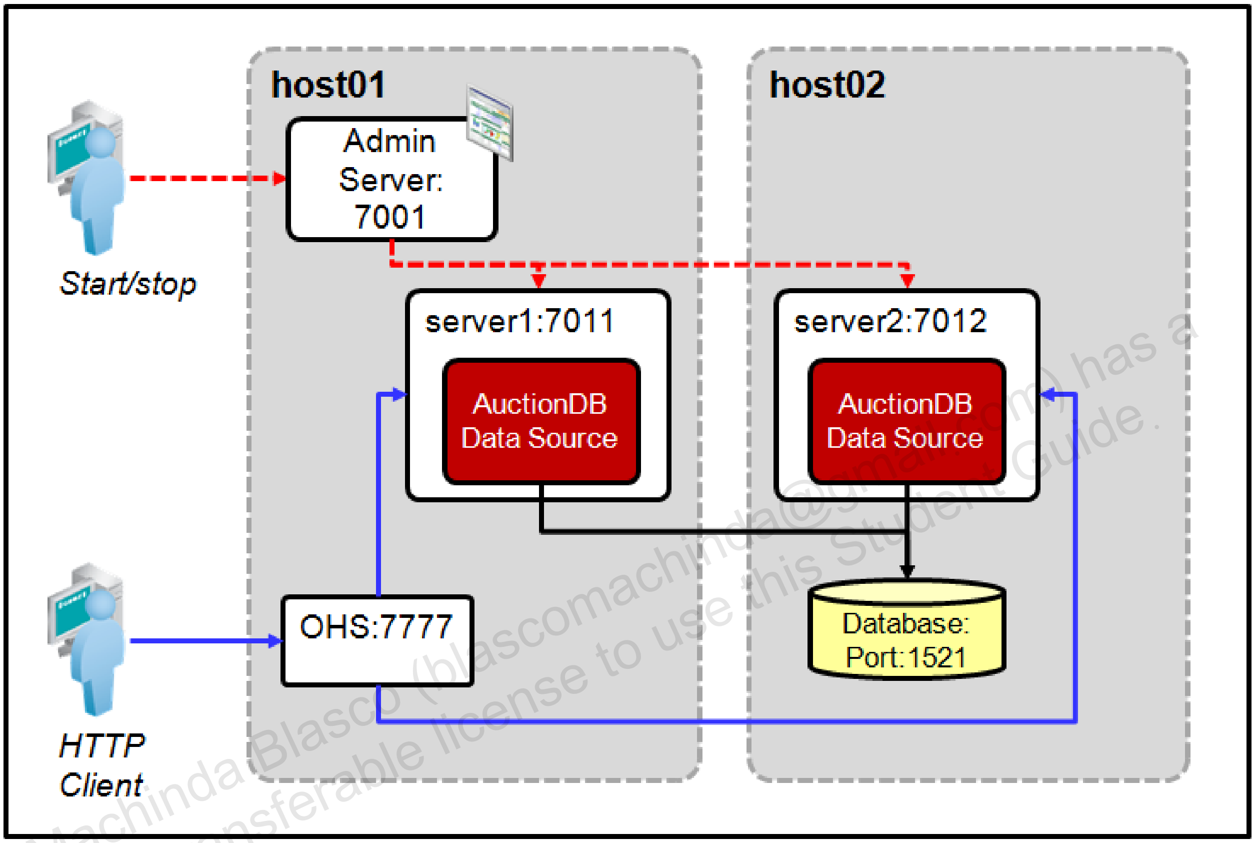
Practices Overview

In the practices for this lesson, you set up the general lab environment for this course and learn how to perform a rolling upgrade that enables you to patch WebLogic Server without affecting the clients using the application.

Practice 3-1: Setting Up the Course Practice Environment

Overview

This practice shows you how to set up the initial course environment used for all the practices of the course. You also verify that the WebLogic domain used for the course is functioning properly. The following image depicts the architecture of the domain used for this practice.



Dependencies

All practices depend on the completion of this practice.

Tasks

1. Connect to the **host01** and **host02** machines.

   Note: Refer to the instructions in “Appendix A: Connecting to the Environment” to connect to your machines.

2. Set up the initial course environment.

   Note: This step is required for all practices and solutions in this course to function properly.

   a. Open a terminal window **on each machine** by clicking the terminal icon in the launch panel located at the top of the screen.

   b. Set the title of each terminal window by selecting Terminal > Set Title and entering `MAIN` as the title. This makes it easier to distinguish the purpose of each window.

   c. Run the following script **on the host01 and host02** machines to set the environment for this course. You will not have to run this script again.

   ```
   $ . /practices/part2/bin/setenv.sh
   ```

   Note: This script writes this same command to your `/home/oracle/.bashrc` file, so the environment is automatically set for you from then on.

3. Set up the practice environment.

   Note: When running any of the scripts in this course that start WebLogic servers, you can generally ignore the following errors displayed in the server console windows:

Message	Description
java.rmi.RemoteException: [Deployer:149145]Unable to contact "serverX". Deployment is deferred until "serverX" becomes available.	This is intentional because the scripts start the administration server first. Then before starting the managed servers, it performs WLST and deployment tasks that are automatically realized when the managed servers start. This eliminates the need to restart servers for changes that may not be dynamic.
<Sep 11, 2013 6:37:23 PM UTC> <Error> <Cluster> <BEA-000109> <An error occurred while sending multicast message: java.io.NotSerializableException: com.sun.jersey.server.impl.cdi.CDIExtension	This is a harmless exception that occurs as part of the Auction application. It will not affect the functionality of the application or the course.

   a. In the MAIN terminal window, navigate to the domain folder **on host01 and host02**:

   ```
   $ cd /u01/domains/part2/wlsadmin
   ```

   b. Run the following command **on host01** to start the domain’s AdminServer. The `startAdmin.sh` script is located in this course's `bin` folder, which is in your `PATH`, so it can be executed from any folder location on host01.

   ```
   $ startAdmin.sh
   ```

   Note: The scripts used in this course automatically set the terminal window title to AdminServer, server1, and server2 for each WebLogic server instance that runs within that window.

   c. Wait for the administration server to start.

d. Run the following command **on host01** to start the domain's server1 managed server. The `startServer1.sh` script is located in this course's `bin` folder, which is in your `PATH` so it can be executed from any folder location on host01.

```
$ startServer1.sh
```

e. Run the following command **on host01** to start the domain's server2 managed server. The `startServer2.sh` script is located in this course's `bin` folder, which is in your `PATH` so it can be executed from any folder location on host01.

```
$ startServer2.sh
```

Note: This script uses `ssh` to open a new terminal window on the host02 machine, and starts server2 on host02.

f. Wait for both managed servers to start.

g. Start OHS by executing the following command in the MAIN terminal window **on host01**. OHS is already configured for this course.

Note: The scripts that you run are built for this course to make starting and stopping OHS convenient for you. If you would like to learn how to start and stop OHS manually, you can review the scripts to see how they work.

```
$ startohs.sh
```

First, the script starts the OHS Node Manager in its own window.

```
. . .
Feb 01, 2016 3:44:34 AM oracle.ohs.plugin.nodemanager.OhsDomain
<init>
INFO: Domain initialized for /u01/domains/ohs
<Feb 4, 2016 3:44:34 AM PDT> <INFO> <Plain socket listener
started on port 5556, host localhost>
```

Next, it uses that Node Manager to start the `ohs1` system component.

```
Starting system Component ohs1 ...

Initializing WebLogic Scripting Tool (WLST) ...
. . .
Reading domain from /u01/domains/ohs

Please enter Node Manager password:
Connecting to Node Manager ...
Successfully Connected to Node Manager.
Starting server ohs1 ...
Successfully started server ohs1 ...
Successfully disconnected from Node Manager.

Exiting WebLogic Scripting Tool.
```

Finally, the script checks the status of OHS. You should see a RUNNING message.

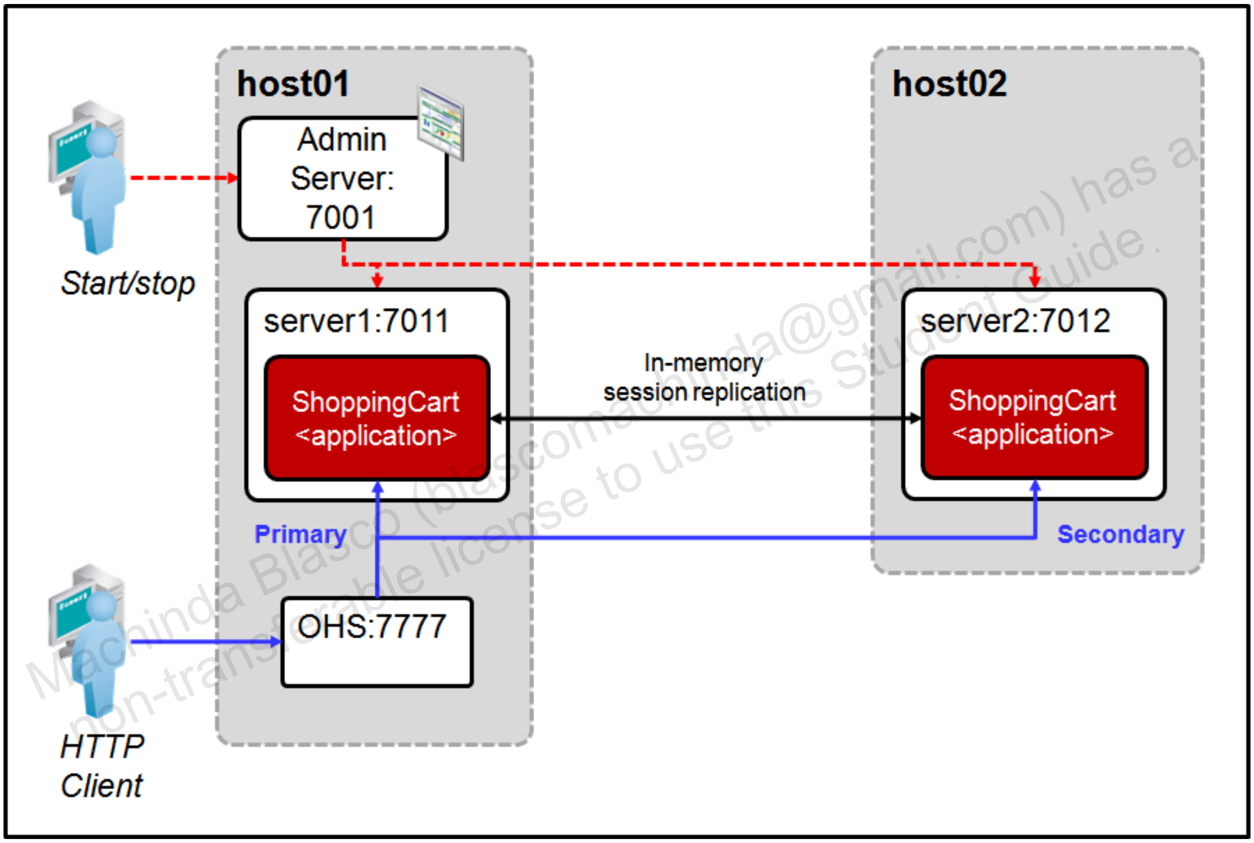
```
Initializing WebLogic Scripting Tool (WLST) ...
. . .
Connecting to Node Manager ...
Successfully Connected to Node Manager.

RUNNING
```

Practice 3-2: Performing a Rolling Upgrade

Overview

The company's ShoppingCart application is running on the current version of WebLogic Server. A new patch has been released that must be applied to the WebLogic Server product bits. The problem is that the application is running and is not scheduled for maintenance anytime soon. This patch must be applied now while the server is running.

The ShoppingCart Java EE application runs on a WebLogic domain within a cluster that spans two machines. The goal of this practice is to upgrade WebLogic Server by performing a rolling upgrade, which involves systematically shutting down the servers on one machine at a time to ensure that the application continues to run while customers are using it.



Tasks

1. Connect to the **host01** and **host02** machines.
2. Run the practice setup scripts.

   Note: It is important that when you complete this practice, you close the terminal windows and begin with new terminal windows for any subsequent practices. This is because the `ORACLE_HOME` environment variable is typically set for using the database, but is set differently for this practice.

   a. In **a terminal window on host01**, navigate to the `practice03-02` folder and execute the `setup.sh` script. Ensure that you include the initial period (.) in the command which causes any environment variables to persist in your terminal window:

      Note: If you are running the practices for the first time, the system will prompt you to enter the WebLogic user password. Use weblogic user password from the credentials section. The password will be captured and used for later exercises. Note that this script stores the password in **/practices/part2/.wlspwd** which can be deleted or edited if errors occur.

```
$ cd /practices/part2/practice03-02
$ . ./setup.sh
. . .
$ Enter the Oracle WebLogic password, followed by [ENTER]:

  Password updated successfully
```

      The script performs the following:

      - Captures the WebLogic password for later use
      - Reuses the domain and servers started in practice 3-1
      - Sets `ORACLE_HOME` to the FMW directory
      - Deploys the ShoppingCart application for this practice
      - Puts the Oracle OPatch tool in the system `PATH`

   b. In **a terminal window on host02**, navigate to the `practice03-02` folder and execute the `setOPatchEnv.sh` script. Ensure that you include the initial period (`.`) in the command, which causes any environment variables to persist in your terminal window:

```
$ cd /practices/part2/practice03-02
$ . ./setOPatchEnv.sh
```

      The script performs the following:

      - Sets `ORACLE_HOME` to the FMW directory
      - Puts the Oracle OPatch tool in the system `PATH`

3. Verify the domain's configuration.

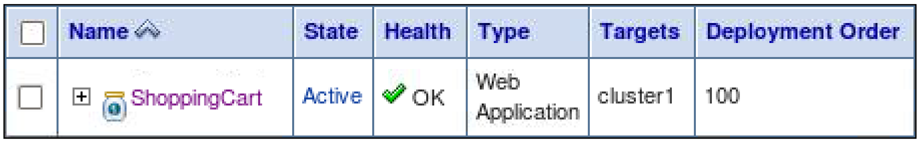
   Note: All browser steps should be run from either **host01** or **host02** for this course.

   a. Launch a web browser and log in to the administration console:
      `http://host01:7001/console`

   b. Login using the WebLogic Server credentials from the Security Credentials section.

   c. In the Domain Structure panel, click Deployments.

d. Verify that the Shopping Cart application is deployed.



□	Name	State	Health	Type	Targets	Deployment Order
□	⊞ ShoppingCart	Active	✓ OK	Web Application	cluster1	100

4. Run the Shopping Cart application.

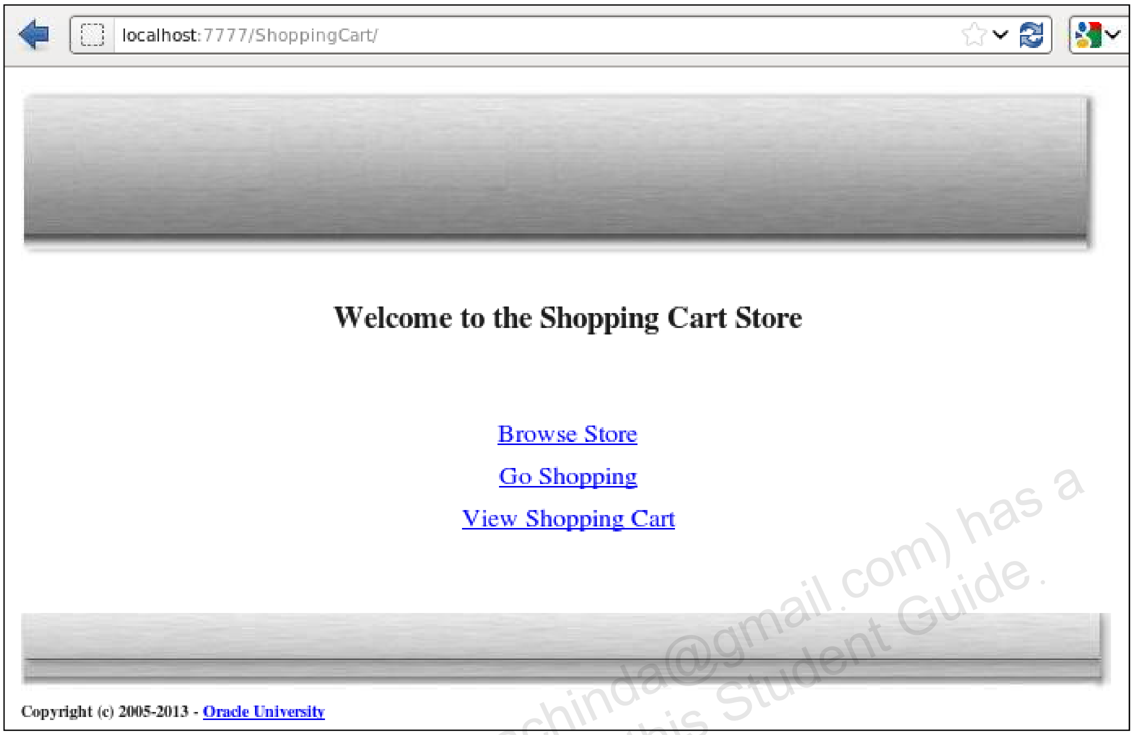
a. Open a new tab in the web browser and run the application by using the OHS networking address:

`http://host01:7777/ShoppingCart`

Note. If OHS does not redirect to the shopping cart application on host01 run the **stopohs.sh** command followed by the **startohs.sh** commands as shown below:

```
$ stopohs.sh
$ startohs.sh
```

b. You should see the main landing page, which shows that OHS and the application are working properly:



Welcome to the Shopping Cart Store

Browse Store

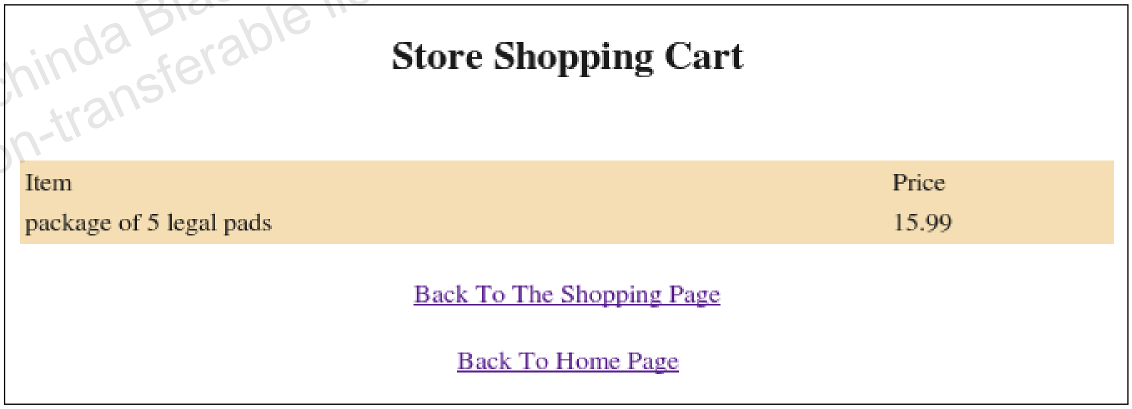
Go Shopping

View Shopping Cart

Copyright (c) 2005-2013 - Oracle University

c. Click the *Go Shopping* link and purchase some items from the store to get the items placed into your shopping cart.

d. Click the *View Shopping Cart* link to view your shopping cart. Take note of what items are in your cart:



Store Shopping Cart

Item	Price
package of 5 legal pads	15.99

Back To The Shopping Page

Back To Home Page

5. Back up before starting the upgrade process.

   Note: You will skip the backup process because this is a practice environment. Backing up the `fmw` folder takes a considerable amount of time and is not necessary for this environment. You also skip the backup process for the domain because there are no patches made directly to the domain itself. Remember that in a real production environment, you should always back up your system before performing an upgrade to ensure you can quickly and easily revert to a known working state.

6. Prepare the patch for use.

   Perform the following steps in the terminal window **on host01** where you ran the `setup.sh` script, to prepare the patch:

a. Create a directory in the current practice folder:

```
$ mkdir PATCH_TOP
```

b. Unzip the patch to this folder:

```
$ cd PATCH_TOP
$ unzip
    /install/weblogicpatch/p22248372_122100_Generic.zip -d .
```

Note: Ensure that you enter the command on a single line.

c. Review the README file but DO NOT execute its instructions because a rolling upgrade is done slightly differently:

```
$ cd 22250572
$ gedit README.txt
```

7. Apply the patch on the **host01** machine by using the rolling upgrade method.

a. Shut down the servers **on host01** by pressing **Ctrl + C** in the terminal windows for `AdminServer` and `server1`. **DO NOT CLOSE THE TERMINAL WINDOWS BECAUSE YOU WILL REUSE THEM!**

b. Use the Shopping Cart application while `server1` is down to add a new item to your cart. This simulates a user continuing to use the application while the system is running, and causes the secondary HTTP session to become the primary HTTP session if `server1` was the primary server.

c. Execute OPatch to apply the patch and answer `y` when it asks if the system is ready for patching (still using the terminal window where the `setup.sh` script was executed):

```
$ opatch apply -jdk $JAVA_HOME
```

Your output should resemble the following:

```
Oracle Interim Patch Installer version 13.3.0.0.0
Copyright (c) 2016, Oracle Corporation.  All rights reserved.


Oracle Home       : /u01/app/fmw/wlserver
Central Inventory : /u01/app/oraInventory
   from           : /u01/app/fmw/wlserver/.../oraInst.loc
OPatch version    : 13.3.0.0.0
OUI version       : 13.3.0.0.0
Log file location : /u01/app/fmw/. .
./cfgtoollogs/opatch/22250572_Feb_01_2016_07_39_26/apply2016-02-
01_07-39-20AM_1.log



OPatch detects the Middleware Home as "/u01/app/fmw"


Verifying environment and performing prerequisite checks...
Patch continues with these patches: 22250572


Please shutdown Oracle instances running out of this ORACLE_HOME
on the local system.
```

```
(Oracle Home = '/u01/app/fmw')


Is the local system ready for patching? [y|n]
y
User Responded with: Y
Backing up files...
Applying interim patch '22250572' to OH
'/u01/app/fmw/wlserver/..'
Patching component oracle.wls.core.app.server, 12.2.1.0.0...
. . .

Patch 22250572 successfully applied
Log file location:
/u01/app/fmw/wlserver/../cfgtools/opatch/22250572_Feb_01_2016_07
_39_26/apply2016-02-01_07-39-20AM_1.log

OPatch succeeded.
```

d. Start `AdminServer` **on host01** again by repeating its startup command in the **AdminServer** window and wait for it to fully start.

```
$ ./startWebLogic.sh
```

e. Start the `server1` managed server **on host01** again by repeating its startup command in the **server1** window and wait for it to fully start.

```
$ ./startManagedWebLogic.sh server1 host01:7001
```

f. You are now half way done with upgrading your WebLogic environment.

8. Apply the patch on the **host02** machine by using the rolling upgrade method.

a. Use the Shopping Cart application again now that the servers on host01 and host02 are running. Again, this simulates a user continuing to use the application while the system is running, and causes WebLogic to ensure that there is a secondary backup session for your shopping cart items. Be sure to click *View Shopping Cart* to verify that you have not lost your session yet.

b. Shut down the server **on host02** by pressing **Ctrl + C** in the terminal window for `server2`. **DO NOT CLOSE THE TERMINAL WINDOW BECAUSE YOU WILL REUSE IT!**

c. Execute OPatch to apply the patch and answer `y` when it asks if the system is ready for patching. This command must be executed in the same terminal window where you ran `setOPatchEnv.sh`.

Note: The `practices` folder is a shared disk that is accessible from both host01 and host02. If you are not using a shared disk environment, you would have to copy the files manually to host02.

```
$ cd /practices/part2/practice03-02/PATCH_TOP/22250572
$ opatch apply -jdk $JAVA_HOME
```

You should see the same output you saw when you applied the patch on host01.

d. Start the `server2` managed server **on host02** again by repeating its start up command and wait for it to fully start.

```
$ ./startManagedWebLogic.sh server2 host01:7001
```

9. Test the Shopping Cart application to check whether your cart still contains all the items that it should contain.

a. Use the Shopping Cart application again now that the servers on host01 and host02 are running and upgraded. Be sure to click *View Shopping Cart* to verify that you have not lost your session yet. If you can see your shopping cart contents, then congratulations; you have successfully performed a rolling upgrade of your domain without causing any client outages.

Note: The rolling upgrade feature enables you to update a live system even while clients are using the system. Remember that only active clients that have a valid primary or secondary session available in the domain while servers are shut down and started again will continue to function with their sessions intact. Idle clients that have timed out will experience a lost session and will have to start using the application from the beginning again. If a reasonable amount of time is taken in-between upgrading different machines in the domain, you give your clients time to keep their sessions active.

10. **OPTIONAL:** Roll back the patch on both machines by using the rolling upgrade method.

a. Perform the same steps that you performed to do the rolling upgrade in reverse order. This is because the `AdminServer` must always run the highest software version in the domain. This time when you perform a rolling downgrade, you run the `OPatch rollback` command instead of the `apply` command. See the `README` file for details.

11. Shut down the environment.

a. Press **Ctrl + C** in all terminal windows that are running servers.

b. Close all terminal windows, including the MAIN entitled windows. This is to ensure that the environment is set properly for the next practice.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Open a new terminal window **on host01**.
2. Change the current directory to the current practice folder.
3. Execute the solution script: (Do not forget the first [.].)

   ```
   $ . ./solution.sh
   ```

   Note: This script behaves differently on each host, performing what is required for each.
4. The solution script performs the following:
   a. Resets the domain to its original state
   b. Starts all of the wlsadmin domain's servers
   c. Sets up the domain by setting the environment and deploying the Shopping Cart application
   d. Prepares the OPatch patch by creating the PATCH_TOP folder and unzipping the patch to it
   e. Opens a window in the practice folder on host02
5. Set the environment in the new terminal window opened by the solution script **on host02**:
   a. Set the environment and navigate to the patch folder: (Do not forget the first [.].)

      ```
      $ . ./setOPatchEnv.sh
      $ cd PATCH_TOP/22250572
      ```
6. Wait for servers **on host01 and host02** to fully start.
7. Continue starting at step 3 and be sure to **skip step 6**.

   Note: No other practices depend on this practice.

Practices for Lesson 4: Creating and Using Domain Templates

Chapter 4

Practices for Lesson 4: Overview

Practices Overview

In the practices for this lesson, you explore the capabilities of the Domain Template Builder and Configuration Wizard tools. These practices also give you the opportunity to work with some custom WLST scripts.

Practice 4-1: Creating and Using a Custom Domain Template

Overview

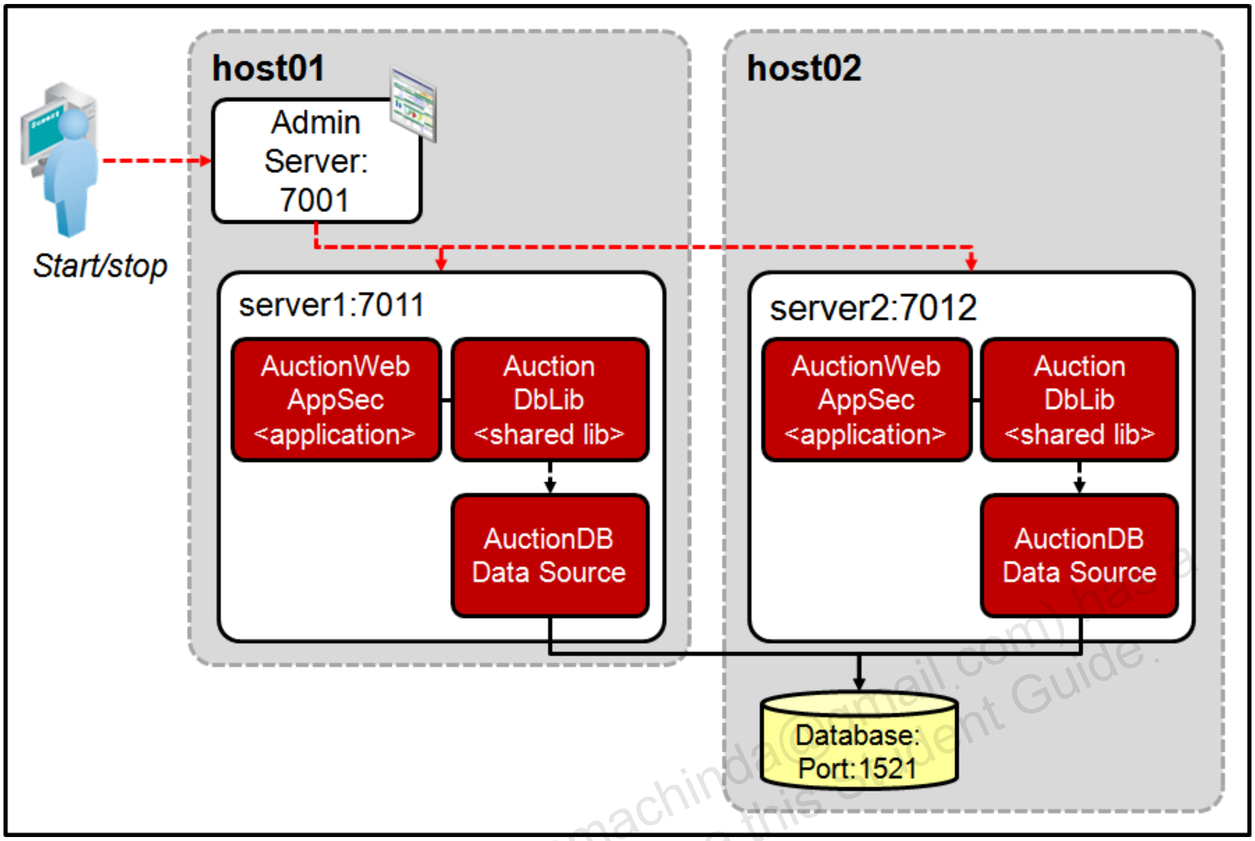
The Auction Java EE application provides a simple application that contains artifacts commonly managed by WebLogic administrators. Users can create the initial data set, which is then maintained and used by the application, by clicking a link on the welcome page. Users can list auctions, create auctions, bid on auctions, and so on.

The Auction application is available in several forms in this course. You do not need to memorize this because the course will explain each as it is encountered.

Form	Uses Database	Uses Security
SimpleAuctionWebApp	N	N
SimpleAuctionWebAppDb	Y	N
SimpleAuctionWebAppSec	N	Y
SimpleAuctionWebAppDbSec	Y	Y
AuctionWebApp	Y if referencing AuctionDbLib N if referencing AuctionMemLib	N
AuctionWebAppSec	Y if referencing AuctionDbLib N if referencing AuctionMemLib	Y

Some forms of the Auction application, like most Java EE applications, are dependent on various server resources. These resources include Java Database Connectivity (JDBC) data sources and shared Java EE libraries. In this practice, you will create the necessary domain infrastructure to support the Auction application and package it within a domain template by using the Domain Template Builder tool. The schema required by the Auction application will also be bundled within the domain template as SQL scripts, so that administrators can quickly initialize any relational database used to support the domain.

The Auction domain infrastructure is depicted here:



Tasks

1. Connect to the **host01** and **host02** machines.
   a. Connect via VNC to **host01**.
   b. Connect via VNC to **host02**.
2. Set up the practice environment.

   Note: Because you have already started the domain manually in `practice03-01`, the setup script now automatically starts the servers of the domain on each machine.

   a. Open a terminal window **on each machine** by clicking the terminal icon in the launch panel located at the top of the screen. Set the title for each terminal window to MAIN.
   b. In the terminal window on **host01**, navigate to the `practice04-01` folder:

   ```
   $ cd /practices/part2/practice04-01
   ```

   c. Execute the `setup.sh` script to set up the initial practice environment:

   ```
   $ ./setup.sh
   ```

   Note: If you are running the practices for the first time, the system will prompt you to enter the WebLogic user password. Use weblogic user password from the credentials section. The password will be captured and used for later exercises.

   ```
   $ Enter the Oracle WebLogic password, followed by [ENTER]:
   . . .
     Password updated successfully
   ```

   This script performs the following:

- Ensures that no previous servers are running on both machines
- Starts the `wlsadmin` domain on host01
- Ensures that no applications or libraries are deployed
- Deploys the starting application used for this practice
- Deletes the application database tables
  Why? Because this practice demonstrates the use of SQL scripts as part of a domain template, you must remove the data related to the Auction application from the database.
- Starts `server2` of the `wlsadmin` domain on host02

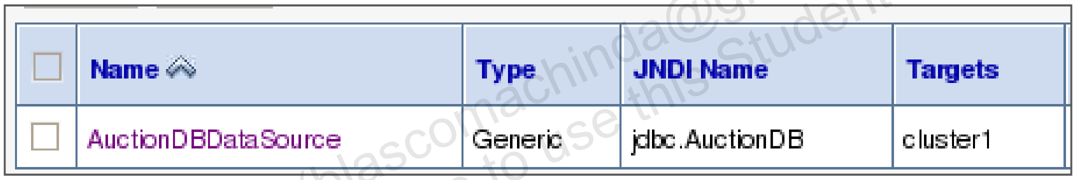
d. Wait for all servers on both machines to start before continuing on to the next step. You can ignore any errors that appear as long as the servers are all in the RUNNING state, and the practice steps are working for you.

3. Verify the domain's configuration.

   a. Launch a web browser and log in to the Administration Console:
   `http://host01:7001/console`

   b. In the Domain Structure panel, navigate to **Services > Data Sources**.

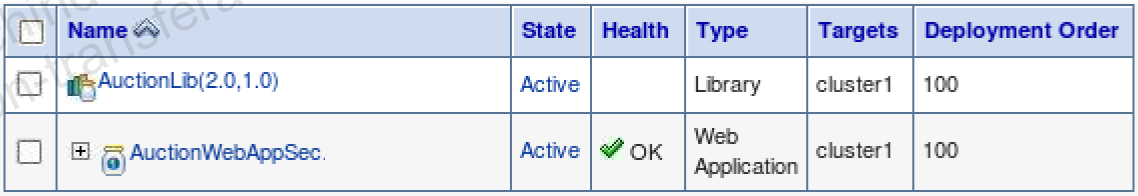
   c. Verify that the AuctionDataSource data source exists and is targeted to cluster1.



□	Name	Type	JNDI Name	Targets
□	AuctionDBDataSource	Generic	jdbc.AuctionDB	cluster1

   d. In the Domain Structure panel, click **Deployments**.

   e. Verify that the AuctionLib shared library and AuctionWebAppSec application are deployed.



□	Name	State	Health	Type	Targets	Deployment Order
□	AuctionLib(2.0,1.0)	Active		Library	cluster1	100
□	⊞ AuctionWebAppSec	Active	✓ OK	Web Application	cluster1	100

4. Shut down the domain.

   In order to create a template of the `wlsadmin` domain, it is safest if it is shut down.

   a. Shut down the `AdminServer`, `server1`, and `server2` servers. Press Ctrl + C in each terminal window to quickly achieve this. **DO NOT CLOSE THE TERMINAL WINDOWS BECAUSE YOU WILL REUSE THEM!**

5. Create a custom template using the existing `wlsadmin` domain.

   a. In the MAIN terminal window **on host01**, execute the following commands to start the WebLogic Template Builder tool:

```
$ cd /u01/app/fmw/oracle_common/common/bin
$ ./config_builder.sh
```

   b. Verify that the *Create Domain Template* option is selected.

   c. Verify that *Use Domain as a Source* is selected.

   d. Ensure that *Source Location* is set to `/u01/domains/part2/wlsadmin`.

e. Set *Template Location* to `/practices/part2/practice04-01/template/AuctionTemplate.jar`.

f. Click **Next**.

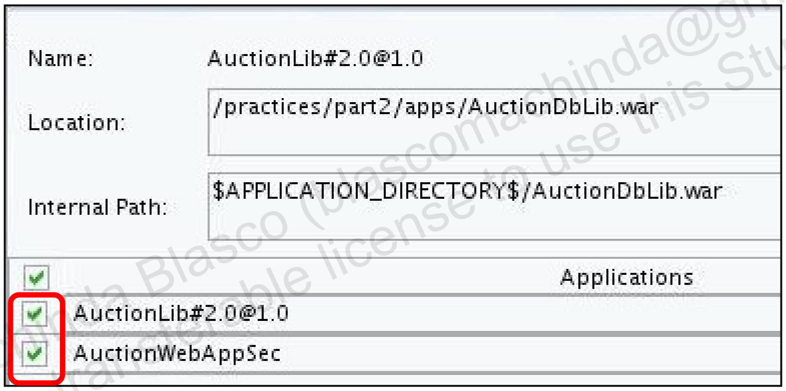
g. Enter the following values on the Template Information page:

Field	Value
Name	AuctionDomain
Version	12.2.1.0.0
Author	Oracle Corporation
Category	Customer-facing
Description	Production domain to support auctions

Click **Next**.

h. Review the information on the Applications page. Take note of the Location and Internal Path fields for each application. Verify that there are two applications, one named *AuctionLib*, which is a shared library, and another *AuctionWebAppSec*, which is its referencing library.

i. Confirm that the two applications are selected and click **Next**:



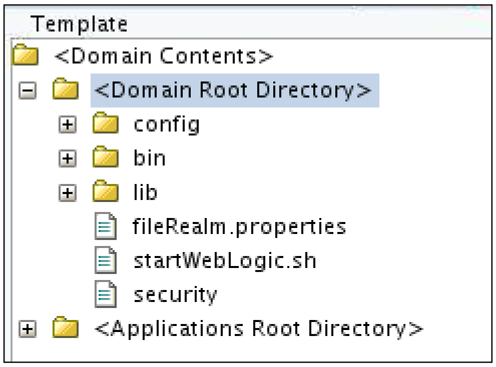
Name: AuctionLib#2.0@1.0

Location: /practices/part2/apps/AuctionDbLib.war

Internal Path: $APPLICATION_DIRECTORY$/AuctionDbLib.war

	Applications
☑	AuctionLib#2.0@1.0
☑	AuctionWebAppSec

j. In the right panel labeled Template, expand and select **<Domain Root Directory>**:



```
Template
<Domain Contents>
⊟ <Domain Root Directory>
    ⊞ config
    ⊞ bin
    ⊞ lib
      fileRealm.properties
      startWebLogic.sh
      security
⊞ <Applications Root Directory>
```

k. In the left panel labeled File System, locate and select the `/practices/part2/practice04-01/resources/bin/startServer1.sh` file.

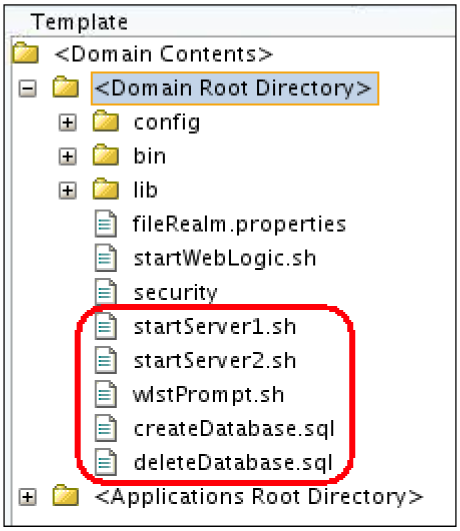
l. Click the **Add** button.

m. Repeat the previous steps to add the following additional files to the template:

– `/practices/part2/practice04-01/resources/bin/startServer2.sh`

– `/practices/part2/practice04-01/resources/bin/wlstPrompt.sh`

- /practices/part2/practice04-01/resources/jdbc/createDatabase.sql
- /practices/part2/practice04-01/resources/jdbc/deleteDatabase.sql



n. Click **Next**.

6. Configure variable replacement for custom template files.

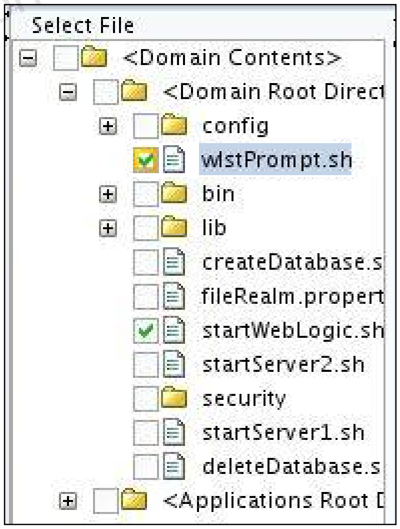
   a. Launch a text editor and inspect the contents of the /practices/part2/practice04-01/resources/bin/wlstPrompt.sh file.

   b. Notice that the path to your WebLogic installation is hard-coded, as in the following example:

   /u01/app/fmw/wlserver

   c. Close the file and return to the Domain Template Builder.

   d. If not already selected, select the check box for the wlstPrompt.sh file. When the check box is already selected it means that the Template Builder has already detected and substituted a variable in the file.



   e. Select wlstPrompt.sh and click the **Edit** button.

f. Notice that the preceding hard-coded text has automatically been replaced with a token (`@oracle.wls…`). A new file is generated that contains these updates:

```
/practices/part2/practice04-01/resources/bin/_edit_wlstPrompt.sh
```

g. Click **Next**.

h. Review the template's contents and click **Create**.

i. When finished, click **Next**, and then click **Finish**.

7. Test the custom template by using the Configuration Wizard.

You will create a new domain based on the template you just created. This new domain will sit in a new folder alongside the existing `wlsadmin` domain.

a. Perform the following commands to launch the Configuration Wizard:

```
$ cd /u01/app/fmw/oracle_common/common/bin
$ ./config.sh
```

b. Make the following selections and entries on the *Configuration Type* page:

Field	Value
What do you want to do?	Create a new domain.
Domain Location	`/u01/domains/part2/AuctionDomain`

Click **Next**.

Select the *Create Domain Using a Custom Template* option, enter the following template location, and click **Next:**

```
/practices/part2/practice04-01/template/AuctionTemplate.jar
```

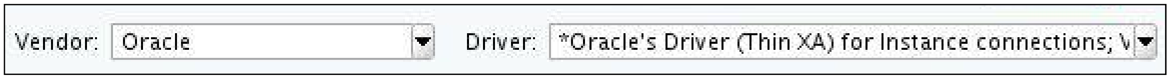
c. Enter the following application location and click **Next**:

```
/u01/domains/part2/AuctionDomain/apps
```

d. Enter the WebLogic Server credentials from the credentials section and click **Next**.

e. Select the Production Mode option, the JDK option, and click **Next**.

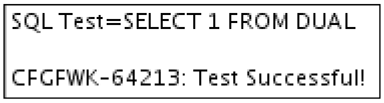
f. Confirm that the domain contains a single data source. Select the check box, verify that its Vendor field is set to `Oracle`, and click **Next**.
Note: If you were using the template to create a domain that uses a different database, you would change the settings of this page to match your target database settings:



Vendor: Oracle

Driver: *Oracle's Driver (Thin XA) for Instance connections; V

g. The data source will be automatically tested. Confirm that the test was successful:



```
SQL Test=SELECT 1 FROM DUAL

CFGFWK-64213: Test Successful!
```

h. Click **Next**.

i. You can optionally select any of the options to see more configuration settings for the Administration Server, Node Manager, and so on. However, it is not necessary as the settings are already identical to the original domain based on the template's settings. Click **Next** to skip the Optional Configuration page.

j. Click **Create** to create your new domain.

k. Wait for the domain creation process to complete, click **Next**, and **Finish**.

8. Perform post-creation tasks:
   a. Create the JAR file for remote managed servers by using the `pack` command **on host01**:

      ```
      $ cd /u01/app/fmw/oracle_common/common/bin
      $ ./pack.sh -domain=/u01/domains/part2/AuctionDomain -
      template=/practices/part2/practice04-01/AuctionDomainManaged.jar
      -template_name="Auction Template" -managed=true
      ```

   b. Verify that your practice folder contains the `AuctionDomainManaged.jar` file.
      Note: Normally, this process requires you to copy the template JAR file to the other machines in the domain. For convenience, the `practices` folder is shared between host01 and host02, so copying is not required.

   c. Create the remote managed server domain side using the `unpack` command **on host02**:

      ```
      $ cd /u01/app/fmw/oracle_common/common/bin
      $ ./unpack.sh -domain=/u01/domains/part2/AuctionDomain -
      template=/practices/part2/practice04-01/AuctionDomainManaged.jar
      ```

   d. Verify that the AuctionDomain domain is now present **on host02**:
      `/u01/domains/part2/AuctionDomain`

   e. Perform a listing of the `AuctionDomain` folder **on host02**. You should notice that the SQL scripts were not copied as part of the `pack` and `unpack` process. You must manually copy these scripts so you can use them on host02. The easiest way is to just copy them directly from the practice's shared resources folder:

      Host02:

      ```
      $ cd /u01/domains/part2/AuctionDomain
      $ cp /practices/part2/practice04-01/resources/jdbc/* .
      ```

      Note: Do not forget the (`.`) on the end of the command. This tells the shell to copy the files to the current directory location.

   f. Perform the following steps **on host02** to create the database required for the Auction application(Using the Oracle Database credentials):

      ```
      $ sqlplus dbuser/dbpassword @createDatabase.sql
      ```

9. Start your new domain.
   a. In the `AdminServer` terminal window **on host01**, perform the following command to start your administration server:

      ```
      $ cd /u01/domains/part2/AuctionDomain
      $ ./startWebLogic.sh
      ```

      Note: If you selected *Production Mode* for the domain during creation, then you must enter the username and password when the server is starting. This is also true for both managed servers. Remember that the login credentials are listed in the credentials section. You can also ignore any errors you encounter as long as the server displays the *in RUNNING mode* message.

   b. In the `server1` terminal window **on host01**, start `server1` by using the following script:

      ```
      $ cd /u01/domains/part2/AuctionDomain
      $ ./startServer1.sh
      ```

c. In the `server2` terminal window **on host02**, start `server2` by using the following script:

```
$ cd /u01/domains/part2/AuctionDomain
$ ./startServer2.sh
```

d. Verify domain settings using the WebLogic Server Administration Console:

– `http://host01:7001/console`

Note: Log in using the WebLogic Server credentials .

– Note that the domain name is AuctionDomain

– Select **Environment > Servers** and verify that all three servers are running.

– Verify that each managed server is part of `cluster1`.

– Select Deployments and verify that the Auction application and library are deployed (Active) and targeted to `cluster1`.

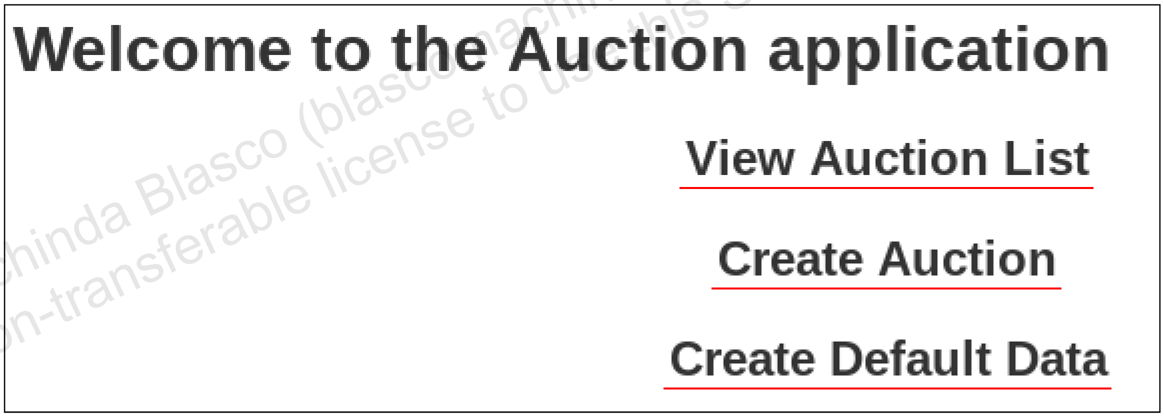
– Select **Services > Data Sources** and verify that the *AuctionDBDataSource* data source is configured and is targeted to `cluster1`.

10. Test the Auction application.

a. Direct your web browser to the following URL:

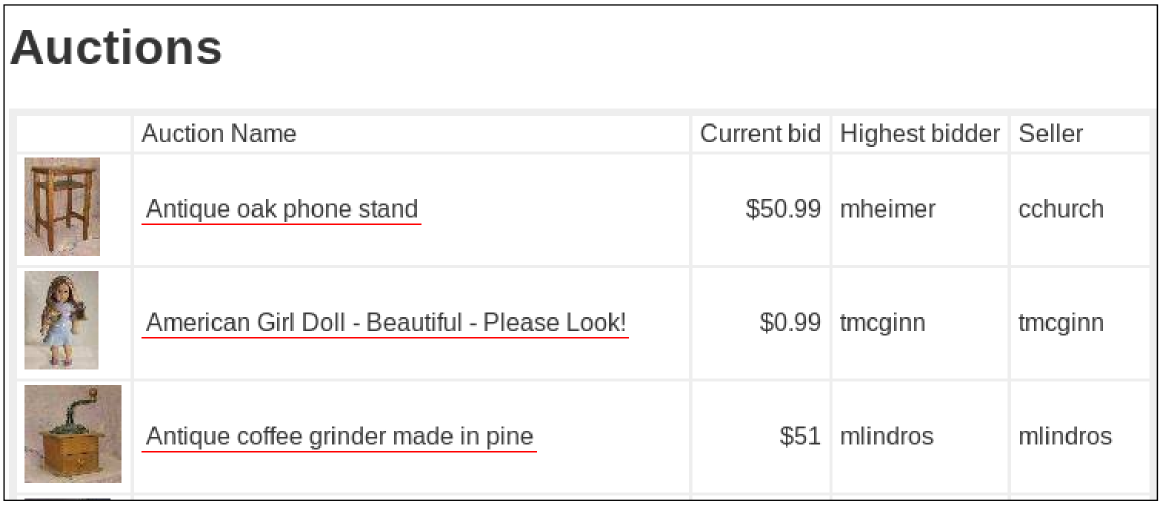
`http://host01:7011/AuctionWebAppSec/index.jsp`

b. Click the *Create Default Data* link to populate the database with data for the Auction application. Follow this by clicking the confirmation to create the data.



Welcome to the Auction application

c. Click the *Go Home* link, and then click the *View Auction List* link to view the list of auctions stored in the database.



Auctions

	Auction Name	Current bid	Highest bidder	Seller
	Antique oak phone stand	$50.99	mheimer	cchurch
	American Girl Doll - Beautiful - Please Look!	$0.99	tmcginn	tmcginn
	Antique coffee grinder made in pine	$51	mlindros	mlindros

If you see the list of auction items, then congratulations; you have successfully created and used a WebLogic domain template.

11. Shut down the environment.
    a. Press **Ctrl + C** in all terminal windows that are running servers.
    b. Close all terminal windows used for each server. You can leave the MAIN terminal windows open if you are continuing on to another practice.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Open a new terminal window on **on host01**:

2. Change the current directory to the current practice folder.
   ```
   $ cd /practices/part2/practice04-01
   ```

3. Execute the solution script:
   ```
   $ ./solution.sh
   ```

   When prompted for the username enter the Oracle Database Server credentials from the credentials section, example:

   ```
   …
   Type the Oracle Database username, followed by [ENTER]: dbuser
   Type the Oracle Database password, followed by [ENTER]: dbpass
   ```

   Note: This script behaves differently on each host, performing what is required for each.
4. The solution script performs the following:
   a. Unpackages the solution domain contents to `/u01/domains/part2/AuctionDomain`
   b. Creates Auction database artifacts

5. Continue starting at step number 9.
   Note: Other practices do not depend on this practice. Therefore, the solution is not required to work on other practices.

Practices for Lesson 5: WebLogic Server Startup and Crash Recovery

Chapter 5

Practices for Lesson 5: Overview

Practices Overview

In the practices for this lesson, you learn how to configure a system to automatically start and restart an entire WebLogic domain that spans multiple machines.

Practice 5-1: Configuring Automatic Start and Restart of a System

Overview

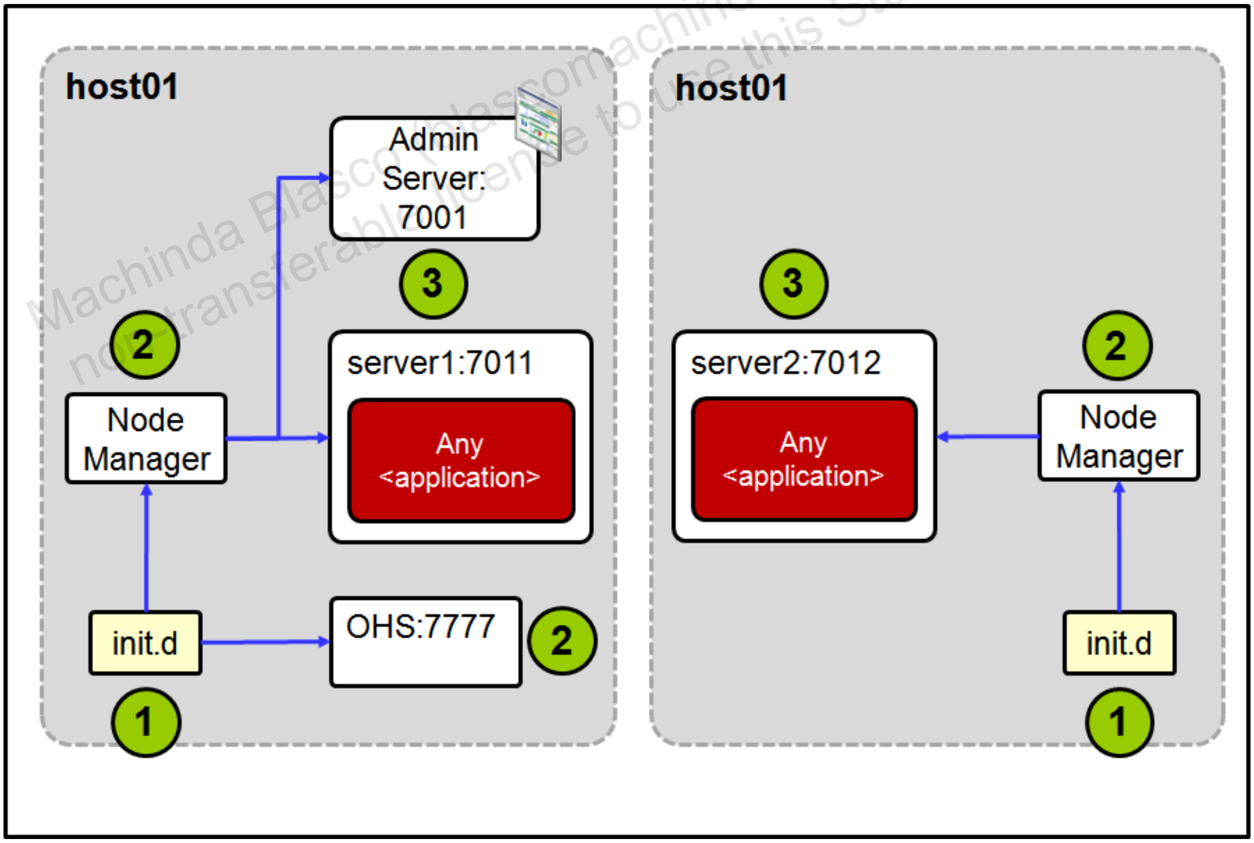
By default, the Node Manager processes responsible for the servers in a domain automatically restart any servers that they started if they shut down ungracefully. There is no mechanism in place by default to do the same for restarting the Node Manager processes in the event that they crash or the machine they are running on fails. And although Node Managers automatically restart failed servers, they do not automatically restart servers in the event of a machine failure.

This practice shows you how to configure your systems and WebLogic to automatically recover the entire system regardless of any Node Manager, server, or machine failure.

This practice focuses on:

- Configuring WebLogic processes to automatically start or restart when the host01 machine starts
- Crashing the host01 machine while the Node Manager, `AdminServer`, and `server1` servers are running
- Starting the host01 machine again
- Checking to see if the WebLogic Node Manager and server processes start automatically

Note: This practice does not perform any related configuration on the host02 machine because it is repetitive.



In the event of a machine failure, on either machine, the system automatically restarts as follows:

1. When the machine starts up, it automatically starts all `init.d` programs configured to start at boot time.
2. The `init.d` system is configured to automatically restart the Node Manager, Oracle HTTP Server (OHS), and the database if any is required (not pictured here).

   Note: This practice does not configure OHS or the database to start automatically. This statement is for informational purposes only. Again, the diagram depicts using `init.d` and Node Manager to start the entire domain, but this practice focuses only on host01.
3. When the Node Manager process starts, it is configured to restart servers as part of a crash recovery process. The Node Manager recognizes that the servers it controls shut down unexpectedly and automatically restarts them all, including the administration server. This ensures that the entire system is ready to process client requests again.

Tasks

1. Connect to the **host01** and **host02** machines.
2. Set up the practice environment.
   a. You may reuse the MAIN terminal windows from previous practices. Otherwise, open a terminal window **on each machine** by clicking the terminal icon in the launch panel located at the top of the screen.
   b. Set the title of each terminal window by selecting Terminal > Set Title and entering `MAIN` as the title. This makes it easier to distinguish the purpose of each window.
   c. Navigate to the practice folder **(host01 only)**:

   ```
   $ cd /practices/part2/practice05-01
   ```

   d. Execute the `setup.sh` script to set up the initial practice environment:

   ```
   $ ./setup.sh
   ```

   This script performs the following:
   – Ensures that no previous servers are running on both machines
   – Restores the domain and the practice to their original state
   – Ensures that Node Manager is not configured for crash recovery or as a system service
   – Removes the server state files on both machines that are associated with how Node Manager determines if it should automatically start a server.

   Note: You can ignore any errors regarding an unrecognized service or files not being found. This just means that Node Manager is not already configured to run as a service.
3. Configure the Node Manager to run as an `init.d` service.
   a. Open the `init.d` configuration script for Node Manager **on host01**:

   Note: This is a script you must write manually. It is not supplied with the product.

   ```
   $ gedit resources/nodemgr
   ```

   b. Review the script to learn how it is configured:
      1) The `### BEGIN INIT INFO` section instructs the `init.d` system how to control Node Manager during the different user run levels for OS start up and shut down. It also says that the network and local file systems are required to be operational and available before starting the Node Manager.

2) The `. /etc/rc.d/init.d/functions` statement sources `init.d` functions that may be used in the script.

3) The next part of the script sets some variables that are used to start and stop the Node Manager process:

Variable	Purpose
`MW_HOME`	The FMW directory
`JAVA_HOME`	The Java SE directory
`DAEMON_USER`	The OS username to use to execute the process
`PROCESS_STRING`	The regular expression used to check whether or not Node Manager is already running
`NODEMGR_HOME`	The Node Manager home directory
`NodeManagerLockFile`	The lock file created when Node Manager runs. This is used to ensure that processing is performed properly.
`PROGRAM`	The program to run as a service
`SERVICE_NAME`	The name of this service
`LOCKFILE`	A lock file used by this script to conditionally control when to restart the Node Manager. If the lock file does not exist, there is no need to stop a Node Manager process that is not running.

4) The next part of the script contains the functions called by `init.d` to start, stop, and restart the Node Manager. It uses the variables described in the table above to make proper decisions; such as, whether the service command should be performed, based on the current state of the Node Manager.

c. Close the file, or keep it open for reference.

d. Register the script with `init.d`, make it executable, and verify its configuration.

Note: The machines for this course have configured the `oracle` user to enable running commands using `root` permissions by using the `sudo` command. Using `sudo` helps to avoid accidentally running certain commands as the `root` user, which can cause problems. When you run `sudo`, you will not be required to enter a password so be careful.

```
$ sudo cp resources/nodemgr /etc/init.d/
$ sudo chmod 755 /etc/init.d/nodemgr
$ sudo chkconfig --add nodemgr
$ sudo chkconfig --list nodemgr
```

The output for listing the service should resemble the following. The numbers represent the run levels of the operating system, and the text specifies whether the service runs in that run level.

```
nodemgr    0:off 1:off 2:off 3:on 4:on 5:on 6:off
```

4. Configure WebLogic servers to automatically restart after a crash.
   a. Navigate to the Node Manager home folder, and open the properties file for editing:

   ```
   $ cd /u01/domains/part2/wlsadmin/nodemanager
   $ gedit nodemanager.properties
   ```

   b. Set the `CrashRecoveryEnabled` property to `true`.
   c. Save and close the file.
5. Start the Node Manager on host01 as a service.

   ```
   $ sudo service nodemgr start
   ```

   Note: Press enter in the terminal window to return to the command prompt. If you see a message that Node Manager is already running, then continue with the next step.
6. Review service output for automatic server start ups.

   This should not happen if you performed these steps in order. However, you may notice that some or all of the servers start automatically. This is because previously they may have shut down unexpectedly and your Node Manager service has automatically restarted them. If this is the case, then everything is ok. Simply continue the steps and ignore any errors or warnings that may occur as long as your servers are all running. You can tell this by looking at the output for your Node Manager service. If you see the following, then servers have started automatically:

   ```
   <The server '<server-name>' is running now.>
   ```

7. Start domain using Node Managers.
   a. Perform the following commands to start the rest of the domain:

   On host02:

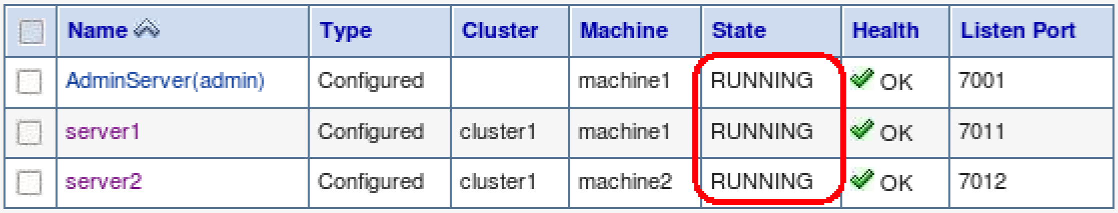
   ```
   $ wlst.sh startNM.py
   ```

   Why? You configure Node Manager to run as a service only on the host01 machine. You must still use a manual method to start Node Manager on host02.

   On host01:

   ```
   $ wlst.sh startDomain.py
   ```

   b. Verify that the entire domain is up and running by logging in to the Administration Console and checking the status of the `AdminServer`, `server1`, and `server2`. If all three are in the `RUNNING` state, continue to the next step.



□	Name	Type	Cluster	Machine	State	Health	Listen Port
□	AdminServer(admin)	Configured		machine1	RUNNING	✓ OK	7001
□	server1	Configured	cluster1	machine1	RUNNING	✓ OK	7011
□	server2	Configured	cluster1	machine2	RUNNING	✓ OK	7012

8. Simulate a crash of **host01** by restarting the machine.

   a. Execute the following command to restart host01:

   ```
   $ sudo reboot
   ```

   Why? If you did not already know, the host01 and host02 machines are VMs running on another environment. This course runs on multiple deployment platforms and the only way to regain control of your VM in some platforms is to restart the machine.

   b. The **host01** session should close with an error or may become unresponsive. Close the window used to access **host01** if it does not close.

9. Verify that everything restarted automatically.

   a. Wait for a few minutes to give the machine time to start. While you are waiting, here is some important information: Note that your servers only restart in this scenario if they were already running and were started by the Node Manager. Again, the Node Manager does not automatically restart servers it did not start or servers that were shut down gracefully. Now go ahead and see if the machine is running yet!

   b. Connect to **host01** again. Keep trying until you are successful. Ignore any errors that you may encounter while **host01** is going through its boot process.

   c. Open a new terminal window **on host01** and name it `MAIN`.

   d. Perform the following command to check if the correct processes are running:

   ```
   $ jps -lv
   ```

   You should see three or more processes appear, one for each server: Node Manager, `AdminServer`, and `server1`. If you look at the output of each entry, you will see the server or process name in each. Node Manager starts each server in order, so if you see only the `AdminServer` running, you still have to wait for it to reach the RUNNING state before the administration console will work or to see `server1` in the process list.

   ```
   2336 weblogic.Server -Xms256m -Xmx512m -XX:MaxPermSize=256m -
   Dweblogic.Name=server1 . . .
   1855 org.apache.derby.drda.NetworkServerControl -
   Dderby.system.home=/u01/domains/part2/wlsadmin/common/db
   1881 weblogic.Server -Xms256m -Xmx512m -XX:MaxPermSize=256m -
   Dweblogic.Name=AdminServer . . .
   2847 sun.tools.jps.Jps . . .
   1604 weblogic.NodeManager -Xms32m -Xmx200m -XX:MaxPermSize=128m
   -Dcoherence.home=/u01/app/fmw/coherence -Dbea.home=/u01/app/fmw
   -Xverify:none . . .
   ```

   e. Log in to the Administration Console and verify that all domain servers are in the `RUNNING` state again. If all the servers are up and running and you are able to use the administration console, then congratulations; you have successfully configured the Node Manager and WebLogic to automatically restart when a machine crashes!

10. Shut down the environment.

    a. Perform the following command to stop the `wlsadmin` domain.

    ```
    $ wlst.sh stopDomain.py
    ```

    b. Close all terminal windows used for any servers. You should still have MAIN terminal windows open on both machines.

11. Run the cleanup script.
    a. Navigate to the current practice folder **on host01** and execute the following script to clean up the practice environment. This script resets the practice to its original state, so there will not be any conflicts with other practices.

```
$ cd /practices/part2/practice05-01
$ ./reset.sh
```

The script performs all of the same tasks that were mentioned when you executed the `setup.sh` script.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Open a new terminal window **on host01**:
2. Change the current directory to the current practice folder.
3. Execute the solution script:

   ```
   $ ./solution.sh
   ```

   This script performs the following:

   - Ensures that no previous servers are running on both machines
   - Restores the domain and the practice to their original state
   - Ensures that Node Manager is not configured for crash recovery or as a system service
   - Removes the server state files on both machines that are associated with how Node Manager determines if it should automatically start a server.
   - Configures Node Manager as a system server, enables crash recovery, and starts the Node Manager service
   - Starts Node Manager on host02
   - Starts all servers in the domain
4. Wait for servers **on host01 and host02** to fully start.
5. Continue starting at step number 7b.

   Note: Other practices do not depend on this practice. Therefore, the solution is not required to work on other practices.

Practices for Lesson 6: WebLogic Scripting Tool (WLST)

Chapter 6

Practices for Lesson 6: Overview

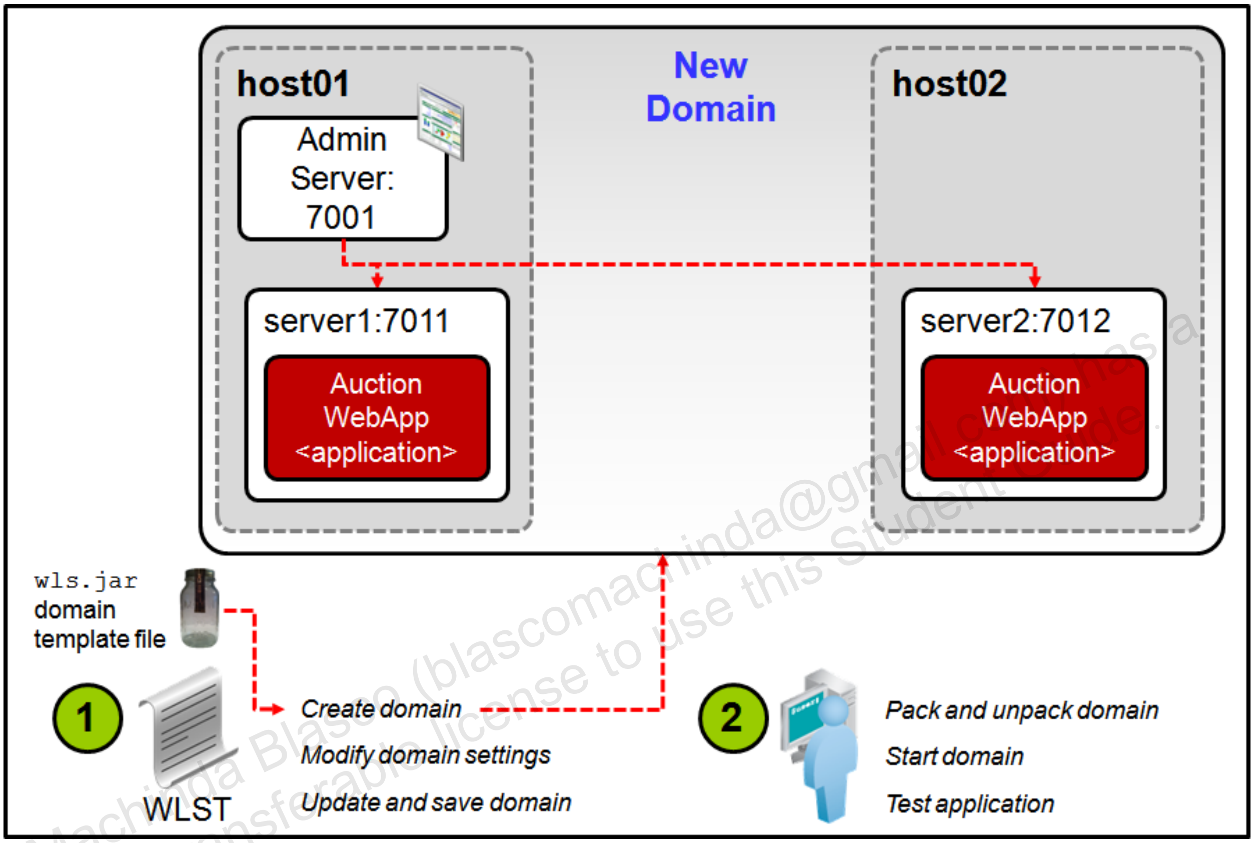
Practices Overview

In the practices for this lesson, you learn how to use WLST to create, modify, and monitor WebLogic domains.

Practice 6-1: Creating and Modifying a Domain with WLST

Overview

This practice shows you how to quickly write a script that creates a WebLogic domain using a standard template. Then it shows you how to modify the configuration by adding servers, a cluster, assigning servers to the cluster, and deploying an application to the cluster.



This practice involves the following:

1. Using WLST to create a domain from a template, modifying the domain, and saving the configuration
2. Using the `pack` and `unpack` commands to set up the configuration for managed servers on host02, starting the domain, and testing the application to verify that everything works properly.

Tasks

1. Connect to the **host01** and **host02** machines.
2. Set up the practice environment.
   a. You may reuse the MAIN terminal windows from previous practices. Otherwise, open a terminal window **on each machine** by clicking the terminal icon in the launch panel located on the top of the screen.
   b. Set the title of each terminal window by selecting Terminal > Set Title and entering `MAIN` as the title. This makes it easier to distinguish the purpose of each window.
   c. Within the terminal window **on host01**, navigate to the `practice06-01` folder and execute the `setup.sh` script to reset the initial practice environment:

   ```
   $ cd /practices/part2/practice06-01
   $ ./setup.sh
   ```

   This script performs the following tasks:

   – Ensures that no previous servers are running on both machines

   – Restores any domains and the practice to their original state
3. Copy the starting WLST script.

   WebLogic includes sample scripts with the installation. You copy the appropriate script that matches what you are trying to do so you do not have to start from the beginning.
   a. Perform the following commands to copy the starting script to your practice folder (Do not forget the (`.`) on the end of the command):

   ```
   $ cp $WL_HOME/common/templates/scripts/wlst/basicWLSDomain.py .
   ```
4. Open the script file for editing.
   a. Remove JMS and JDBC code from the script. Find the following lines in the code and delete them. Feel free to review them before deleting them.

   ```
   #=======================================================
   # Create a JMS Server.
   #=======================================================
   cd('/')
   create('myJMSServer', 'JMSServer')

   #=======================================================
   # Create a JMS System resource.
   #=======================================================
   cd('/')
   create('myJmsSystemResource', 'JMSSystemResource')
   cd('JMSSystemResource/myJmsSystemResource/JmsResource/NO_NAME_0'
   )

   #=======================================================
   # Create a JMS Queue and its subdeployment.
   #=======================================================
   myq=create('myQueue','Queue')
   myq.setJNDIName('jms/myqueue')
   ```

```
myq.setSubDeploymentName('myQueueSubDeployment')
cd('/')
cd('JMSSystemResource/myJmsSystemResource')
create('myQueueSubDeployment', 'SubDeployment')

#=========================================================
# Create and configure a JDBC Data Source, and sets the JDBC
user.
#=========================================================
cd('/')
create('myDataSource', 'JDBCSystemResource')
cd('JDBCSystemResource/myDataSource/JdbcResource/myDataSource')
create('myJdbcDriverParams','JDBCDriverParams')
cd('JDBCDriverParams/NO_NAME_0')
set('DriverName','org.apache.derby.jdbc.ClientDriver')
set('URL','jdbc:derby://localhost:1527/db;create=true')
set('PasswordEncrypted', 'PBPUBLIC')
set('UseXADataSourceInterface', 'false')
create('myProps','Properties')
cd('Properties/NO_NAME_0')
create('user', 'Property')
cd('Property/user')
cmo.setValue('PBPUBLIC')
cd('/JDBCSystemResource/myDataSource/JdbcResource/myDataSource')
create('myJdbcDataSourceParams','JDBCDataSourceParams')
cd('JDBCDataSourceParams/NO_NAME_0')
set('JNDIName', java.lang.String("myDataSource_jndi"))
cd('/JDBCSystemResource/myDataSource/JdbcResource/myDataSource')
create('myJdbcConnectionPoolParams','JDBCConnectionPoolParams')
cd('JDBCConnectionPoolParams/NO_NAME_0')
set('TestTableName','SYSTABLES')

#=============================  ============================
# Target resources to the servers.
#=========================================================
cd('/')
assign('JMSServer', 'myJMSServer', 'Target', 'AdminServer')
assign('JMSSystemResource.SubDeployment',
'myJmsSystemResource.myQueueSubDeployment', 'Target',
'myJMSServer')
assign('JDBCSystemResource', 'myDataSource', 'Target',
'AdminServer')
```

b. Add the administrative password. Find the following line in the code: For consistency use the password from the Credentials section.

```
# Please set password here before using this script, e.g.
cmo.setPassword('value')
```

And add the following line below it:

```
cd(/)
cd('Security/base_domain/User/weblogic')
cmo.setPassword('Enter Password Here')
```

c. Add two managed servers. After the code you just wrote to set the administrative password, add the following code that creates `server1` on host01, and `server2` on host02:

```
# Creating Managed Servers
cd('/')
create('server1', 'Server')
cd('Server/server1')
set('ListenPort', 7011)
set('ListenAddress', 'host01')

cd('/')
create('server2', 'Server')
cd('Server/server2')
set('ListenPort', 7012)
set('ListenAddress', 'host02')
```

d. Add a cluster and assign the managed servers to it. After the code you just added to create managed servers, add the following code that creates a cluster named `cluster1` and assigns the two servers to it:

```
# Create a cluster and assign the managed servers to that
cluster.
cd('/')
create('cluster1', 'Cluster')
assign('Server', 'server1,server2','Cluster','cluster1')
```

e. Deploy the `SimpleAuctionWebApp` application. After the cluster creation code, add the following code to deploy an application to `cluster1`:

```
# Deploy application
cd('/')
myApp=create('SimpleAuctionWebApp', 'AppDeployment')
myApp.setSourcePath('/practices/part2/apps/SimpleAuctionWebApp.w
ar')
assign('AppDeployment', 'SimpleAuctionWebApp', 'Target',
'cluster1')
```

f. Set the domain name and location. Find the line in the code that uses the `writeDomain` command and change the domain path and name so the command looks as follows:

```
writeDomain('/u01/domains/part2/SimpleAuctionDomain')
```

g. Save the script.

5. Run the script to create the domain with your modified settings.
   a. Execute the following command to run the script. It may take a few minutes for the script to finish.

```
$ ./wlst.sh basicWLSDomain.py
Initializing WebLogic Scripting Tool (WLST) ...
Welcome to WebLogic Server Administration Scripting Shell
Type help() for help on available commands
Exiting WebLogic Scripting Tool.
```

   b. Verify that your domain was created:

```
$ cd /u01/domains/part2
$ ls -l
drwxr-x--- 11 oracle oinstall  4096 Feb 18 09:08
SimpleAuctionDomain
```

6. Perform post-creation tasks:
   a. Create the jar file for remote managed servers using the `pack` command **on host01**:

```
$ cd /u01/app/fmw/wlserver/common/bin
$ ./pack.sh -domain=/u01/domains/part2/SimpleAuctionDomain -
template=/practices/part2/practice06-01/SimpleDomainManaged.jar
-template_name="My Domain Template" -managed=true
```

   b. Verify that your practice folder contains the `SimpleDomainManaged.jar` file.
   c. Create the remote managed server domain side using the `unpack` command **on host02**:

      Note: Again, recall that `/practices` is a shared file system between host01 and host02.

```
$ cd /u01/app/fmw/wlserver/common/bin
$ ./unpack.sh -domain=/u01/domains/part2/SimpleAuctionDomain -
template=/practices/part2/practice06-01/SimpleDomainManaged.jar
-app_dir=/u01/domains/part2/SimpleAuctionDomain/apps
```

   d. Verify that the `SimpleAuctionDomain` domain is now present **on host02**:

```
/u01/domains/part2/SimpleAuctionDomain
```

7. Start your new domain **(execute on host01)**.
   a. Execute the following commands to start the AdminServer:

      Note: These scripts are separately located from the scripts used to start the wlsadmin domain.

```
$ cd /practices/part2/practice06-01
$ ./startAdmin.sh
```

   b. Wait for the server to finish starting.
   c. Execute the following commands to start server1 and server2:

      Note: Login to each server using the WebLogic Server credentials.

```
$ ./startServer1.sh
$ ./startServer2.sh
```

   d. Wait for all servers to finish starting.

8. Verify domain settings using the WebLogic administration console:
   a. Log in to the administration console and check the following settings.

      `http://host01:7001/console`

      – Verify that the domain name is SimpleAuctionDomain.

      – Select Environment > Servers and verify that all three servers are running.

      – Verify that each managed server is part of `cluster1`.

      – Select Deployments and verify that the `SimpleAuctionWebApp` application is deployed (Active) and targeted to `cluster1`.

9. Test the Auction application.
   a. Direct your Web browser to the following URL:

      `http://host01:7011/SimpleAuctionWebApp`

      If you see the main application landing page then congratulations, you have successfully created and modified a WebLogic domain from the beginning using WLST.

10. **IMPORTANT** Instruction: Leave servers running for next practice.

    The domain you just created is used for the next practice in this lesson. Leave the servers on both machines running. However, if you are not going to perform the next practice, then you can optionally perform the next step to clean up the practice folder.

11. Run the clean up script **(optional). YOU SHOULD NOT PERFORM THIS STEP IF YOU ARE DOING PRACTICE 6-2**.
    a. Navigate to the current practice folder **on host01** and execute the following script to clean up the practice environment. This script resets the practice to its original state, so only execute it if you want to undo your configuration.

       ```
       $ ./reset.sh
       ```

       The script performs the following:

       – Kills all **`java`** processes on both machines

       – Deletes **`WLST`** scripts from the practice folder

       – Deletes the **`SimpleDomainManaged.jar`** file from the practice folder

       – Deletes **`SimpleAuctionDomain`** on both machines

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Open a new terminal window **on host01**:
2. Change the current directory to the current practice folder.
3. Execute the solution script:

   ```
   $ ./solution.sh
   ```

   The solution script performs the following:

   - Cleans up the practice to the starting point
   - Copies solution WLST scripts to the practice folder
   - Creates the `SimpleAuctionDomain` domain using the solution script
   - Executes `pack` to create the `SimpleDomainManaged.jar` managed server file
   - Executes `unpack` on the host02 machine to set up domain files
   - Starts the `AdminServer`, `server1`, and `server2` servers in their own terminal windows. It uses a `boot.properties` file for `server1` and `server2` to start.
4. Wait for servers **on host01 and host02** to fully start.
5. Continue starting at step number 8.

   Note: Only `practice06-02` depends on this practice so the solution is not required to work on other practices.

Practice 6-2: Monitoring a Domain with WLST

Overview

This practice builds on practice06-01 by capturing some runtime metrics associated with the SimpleAuctionWebApp running on the server. You will begin with a partial script and add code to it to monitor the session count and number of requests of each server that hosts the SimpleAuctionWebApp application. Next, you execute the script, which runs in a loop displaying the statistics of the running servers. Then you run the SimpleAuctionWebApp application to change the statistics and see them in your monitoring script display.

This practice shows you the best practice for using WLST to get domain-wide runtime statistics. There are two ways to capture runtime statistics per server, either by connecting to each server individually and capturing the ServerRuntime data on each server, or by connecting to the AdminServer and using the DomainRuntime MBean to capture the ServerRuntime statistics for any and all servers in the domain. The best practice is to use the DomainRuntime MBean to capture all statistics for the domain.

Tasks

1. Connect to the **host01** and **host02** machines.
2. Set up the practice environment.
   a. You may reuse the MAIN terminal windows from previous practices. Otherwise, open a terminal window **on each machine** by clicking the terminal icon in the launch panel located on the top of the screen.
   b. Set the title of each terminal window by selecting Terminal > Set Title and entering MAIN as the title. This makes it easier to distinguish the purpose of each window.
   c. If you completed practice06-01 and the servers are all still running, then skip the next step.
   d. If the servers for practice06-01 are not running, then run the solution for practice06-01 to set the environment up for this practice. Follow the instructions for the practice solution for practice06-01, and then continue with the next step after all the servers are running.
3. Copy the partial starting script for monitoring the domain.
   a. Perform the following commands to copy the practice script to the practice folder:

```
$ cd /practices/part2/practice06-02
$ cp resources/monitorapp.py .
```

4. Modify script to monitor the SimpleAuctionWebApp application.
   a. Open the script for editing:

```
$ gedit monitorapp.py
```

b. Review the variables at the beginning of the file:

Variable	Value	Description
`url`	`host01:7001`	The URL of the administration server
`username`	***See credentials***	The administrative user of the domain
`password`	***See credentials***	The password of the administrative user
`appName`	`SimpleAuctionWebApp`	The name of the application used in this practice
`appWebRoot`	`SimpleAuctionWebApp`	The web context root of the application used in this practice
`wmName`	`default`	The name of the work manager used by the application in this practice

c. Review the script's connection code. The code must connect to the running server before it can access runtime information. See that the parameters passed in to the `connect()` command are the administrative username and password, and the URL of the `AdminServer`.

```
try:
  #Connect to the Admin Server
  connect(username, password, url)
except:
  print 'Server not available.'
  exit()
```

d. Review the script's outer loop code. The script performs two different loops to capture and display metrics. The first loop is a `while` loop. This loop runs forever and is used to repeat the process of capturing the live runtime data from the domain and displaying it on the screen. This loop uses a 15 second `sleep` between iterations and executes all of its commands within a `try` block. This code currently contains `TODO` comments that you will replace with WLST commands to perform the tasks described in them.

```
# Loop indefinitely
while 1:
  sessions = '-'
  invokeCount = '-'

  # Connect to each managed server's runtime MBeans
  # and retrieve session and request counts via the main
  # domain's domainRuntime MBean
  try:
    #TODO: Switch to the domainRuntime tree
    #TODO: Navigate to the ServerRuntimes MBean
    #TODO: Create a variable and store a list of
    #      server runtimes in it
    rc=os.system('clear')
    print ''
    print '--------------------------------------------'
    print 'Server\tSessions\tRequests'

    #Loop through the servers
    ...
      print server.getName() + '\t' + sessions +
                               '\t\t' + invokeCount

    print '--------------------------------------------'
    jythontime.sleep(15)
  except Exception, e:
    print 'Exception: ' + e
```

e. Review the script's inner loop code. The second loop is a `for` loop. This loop runs for each `ServerRuntime` MBean returned by the domain, and iterates through all of the `ServerRuntime` MBeans of the domain to capture the metrics to display. This code currently contains `TODO` comments that you will replace with WLST commands to perform the tasks described in them.

IMPORTANT NOTE: Use spaces, not tabs, to indent your code. WLST is very picky when it comes to whitespace because indentation is used to delimit loops and other control structures.

```
#Loop through the servers
for server in servers:
  #TODO: Skip AdminServer because the web app
  #      is not deployed there
  #TODO: Navigate to the server runtime of the
  #      current server entry represented by your variable
  #TODO: Get the MBean associated with the runtime
  #      of the SimpleAuctionWebApp application
  #TODO: Use the SimpleAuctionWebApp MBean to get
  #      the current open session count for the server
  #TODO: Get the MBean for the default work
  #      manager used by the application and store it in a var
  #TODO: Use the work manager MBean to get the
  #      current number of completed requests on the server
  print server.getName() + '\t' + sessions +
                              '\t\t' + invokeCount
```

f. Find the `#TODO: Switch to the domainRuntime tree` comment in the code and replace it with the code to navigate to the `domainRuntime` tree:

```
domainRuntime()
```

g. Find the `#TODO: Navigate to the ServerRuntimes MBean` comment in the code and replace it with the code to change to the MBean "folder" that represents the `ServerRuntimes` service that contains all the `ServerRuntimes` of the domain.

```
cd('/ServerRuntimes')
```

h. Find the `#TODO: Create a variable and store a list of server runtimes in it` comment in the code and replace it with the code that uses the `DomainRuntimeService` to get the list of available `ServerRuntime` MBeans from the domain. Store the result in a variable called `servers`.

```
servers=domainRuntimeService.getServerRuntimes()
```

i. Find the `#TODO: Skip AdminServer because the web app is not deployed there` comment in the code and replace it with code that determines whether the current server for this iteration is the `AdminServer` or not. If it is the `AdminServer`, then go to the next iteration in the loop because the application is not deployed to the `AdminServer` to avoid the code raising an exception.

```
if server.getName() == 'AdminServer':
  continue
```

j. Find the `#TODO: Navigate to the server runtime of the current server entry represented by your variable` comment in the code and replace it with the code that navigates to the MBean that represents the current server. All of the server's associated MBeans, operations, and attributes are available to your code from this MBean. You use the `getName()` method to get the name of the server MBean because the hierarchy is structured to use the name of the server for this location in the tree.

```
cd('/ServerRuntimes/' + server.getName())
```

k. Find the `#TODO: Get the MBean associated with the runtime of the SimpleAuctionWebApp application` comment in the code and replace it with the code that returns the MBean that represents the runtime information associated with the running `SimpleAuctionWebApp` on this particular server. Store the result in a variable called `webModule`. The `getMBean()` method is used to return the object to use for subsequent commands. Instead of traversing the MBean tree, the code passes the known path to the command to preserve the current location in the tree. Some aspects of the path comprise real-time data, so variables are used to get the correct path to the required MBean object.

```
webModule = getMBean('ApplicationRuntimes/' + appName +
'/ComponentRuntimes/' + server.getName() + '_/' + appWebRoot)
```

l. Find the `#TODO: Use the SimpleAuctionWebApp MBean to get the current open session count for the server` comment in the code and replace it with the code that uses the `webModule` variable from the last command to call the represented MBean's `getOpenSessionsCurrentCount()` method. Store the result in a variable called `sessions`. Remember that this only returns the session count for the `SimpleAuctionWebApp` application on this server. Also note that the code casts the return to a string value.

```
sessions = str(webModule.getOpenSessionsCurrentCount())
```

m. Find the `#TODO: Get the MBean for the default work manager used by the application and store it in a var` comment in the code and replace it with the code to get the MBean object associated with the work manager for the `SimpleAuctionWebApp` application. Store the result in a variable called `appWM`. Again, this code uses the `getMBean()` method, referencing the relative path from the current location in the tree to the MBean for the `default` work manager. And again, the path comprises elements that are based on real-time data, so variables are used to create the path.

```
appWM = getMBean('ApplicationRuntimes/' + appName +
'/WorkManagerRuntimes/' + wmName)
```

n. Find the `#TODO: Use the work manager MBean to get the current number of completed requests on the server` comment in the code and replace it with the code that uses the `appWM` variable from the last command to call the represented MBean's `getCompletedRequests()` method. Store the result in a variable called `invokeCount`. Remember that this result only represents the data for this particular server.

```
invokeCount = str(appWM.getCompletedRequests())
```

o. Review the printout of server metric data. Find the code that prints the captured data to the screen. It uses the variables defined along the way to display the data for each iteration of the `for` loop.

```
print server.getName() + '\t' + sessions + '\t\t' + invokeCount
```

5. Run the application and test the monitoring script.

a. Execute the following command to run the script to monitor server statistics:
Note the leading **'.'** before the **wlst.sh**.

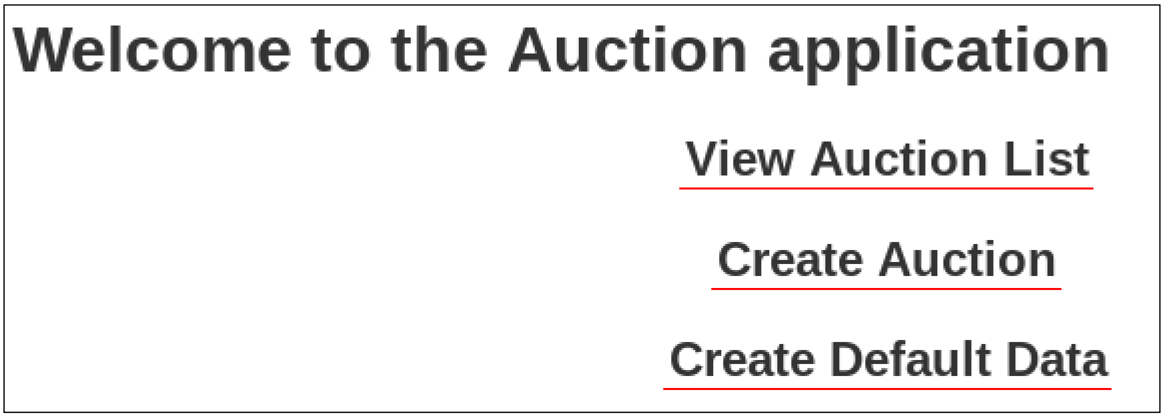
```
$ ./wlst.sh monitorapp.py
```

After a few seconds, the screen should clear and display the statistics it captures from the domain. If you have not run the application yet, your display should have zero values for everything. It is ok if there are some requests too. The script sleeps for 15 seconds, clears the screen, and then retrieves and displays the data again. Leave the script running in its own window.

```
---------------------------------------------
Server    Sessions   Requests
server1   0          0
server2   0          0
---------------------------------------------
```

b. Direct your web browser to the following URL to drive some server traffic:
`http://host01:7011/SimpleAuctionWebApp`

c. Click the *Create Default Data* link to populate the database with data for the Auction application. Follow this by clicking the confirmation to create the data.



Welcome to the Auction application

View Auction List

Create Auction

Create Default Data

d. Click the *Go Home* link and then click the *View Auction List* link to view the list of auctions stored in the database. Click through several links just to drive more traffic.

e. View the display of your running script. The numbers should have gone up for `server1`.

```
-------------------------------------------
Server     Sessions   Requests
server1    1          21
server2    0          0
-------------------------------------------
```

f. Try running the application using the `server2` address of `host02:7012` to drive more traffic on `server2`.

g. View the display of your running script again. The numbers should now reflect statistics for `server2`.

```
-------------------------------------------
Server     Sessions   Requests
server1    1          21
server2    1          10
-------------------------------------------
```

If you see the statistics in your script display then congratulations, you have successfully written a WLST domain monitoring script.

6. Shut down the environment.

   a. Press **Ctrl + C** in all terminal windows that are running servers.

   b. Close all terminal windows used for each server.

   c. You should still have MAIN terminal windows open on both machines.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Perform the practice solution for `practice06-01`.
2. Open a new terminal window **on host01**:
3. Change the current directory to the current practice folder.
4. Execute the following command to copy the solution script to the practice folder:
   ```
   $ cp solution/monitorapp.py .
   ```
5. Continue starting at step number 5.

Practices for Lesson 7: WebLogic Server and REST

Chapter 7

Practices for Lesson 7: Overview

Practices Overview

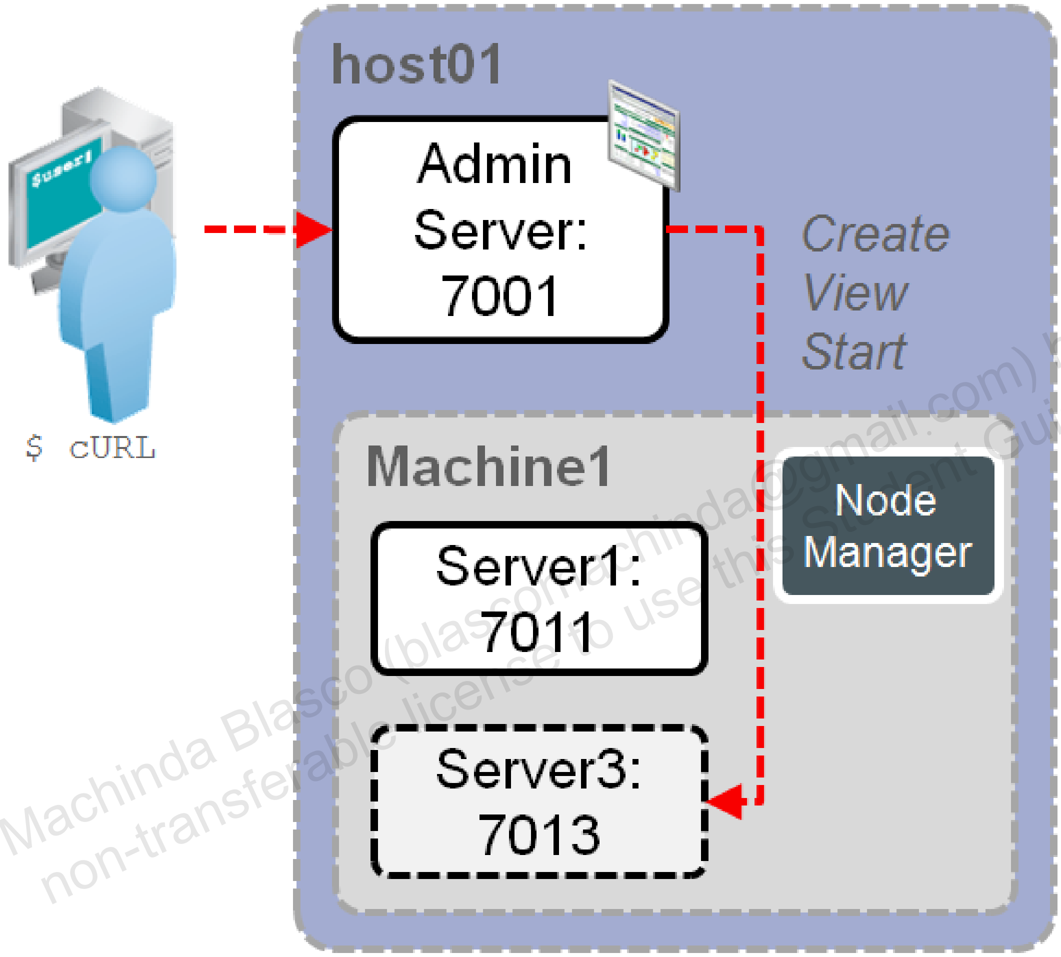
WebLogic Server provides a number of important features, one of which is support for Representational State Transfer (REST). Using WebLogic REST, languages other than Java, such as Python, can access WebLogic Server Domains and domain elements, creating, managing, monitoring and otherwise interacting with a domain and its elements.

In the practices for this lesson, you learn how to use REST to create, view, and manage WebLogic domain resources.

Practice 7-1: Creating and Managing Servers Using REST

Overview

This practice shows you how to use cURL to interact with a running WebLogic Server domain to list, create, start, stop and otherwise manage WebLogic Server server instances.



This practice involves the following:

1. Confirming that a domain is configured to support REST.
2. Creating a server in a known domain using REST and JSON
3. Starting a server using REST.
4. Listing various information about servers using REST.

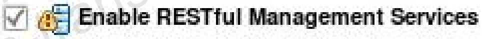
Tasks

1. Connect to the **host01** and **host02** machines.
2. Set up the practice environment.
   a. You may reuse the MAIN terminal windows from previous practices. Otherwise, open a terminal window **on each machine** by clicking the terminal icon in the launch panel located on the top of the screen.
   b. Set the title of each terminal window by selecting Terminal > Set Title and entering `MAIN` as the title. This makes it easier to distinguish the purpose of each window.
   c. Within the terminal window **on host01**, navigate to the `practice07-01` folder and execute the `setup.sh` script to reset the initial practice environment:

      ```
      $ cd /practices/part2/practice07-01
      $ ./setup.sh
      ```

      This script performs the following tasks:

      – Ensures that no previous servers are running on both machines
      – Restores any domains and the practice to their original state
      – Starts nodemanager, an administration server, and a server instance
      –
3. Validate that the wlsadmin domain is enabled for REST.
   a. Launch a web browser and log in to the Administration Console:
      `http://host01:7001/console`
   b. In the Domain Structure panel, select **wlsadmin**.
   c. In the right pane, select **Configuration > General** if not already selected.
   d. Scroll to the bottom of the page and click **Advanced**.
   e. Scroll approximately 2/3 of the way down the Advanced section and find **Enable RESTFul Management Services**.
   f. Confirm that RESTFul management services are enabled, which should resemble:

      ☑ Enable RESTful Management Services
   g. If RESTful Management services are not enabled:
      1) Click Lock and Edit
      2) Select Enable RESTFul Management Services
      3) Click **Save**
      4) Click **Activate Changes**
      5) Restart the administration server

4. Validate RESTFul Web Services are running correctly.
   a. In the browser, open a second tab.
   b. In the new tab, enter the URL **host01:7001/management/tenant-monitoring/servers.**
   c. When prompted, provide the WebLogic Server user and password from the credentials section.
   d. Examine the generated page.
      How many servers are defined?

e. Select a server name from the list and add it to the end of the URL. For example: **host01:7001/management/tenant-monitoring/servers/server1**

f. **What can you tell from the result? Can you answer the questions below?**

What Java version is the server using?	
What version of WebLogic Server is running?	

5. Examine existing servers using cURL.

   a. Return to the HOST01 command prompt.
   Remember this window should be open to the practice07-01 directory.

   b. In the command window, enter a curl command to list all known servers using the *tenant-monitoring/servers* relative path. The command should resemble:

```
$ curl --user weblogicUser  -H Accept:application/json
-X GET http://localhost:7001/management/tenant-
monitoring/servers
```

   Replace `webLogicUser` with the username for the WebLogic Administration account. When prompted, enter the associated password.

   Note that the command should be entered as a single line.

   c. Examine the result, which should resemble that shown below (formatted for readability).

```
{"body":{
    "items":[
{"name":"AdminServer","state":"RUNNING","health":"HEALTH_OK"},
{"name":"server1","state":"RUNNING","health":"HEALTH_OK"},
{"name":"server2","state":"SHUTDOWN"}]},
"messages":[]
}
```

   How many servers are currently defined?

6. Add a server instance using cURL.
   For ease, a simple script is provided for creating users. This script is only partially complete.

   a. Open the script **createServer3.sh**, which contains a partial curl command to create a server.

```
$ gedit createServer3.sh
```

   b. Find **the –user webLogicUser:webLogicPassword** section and replace the username password with the WebLogic administrator password from the credentials section.

   c. Confirm that the URL uses the **weblogic/latest/edit/servers** relative path.

   d. The complete URL and POST command should resemble:
   -X POST http://host01:7001/management/weblogic/latest/edit/servers

e. The completed file should resemble (see Credentials for WebLogic Server):

```
curl  --user weblogic:webLogicPassword \
     -H X-Requested-By:MyClient \
     -H Content-Type:application/json \
      -d " `cat server3.options` " \
     -X POST
http://host01:7001/management/weblogic/latest/edit/servers
```

f. Save your changes and exit gedit.

g. Open the partially complete `server3.options` file. This file contains the required fields for adding a server and is partial complete.

```
$ gedit server3.options
```

h. Complete the server 3 definition by specifying the name as server3 and the listenPort as 7013. The completed file should resemble:

```
{
name: "server3",
listenPort:7013,
listenAddress: "",
defaultProtocol:"t3",
cluster: [ "clusters", "cluster1"],
machine: [ "machines", "machine1" ]
}
```

i. Examining the file, can you answer the question: What cluster and machine will the new server be part of?

j. Save your changes and exist gedit.

k. Execute the command:

```
$ ./createServer3.sh
```

7. Examine the server.

a. Using a command similar to the one below, examine the newly added server. The definitions of servers can be found in the relative path **weblogic/latest/domainConfig/servers**

```
$ curl --user weblogicUser:webLogicPassword \
     -H Accept:application/json \
     -X GET
http://localhost:7001/management/weblogic/latest/domainConfig/se
rvers/server3?links=none
```

Consider piping the result to less or a file for easier review.

Note the addition of the **`?links=none`** query argument. This stops WebLogic REST from displaying associated links.

8. Start the newly added server using cURL.

   Reminder: REST operations can be executed by using PORT requests against a specific object.

   The relative path **weblogic/latest/domainRuntime/serverLifeCycleRuntimes** includes all servers. Operations can be executed against servers by appending the server name and the operation to execute to the relative path.

   Start the newly added server by entering a cURL command, similar to:

```
curl --user weblogicUser:webLogicPassword \
     -H X-Requested-By:MyClient \
     -H Accept:application/json \
     -H Content-type:application/json \
     -d " {} " \
     -X POST
http://host01:7001/management/weblogic/latest/domainRuntime/serverL
ifeCycleRuntimes/server3/start
```

   For convenience, the partially completed script **startServer3.sh** is provided. This script will need to be updated with the correct username and password in order to be used, replacing *webLogicUser*:*webLogicPassword* with the correct credentials.

9. Confirm that the newly started server is running.
   List the state of the newly added server by entering a cURL command similar to:
   (Note you will need to use the credentials for WebLogic Server)

```
curl --user weblogic:webLogicPassword \
     -H Accept:application/json \
     -X GET
http://localhost:7001/management/weblogic/latest/domainRuntime/serv
erLifeCycleRuntimes/server3?links=none
```

   For convenience, the partially completed script `listServerState.sh` is provided. This script will need to be updated with the correct username and password in order to be used.

   In addition the script takes a server name as a parameter and can list the server state for any server.

10. Run cleanup script.
    a. Navigate to the current practice folder on `host01` and execute the following script to clean up the practice environment. This script resets the practice to its original state, so only execute it if you want to undo your configuration.

```
$ ./reset.sh
```

       The script performs the following:
       - Kills all `java` processes on both machines, including node manager processes
       - Resets the environment

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution. Note that this practice has no side effects and the solution can be ignored.

Solution Tasks

1. Open a new terminal window on **host01**.
2. Change the current directory to the current practice folder.
3. Configure the practice by running the setup script.
   ```
   $ ./setup.sh
   ```
4. Wait for administration and server 1 instances to fully start.
5. Change directory to the solution directory.
   ```
   $ cd solution
   ```
6. Execute the create server 3 script.
   ```
   $ ./createServer3.sh
   ```
7. Execute the start server 3 script.
   ```
   $ ./startServer3.sh
   ```
8. Execute the `listServerState` script with a parameter of server3.
   ```
   $ ./listServerState.sh server3
   ```
9. Execute the `stopServer3` script.
   ```
   $ ./stopServer3.sh
   ```
10. Reset the environment.
    ```
    $ cd ../
    $ ./reset.sh
    ```

Practices for Lesson 8: Secure Sockets Layer (SSL)

Chapter 8

Practices for Lesson 8: Overview

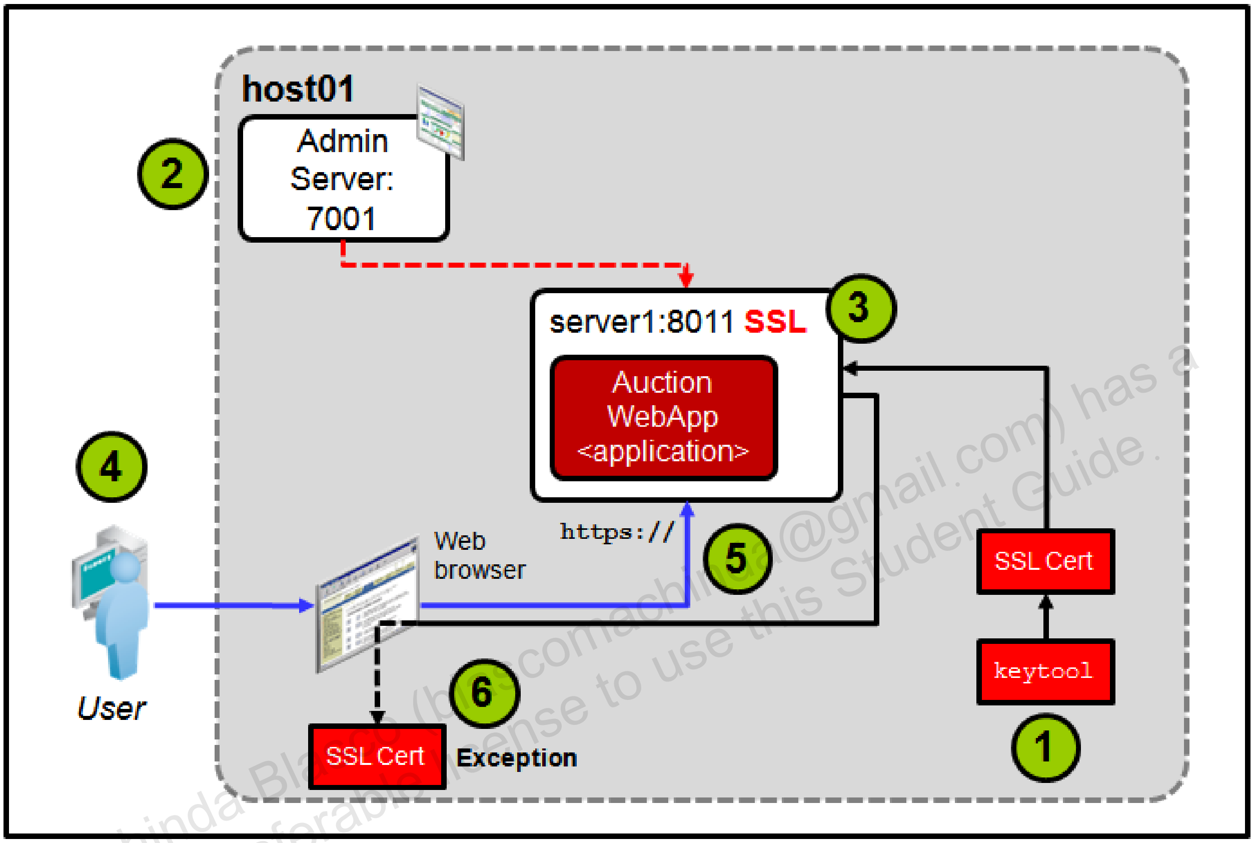
Practices Overview

Many applications need the security of communicating over the secure sockets layer (SSL) networking protocol. This provides secure communications between the server and the client, or between two servers. In this lab, you configure SSL and the keystores for the `server1` managed server in the `wlsadmin` domain.

Practice 8-1: Setting Up SSL

Overview

This practice shows you how to configure SSL certificates using `keytool`, and configuring WebLogic servers to use those certificates to establish secure SSL connections.



This practice involves the following:

1. Using `keytool` to generate an identity keystore that contains a private key and a self-signed public certificate
2. Configuring keystores in the administration console
3. Configuring SSL for a managed server
4. Using a web browser to access the application
5. The web browser uses the HTTPS protocol to access the server
6. The server returns its SSL certificate to the web browser and the user adds an SSL exception, thus allowing the connection

Tasks

1. Connect to the **host01** and **host02** machines.
2. Set up the practice environment.
   a. You may reuse the MAIN terminal windows from previous practices. Otherwise, open a terminal window **on host01** by clicking the terminal icon in the launch panel located on the top of the screen.
   b. Set the title of each terminal window by selecting Terminal > Set Title and entering `MAIN` as the title. This makes it easier to distinguish the purpose of each window.
   c. Within the terminal window **on host01**, navigate to the `practice08-01` folder and execute the `setup.sh` script to set up the initial practice environment:

   ```
   $ cd /practices/part2/practice08-01
   $ ./setup.sh
   ```

   This script performs the following:

   – Ensures that no previous servers are running on both machines
   – Restores the domain and practice to their original state
   – Deploys the practice application
   – Starts the `wlsadmin` domain on host01 (host02 is not used in this practice)
3. Create a new key pair using the Java `keytool` utility, copy the key to your domain folder, and configure `server1` to use your custom keystore.
   a. Perform the following command to run `keytool` to create a keystore and a key pair within the keystore (all in one line). You can use the `genkey.sh` script in this folder for convenience.

   ```
   $ keytool -genkey -v -alias wlskey -keyalg RSA -keysize 2048
     -sigalg MD5withRSA -dname "CN=wls-sysadm"
     -keypass wlskeypass -validity 365
     -keystore wls_identity.jks -storepass wlsstorepass
   ```

   b. Copy the key file you generated to your domain folder.

   ```
   $ cp wls_identity.jks /u01/domains/part2/wlsadmin
   ```

   c. Generate a Certificate Signing Request (CSR) using the key you have created. (You can use `certreq.sh` instead of entering the `keytool` command.)

   ```
   $ keytool -certreq -v -alias wlskey -file wls_cert_request.pem
    -keypass wlskeypass -storepass wlsstorepass
    -keystore wls_identity.jks
   ```

   d. Copy the CSR you generated to your domain folder.

   ```
   $ cp wls_cert_request.pem /u01/domains/part2/wlsadmin
   ```

   These copy steps are because you are more likely to back up your domains folder than you would be to back up the `practices` folder. Nothing on the local server uses this CSR `.pem` file. This `.pem` would be forwarded to your CA for a real production application.
   e. In the Administration Console, navigate to **Environment > Servers > server1 > Configuration > Keystores**.
   f. In Change Center, click Lock & Edit.
      1) On the Keystores page, click on **change** button.

2) Select “***Custom Identity and Java Standard Trust”*** option and click **Save**

3) Enter the following values and click **Save**.

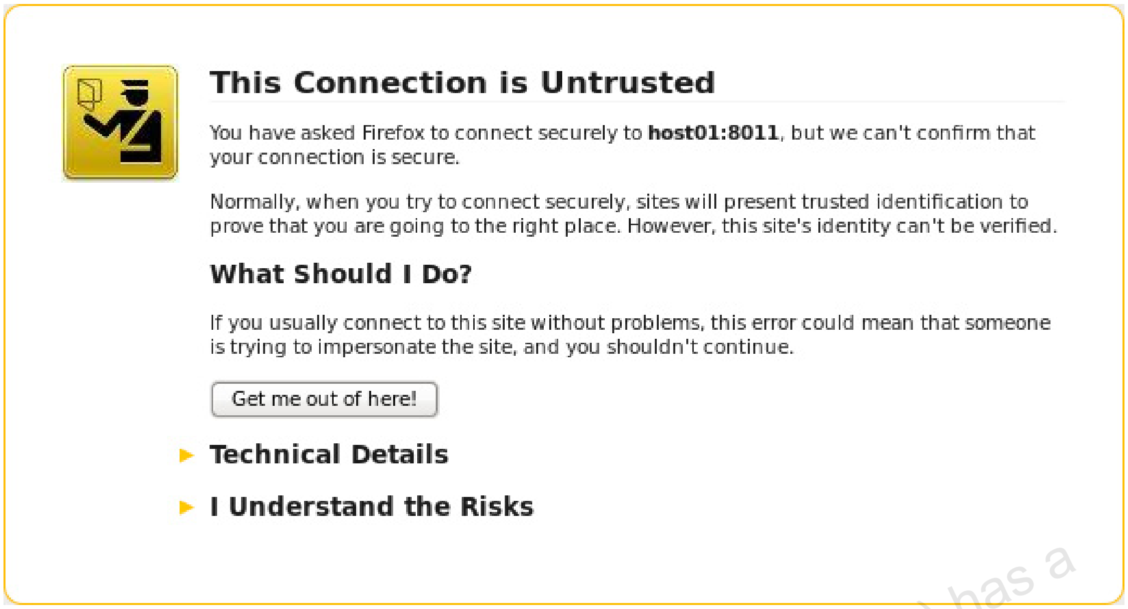
Description	Choices or Values
Keystores	Custom Identity and Java Standard Trust
Custom Identity Keystore	`wls_identity.jks`
Custom Identity Keystore Type	JKS
Custom Identity Keystore PassPhrase	`wlsstorepass`
Java Standard Trust Keystore PassPhrase	`changeit`

4. Configure `server1` to enable and support SSL using your custom identity keystore.
   a. In the Administration Console, navigate to **Environment > Servers > server1 > Configuration > SSL**.
   b. On the SSL page, specify the following properties and click **Save**.
      - Identity and Trust Locations: Keystores
      - Private Key Alias: `wlskey`
      - Private Key Passphrase: `wlskeypass`
   c. Navigate to **Environment > Servers > server1 > Configuration > General**.
   d. Select the check box next to **SSL Listen Port Enabled** and set the **SSL Listen Port** as **8011**.
   e. Click **Save**.
   f. Click **Activate Changes**.
   g. In the Domain Structure pane, expand **Environment > Servers**, select the **Control** tab and stop **server1** if it is running.
   h. Return to the terminal window on Host01 and start server1.

```
$ startServer1.sh
```

5. Test your SSL configuration.
   a. In another browser window or tab, access the URL:
      `https://host01:8011/SimpleAuctionWebApp`

      Note we are using `https`
      Note that browsing from the root machine (not host01 or host02) will cause errors.
      Use a browser on host01 to avoid errors.
   b. You may receive an error or warning.



This Connection is Untrusted

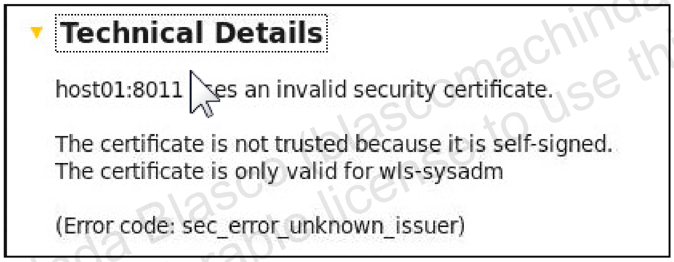
You have asked Firefox to connect securely to **host01:8011**, but we can't confirm that your connection is secure.

Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified.

What Should I Do?

If you usually connect to this site without problems, this error could mean that someone is trying to impersonate the site, and you shouldn't continue.

c. Click the *Technical Details* link to see why the warning is displayed. It will say that the certificate is not truly trusted because it is self-signed rather than being signed by a known certificate authority.



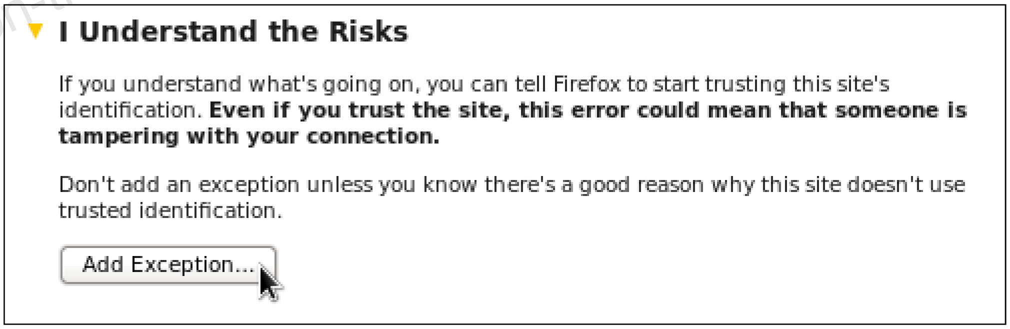
▼ Technical Details

host01:8011 uses an invalid security certificate.

The certificate is not trusted because it is self-signed.
The certificate is only valid for wls-sysadm

(Error code: sec_error_unknown_issuer)

d. Click the *I Understand the Risks* link to add an exception and click Add Exception (different web browsers may display this dialog differently):

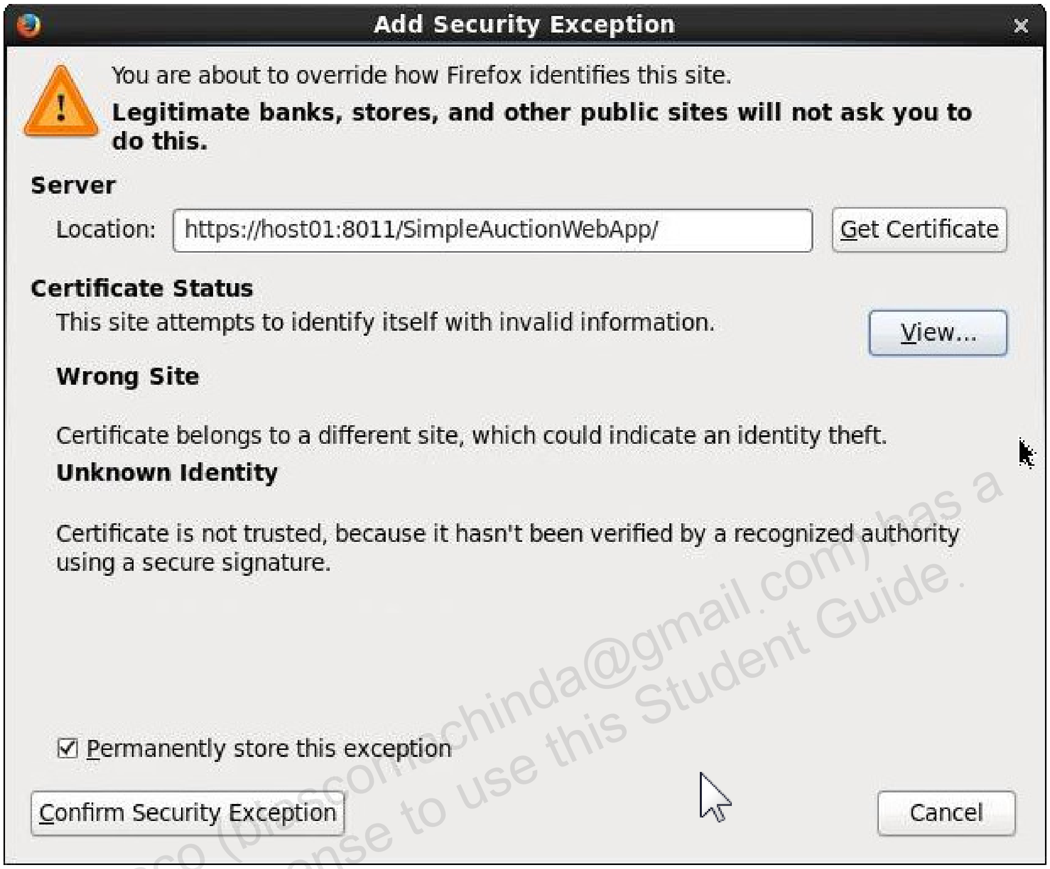


▼ I Understand the Risks

If you understand what's going on, you can tell Firefox to start trusting this site's identification. **Even if you trust the site, this error could mean that someone is tampering with your connection.**

Don't add an exception unless you know there's a good reason why this site doesn't use trusted identification.

e. Then click Get Certificate to add the server certificate to your browser. You can also click View to see details about the certificate.



Add Security Exception

You are about to override how Firefox identifies this site.

Legitimate banks, stores, and other public sites will not ask you to do this.

Server

Location: https://host01:8011/SimpleAuctionWebApp/

Certificate Status

This site attempts to identify itself with invalid information.

Wrong Site

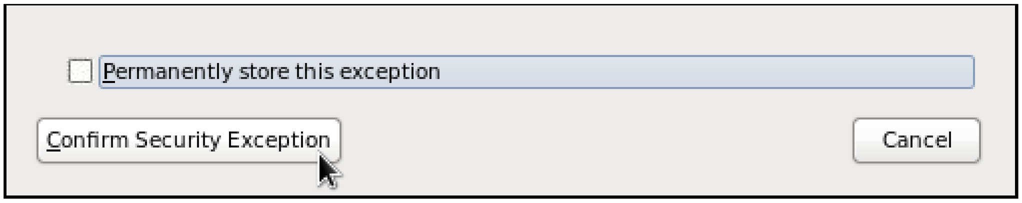
Certificate belongs to a different site, which could indicate an identity theft.

Unknown Identity

Certificate is not trusted, because it hasn't been verified by a recognized authority using a secure signature.

☑ Permanently store this exception

f. Deselect *Permanently store this exception* so that the browser continues to perform this security check for this certificate in the future. Click Confirm Security Exception. If you make this exception permanent by selecting the *Permanently store this exception* check box, then the browser will no longer validate the certificate for subsequent requests to this server.


☐ Permanently store this exception
Confirm Security Exception
Cancel

g. Now, you can access the application on `server1`.

If you see the main application landing page then congratulations, you have successfully created a custom security certificate and configured WebLogic SSL to use it.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Open a new terminal window **on host01**.
2. Change the current directory to the /practices/part2/practice08-01 folder.
3. Execute the solution script:

   ```
   $ ./solution.sh
   ```

   The solution script performs the following tasks:

   - Ensures that no previous servers are running on both machines
   - Restores the domain and practice to their original state
   - Executes `genkey.sh` to create the practice key pair
   - Executes `certreq.sh` to self-sign the certificate
   - Copies the solution `config.xml` with keystore and SSL configuration to the domain
   - Copies all `.jks` and `.pem` files to the domain folder
   - Deploys the practice application
   - Starts the `wlsadmin` domain on host01
4. Wait for servers **on host01** to fully start.
5. Continue starting at step number 5.

Practices for Lesson 9: Shared Java EE Libraries

Chapter 9

Practices for Lesson 9: Overview

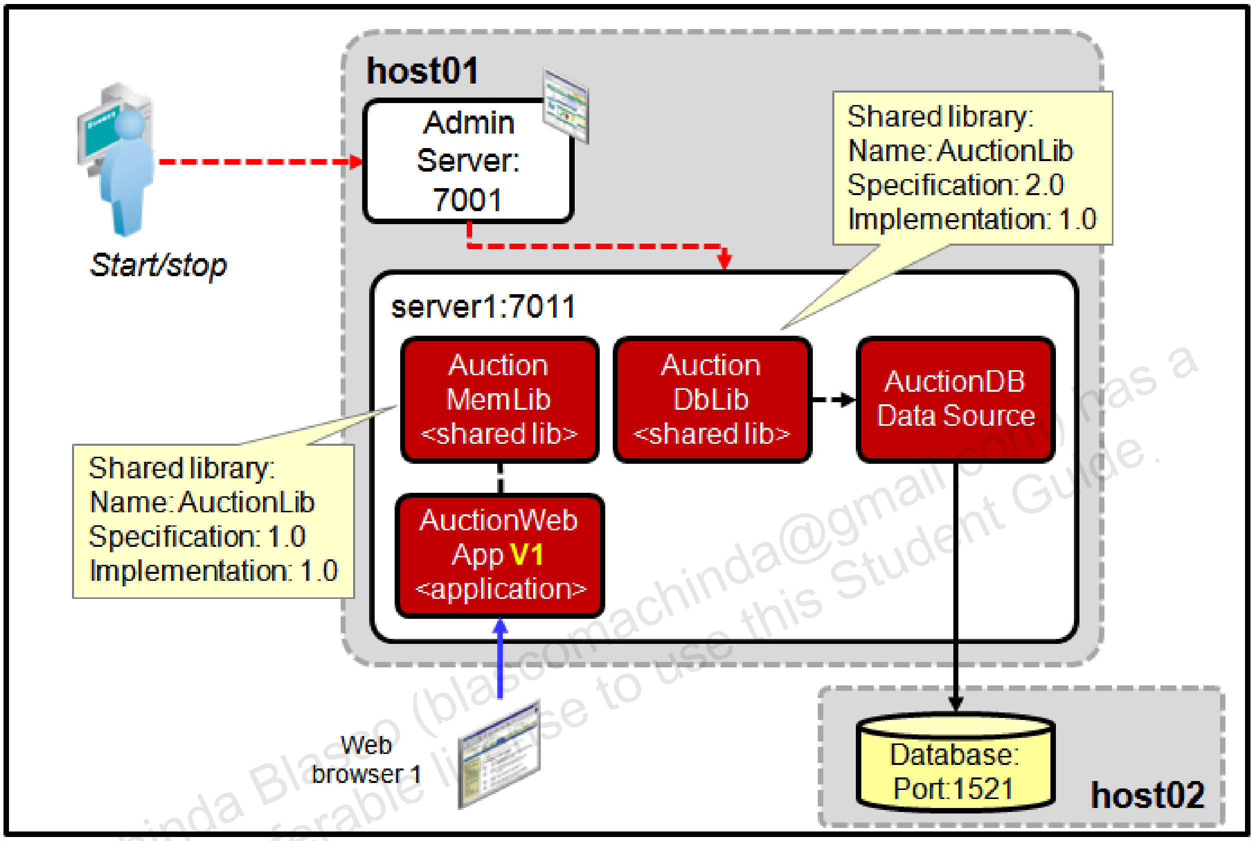
Practices Overview

In the practices for this lesson, you configure and deploy a shared Java EE library and an application that references it.

Practice 9-1: Configuring and Deploying a Shared Library

Overview

This practice shows you how to configure a shared library and a referencing application and how to deploy them so that they work together as a single deployment.


host01
Admin Server: 7001
Start/stop
Shared library: Name: AuctionLib Specification: 2.0 Implementation: 1.0
server1:7011
Auction MemLib <shared lib>
Auction DbLib <shared lib>
AuctionDB Data Source
Shared library: Name: AuctionLib Specification: 1.0 Implementation: 1.0
AuctionWeb App V1 <application>
Web browser 1
Database: Port:1521
host02

This image depicts the architecture of the environment for this practice:

1. You configure two shared libraries and deploy them:
   a. `AuctionMemLib`: A memory-based version of the library
   b. `AuctionDbLib`: A database version of the library
2. You configure and deploy the first release (V1) of the `AuctionWebApp` application, which references the memory-based version of the `AuctionLib` library, `AuctionMemLib`.
3. The `AuctionDbLib` library is deployed as part of this practice for two reasons:
   a. To demonstrate that you can control which library your application references
   b. To prepare for the production redeployment practice that you perform after this practice
4. You test the application and verify that it is configured and working correctly.

Tasks

1. Connect to the **host01** machine.
2. Set up the practice environment.
   a. You can reuse the MAIN terminal window from previous practices. Otherwise, open a terminal window **on host01** by clicking the terminal icon in the launch panel located at the top of the screen.
   b. Set the title of the terminal window by selecting Terminal > Set Title and entering `MAIN` as the title. This makes it easier to distinguish the purpose of each window.
   c. In the terminal window **on host01**, navigate to the `practice09-01` folder and execute the `setup.sh` script to set up the initial practice environment:

```
$ cd /practices/part2/practice09-01
$ ./setup.sh
```

   This script performs the following:

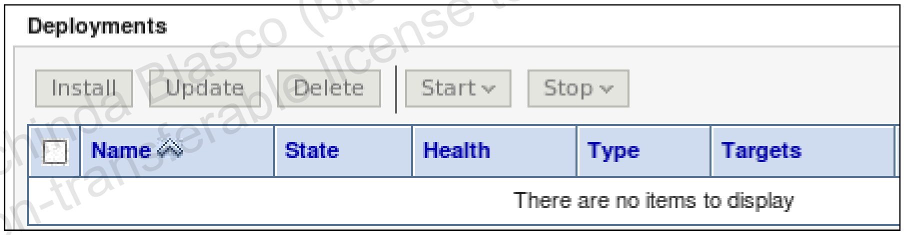
   – Ensures that no previous servers are running on both machines

   – Restores the domain and practice to their original state

   – Starts the `wlsadmin` domain on host01 (host02 is not used in this practice.)

   – Ensures that no applications or libraries are deployed to `server1`.

3. Verify the domain's configuration.
   a. Launch a web browser and log in to the administration console:
   `http://host01:7001/console`
   b. In the Domain Structure pane navigate to **wlsadmin > Deployments** and verify that no applications are deployed.



☐	Name	State	Health	Type	Targets
There are no items to display					

4. Configure the in-memory version of the library.

Note: You can sometimes mistakenly add `^M` carriage return characters to the end of your edited lines in the application manifest files. This may occur when copying and pasting from the course. If you do have issues when deploying your libraries or application, execute the `fixmanifest.sh` script to strip all `^M` characters from your files and try again. You can verify that ^M characters exist in manifests using the *hexdump –c filename* command.

Additionally, if you have any problems at all then you may have accidentally added some other type of non-readable character into the manifest file. If this happens, then you should copy the application's solution manifest file over yours and try deploying the application again to see if that works.

The practice folder contains the two libraries in exploded deployment form for easy modification. You will deploy them in exploded form as well.

   a. Perform the following commands to navigate to the in-memory library folder and open its manifest file for editing:

```
$ cd resources/AuctionMemLib/META-INF
$ gedit MANIFEST.MF
```

b. Ensure that the current file contains only the following entries:

```
Manifest-Version: 1.0
Class-Path:
```

c. Add the following entries to the end of the file to add shared library configuration for a library named `AuctionLib`, with a `specification-version` of `1.0` and an `implementation-version` of `1.0`:

```
Extension-Name: AuctionLib
Specification-Version: 1.0
Implementation-Version: 1.0
```

d. Save the file.

5. Configure the database version of the library.

   a. Perform the following commands to navigate to the database library folder and open its manifest file for editing:

```
$ cd ../../AuctionDbLib/META-INF
$ gedit MANIFEST.MF
```

   b. Ensure that the current file contains only the following entries:

```
Manifest-Version: 1.0
Class-Path:
```

   c. Add the following entries to the end of the file to add shared library configuration for a library named `AuctionLib`, with a `specification-version` of `2.0` and an `implementation-version` of `1.0`:

```
Extension-Name: AuctionLib
Specification-Version: 2.0
Implementation-Version: 1.0
```

   Note that the name of the library is the same for each library, but the `specification-version` is different. This allows referencing applications to reference the same library by name while simultaneously controlling which version of that library to use.

   d. Save the file.

6. Deploy the in-memory library by using the administration console.

   a. Browse to Deployments in the administration console.

   b. Click **Lock & Edit** to begin an edit session.

   c. Click **Install** to begin the deployment process.

   d. Use the wizard to browse to `/practices/part2/practice09-01/resources/AuctionMemLib` and click Next.

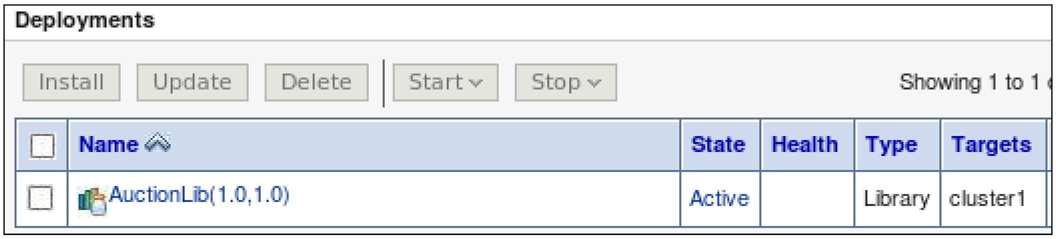
   e. Select ***Install this deployment as a library***, which should be selected by default because WebLogic detects the library configuration in the manifest file, and click Next.

   f. Select *All servers in the cluster*, which automatically selects `cluster1` as a target and click **Next**.

   g. Review the settings in the General section to verify that the deployment is named `AuctionLib` and displays the correct version information, and click **Finish**.

   h. Click **Activate Changes** to realize your configuration.

   i. Verify that your library is deployed as a library with the correct version information:



Deployments

□	Name	State	Health	Type	Targets
□	AuctionLib(1.0,1.0)	Active		Library	cluster1

Now you deploy the database version of the library.

7. Deploy the database library by using `weblogic.Deployer`.

   You should have successfully deployed the in-memory version of the library in the previous step by using the administration console. For the database version of the library, you use the `weblogic.Deployer` tool to achieve the same result.

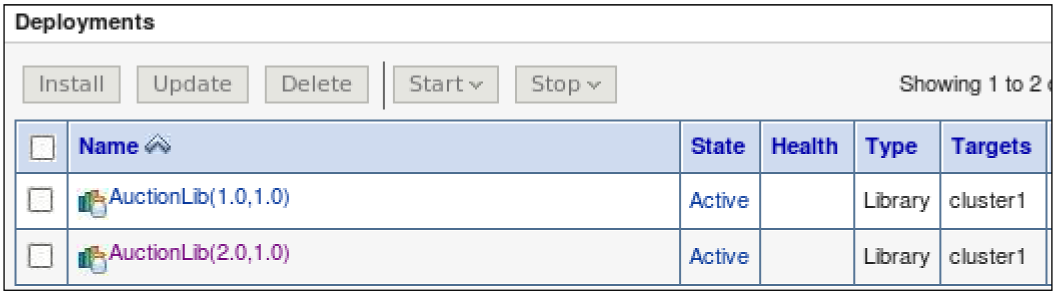
   a. Perform the following commands to deploy the database version of the library:
   Note: Use the WebLogic Server credentials for the WebLogic password below.

   ```
   $ java weblogic.Deployer -adminurl host01:7001 -username
   weblogic -password webLogicPassword -deploy -targets cluster1 -
   library /practices/part2/practice09-01/resources/AuctionDbLib
   ```

   Your output should resemble the following. You can ignore any exceptions relating to `server2` unavailability. This is because you are not using `server2` for this practice. The deployment for `server2` is deferred until the server is started and has also successfully deployed on `server1`.

   ```
   weblogic.Deployer invoked with options:  -adminurl host01:7001 -
   username weblogic -deploy -targets cluster1 -library
   /practices/part2/practice09-01/resources/AuctionDbLib
   <Mar 12, 2013 5:31:23 PM UTC> <Info> <J2EE Deployment SPI> <BEA-
   260121> <Initiating deploy operation for application, AuctionLib
   [archive: /practices/part2/practice09-
   01/resources/AuctionDbLib], to cluster1 .>
   Task 3 initiated: [Deployer:149117]deploy library AuctionLib
   [LibSpecVersion=2.0,LibImplVersion=1.0] on cluster1.
   Task 3 deferred: [Deployer:149117]deploy library AuctionLib
   [LibSpecVersion=2.0,LibImplVersion=1.0] on cluster1.
   Target state: deploy deferred on Cluster cluster1
   java.rmi.RemoteException: [Deployer:149145]Unable to contact
   "server2". Deployment is deferred until "server2" becomes
   available.
   ```

   b. Verify that your library is deployed as a library with the correct version information:



Deployments

□	Name	State	Health	Type	Targets
□	AuctionLib(1.0,1.0)	Active		Library	cluster1
□	AuctionLib(2.0,1.0)	Active		Library	cluster1

8. Configure an application to reference the in-memory library.

You should have successfully deployed the in-memory and database versions of the library in the previous steps. Now you configure an application to reference the in-memory version of the library. The database version of the library is used in the next practice.

a. Perform the following commands to navigate to the `AuctionWebApp` folder and open its deployment descriptor for editing:

```
$ cd /practices/part2/practice09-01/resources/AuctionWebApp/WEB-
INF
```

```
$ gedit weblogic.xml
```

b. Find the following lines in the file:

```
<wls:weblogic-version>12.2.1</wls:weblogic-version>
<wls:context-root>AuctionWebApp</wls:context-root>
```

c. Add the following XML code right after the lines to add a reference to the `AuctionLib` shared library, with a `specification-version` of `1.0` and an `implementation-version` of `1.0`. Note that this XML file uses an XML namespace, which is why each tag is prefixed with `wls:`. If you are not aware, this is the way that XML avoids naming conflicts of elements in different XML files and schemas.

```
<wls:library-ref>
  <wls:library-name>AuctionLib</wls:library-name>
  <wls:specification-version>1.0</wls:specification-version>
  <wls:implementation-version>1.0</wls:implementation-version>
  <wls:exact-match>true</wls:exact-match>
</wls:library-ref>
```

The `exact-match` tag ensures that the `AuctionWebApp` application references the in-memory version of the library. Otherwise, WebLogic could potentially make the application reference the latest version of the library, which is the database version.

d. Save the file.

9. Deploy the application.

a. Perform the following commands to deploy the `AuctionWebApp` application:
Note: Use the WebLogic Server credentials for the WebLogic password below.

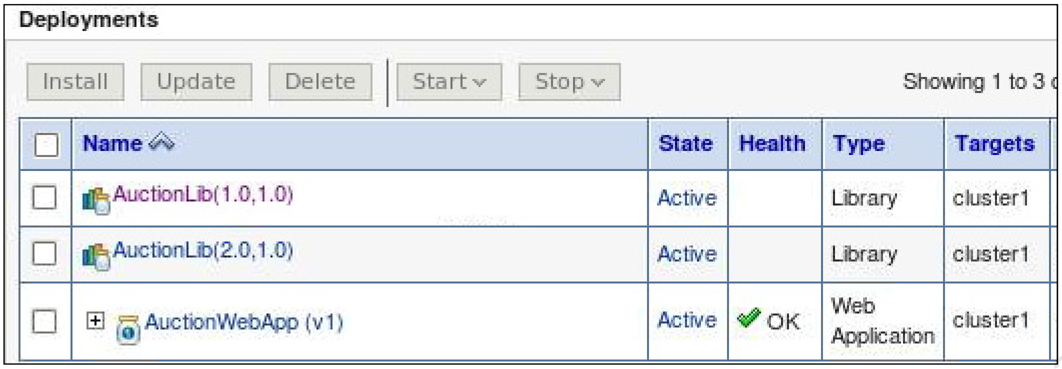
```
$ java weblogic.Deployer -adminurl host01:7001 -username
weblogic -password webLogicPassword -deploy -targets cluster1
/practices/part2/practice09-01/resources/AuctionWebApp
```

Your output should resemble the following. Again, you can ignore any exceptions relating to `server2` unavailability.

```
weblogic.Deployer invoked with options:  -adminurl host01:7001 -
username weblogic -deploy -targets cluster1
/practices/part2/practice09-01/resources/AuctionWebApp

<Mar 12, 2013 5:55:53 PM UTC> <Info> <J2EE Deployment SPI> <BEA-
260121> <Initiating deploy operation for application,
AuctionWebApp [archive: /practices/part2/practice09-
01/resources/AuctionWebApp], to cluster1 .>

Task 4 initiated: [Deployer:149026]deploy application
AuctionWebApp [Version=v1] on cluster1.

Task 4 deferred: [Deployer:149026]deploy application
AuctionWebApp [Version=v1] on cluster1.
```

```
Target state: deploy deferred on Cluster cluster1
java.rmi.RemoteException: [Deployer:149145]Unable to contact
"server2". Deployment is deferred until "server2" becomes
available.
```

b. Verify that your application is deployed:



Deployments

□	Name	State	Health	Type	Targets
□	AuctionLib(1.0,1.0)	Active		Library	cluster1
□	AuctionLib(2.0,1.0)	Active		Library	cluster1
□	⊞ AuctionWebApp (v1)	Active	✓ OK	Web Application	cluster1

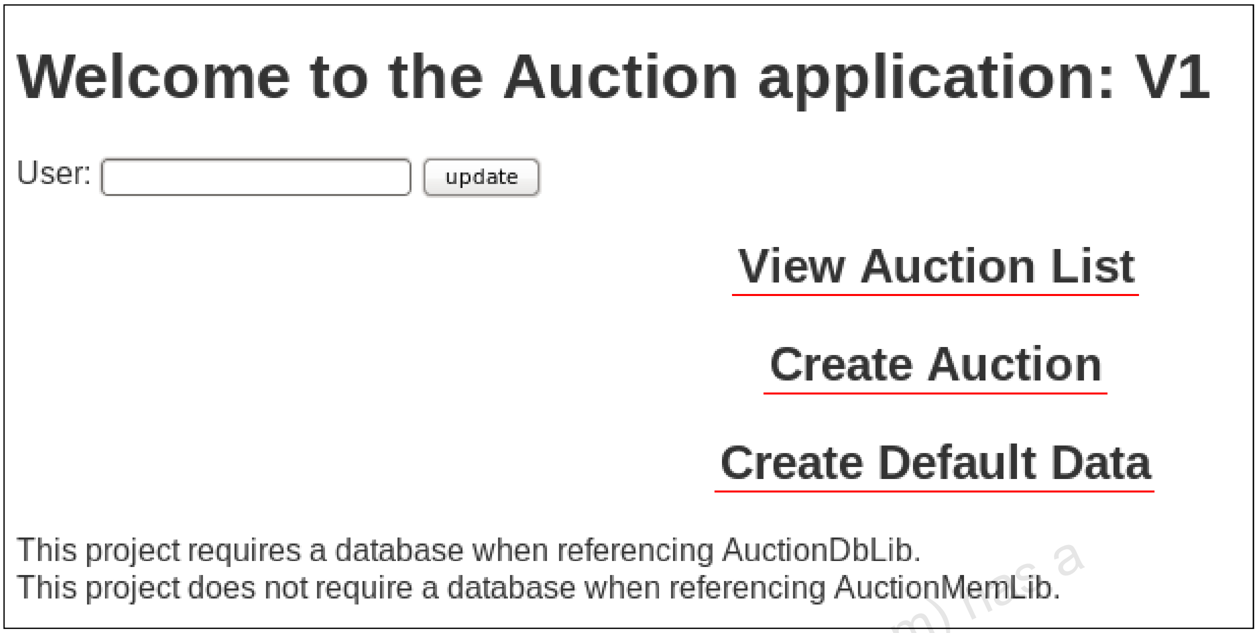
Note: The `AuctionWebApp` application has a (v1) notation next to it. This is because you provided an application that has version information configured within its manifest file. You can ignore this for now. The next practice covers this in more detail.

10. Test the application.

a. Direct your web browser to the following URL:
`http://host01:7011/AuctionWebApp/index.jsp`

b. First, click *View Auction List* to see the list of auctions. You should get a page with no results. This is because the application is successfully referencing the in-memory version of the library. If you have completed other practices in this course, the database is already populated with auction records. If the application was referencing the database version of the library, auction records would be displayed on this page.

c. Click the *Create Default Data* link to populate the in-memory data model with data for the Auction application. Follow this by clicking the confirmation to create the data. Note that this application informs you that the database is required if the application references the database version of the library. This page also shows you that this is the V1 version of the `AuctionWebApp` application.



Welcome to the Auction application: V1

User: □ update

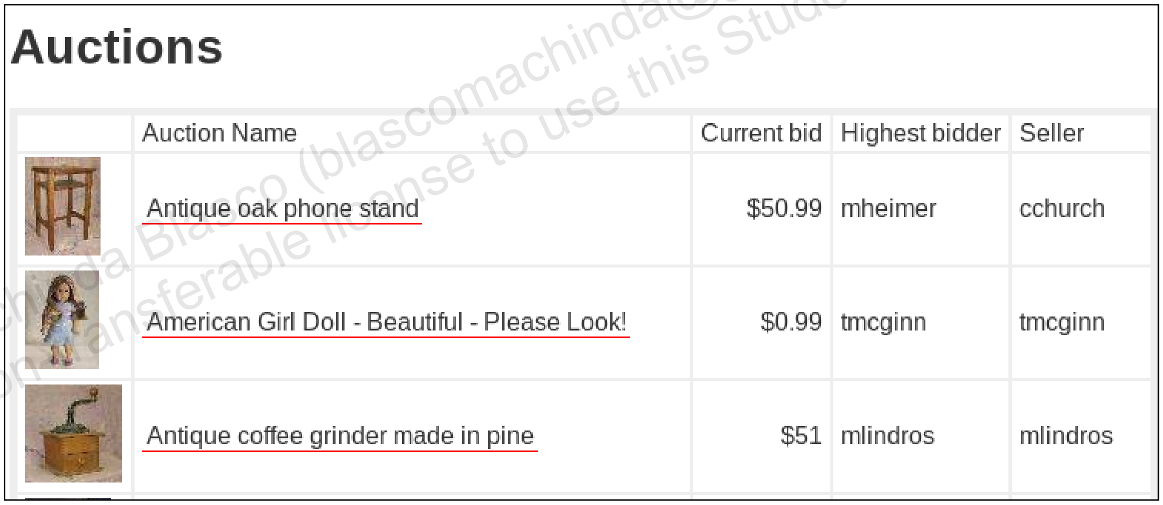
View Auction List

Create Auction

Create Default Data

This project requires a database when referencing AuctionDbLib.
This project does not require a database when referencing AuctionMemLib.

d. Click the *Go Home* link, and then click the *View Auction List* link to view the list of auctions stored in the in-memory data model.



Auctions

	Auction Name	Current bid	Highest bidder	Seller
	Antique oak phone stand	$50.99	mheimer	cchurch
	American Girl Doll - Beautiful - Please Look!	$0.99	tmcginn	tmcginn
	Antique coffee grinder made in pine	$51	mlindros	mlindros

Congratulations! This shows that the Auction application is working properly and is referencing the in-memory version of the library.

11. Shut down all servers and close all windows in preparation for the next lab.

In a host01 terminal window execute:

```
$ /practices/part2/bin/stopDomain.sh
```

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Open a new terminal window **on host01**:
2. Change the current directory to the current practice folder.
3. Execute the solution script:

   ```
   $ ./solution.sh
   ```

   This script performs the following:

   - Ensures that no previous servers are running on both machines
   - Restores the domain and practice to their original state
   - Starts the wlsadmin domain on host01
   - Deploys the solution applications used for this practice
4. Wait for servers **on host01** to fully start.
5. Review the solution files in the practice folder's `solution` folder.
6. Continue starting at step number 10.

Practices for Lesson 10: Application Work Managers

Chapter 10

Practices for Lesson 10: Overview

Practices Overview

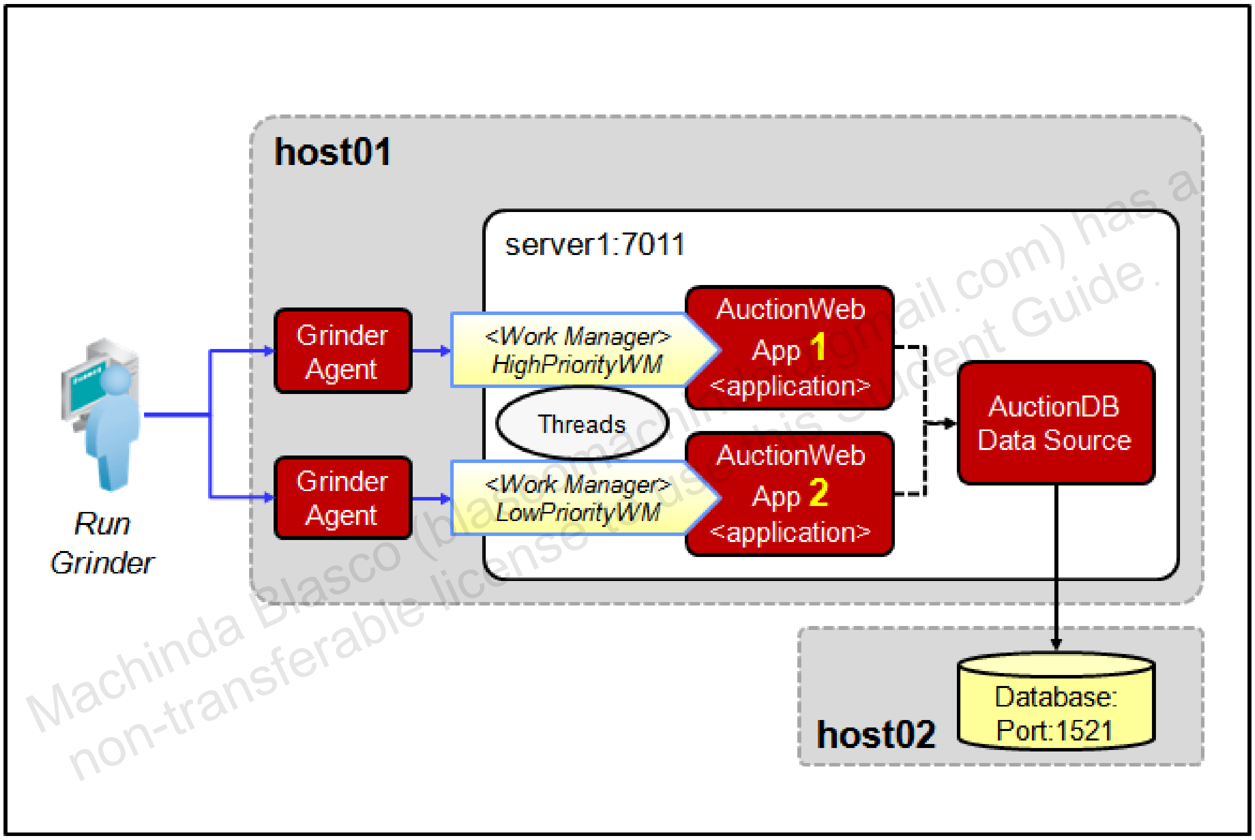
In the practices for this lesson, you create and configure work managers for two instances of the Auction application and run some load tests to see how they affect performance.

Practice 10-1: Creating and Using Work Managers

Overview

Each WebLogic Server instance uses a self-tuning thread pool to process all requests, and all types of requests have the same level of service by default. Work managers enable administrators to prioritize different services, applications, and application components by using request classes and constraints.

The following diagram shows the environment that you will use to experiment with the work manager feature. You will deploy multiple versions of the Auction application and stress test both concurrently using the Grinder:


host01
server1:7011
Run Grinder
Grinder Agent
Grinder Agent
<Work Manager> HighPriorityWM
<Work Manager> LowPriorityWM
Threads
AuctionWeb App 1 <application>
AuctionWeb App 2 <application>
AuctionDB Data Source
host02
Database: Port:1521

This image depicts the architecture of the environment for this practice:

1. There are two instances of the database version of the `SimpleAuction` web application deployed:
   a. `SimpleAuctionWebAppDb1`: Accessed by `context-root`: `SimpleAuctionWebAppDb1`
   b. `SimpleAuctionWebAppDb2`: Accessed by `context-root`: `SimpleAuctionWebAppDb2`
2. You test both versions of the application and verify that it is configured and working correctly.
3. You configure two work managers to set different processing priorities for each application:
   a. `SimpleAuctionWebAppDb1`: Set to a high priority request class
   b. `SimpleAuctionWebAppDb2`: Set to a low priority request class
4. You use deployment plans to assign a work manager to each application.

5. You run Grinder to place a load of requests on each application.
6. You review the results.

Tasks

1. Connect to the **host01** machine.
2. Set up the practice environment.
   a. You may reuse the MAIN terminal window from previous practices. Otherwise, open a terminal window **on host01** by clicking the terminal icon in the launch panel located at the top of the screen.
   b. Set the title of the terminal window by selecting Terminal > Set Title and entering MAIN as the title. This makes it easier to distinguish the purpose of each window.
   c. In the terminal window on `host01`, navigate to the `practice10-01` folder.

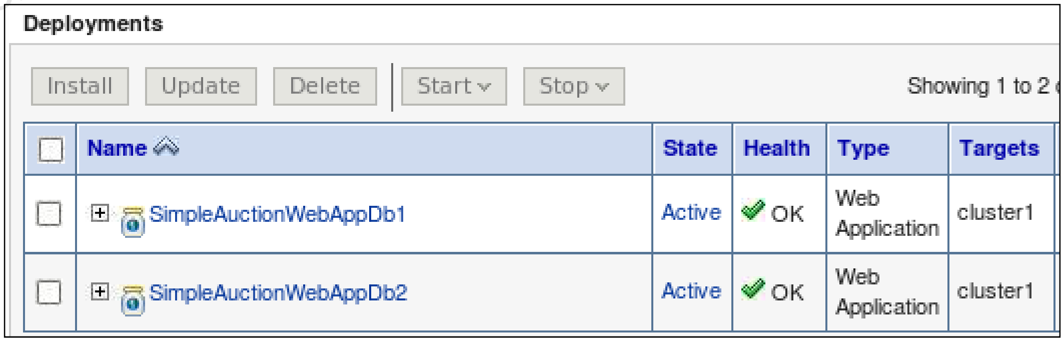
   ```
   $ cd /practices/part2/practice10-01
   ```

   d. Execute the `setup.sh` script to set up the initial practice environment:

   ```
   $ ./setup.sh
   ```

   This script performs the following:
   - Ensures that no previous servers are running on both machines
   - Restores the domain and practice to their original state
   - Starts the `wlsadmin` domain on host01 (host02 is not used in this practice.)
   - Ensures that no applications or libraries are deployed
   - Deploys the starting applications used for this practice
   - **Note:** If the deployment fails with an error similar to CDIExtension cannot rebind because it was versioned before, run the `setup.sh` script again from the MAIN terminal window.

3. Verify the domain's configuration.
   a. Launch a web browser and log in to the administration console:
   `http://host01:7001/console`
   b. Verify that two versions of the `SimpleAuction` application are deployed.



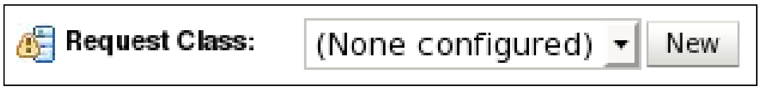
Deployments

☐	Name	State	Health	Type	Targets
☐	⊞ SimpleAuctionWebAppDb1	Active	✓ OK	Web Application	cluster1
☐	⊞ SimpleAuctionWebAppDb2	Active	✓ OK	Web Application	cluster1

4. Test both `SimpleAuctionWebAppDb` applications using the current web browser to ensure that the applications are working as expected:
   a. Browse to `http://host01:7011/SimpleAuctionWebAppDb1`. You should see the Auction application appear.
   b. If you have already created the database, you should be able to click *View Auction List* link and successfully get a list of auctions returned.

c. Repeat the same using the following URL:
   `http://host01:7011/SimpleAuctionWebAppDb2`

5. Create global work managers.
   a. Launch the administration console
   b. Click Lock & Edit
   c. In the Domain Structure panel, select Environment > Work Managers.
   d. Click New.
   e. Select the Work Manager option and click Next.
   f. Name the work manager `HighPriorityWM` and click Next.
   g. Target the work manager to `cluster1` and click Finish.
   h. Edit the new work manager.
   i. Locate the Request Class field and click New:



   j. Select the option Fair Share Request Class and click Next.
   k. Enter the following values:

Field	Value
Name	FairShare90
Fair Share	90

   l. Click Next.
   m. Target the request class to `cluster1` and click Finish.
   n. Click Save.
   o. Repeat these steps to create a second work manager and associated request class by using the details in the following table:

Resource	Field	Value
Work Manager	Name	LowPriorityWM
	Target	`cluster1`
Fair Share Request Class	Name	FairShare20
	Fair Share	20
	Target	`cluster1`

p. Activate your changes.

Note: It is important to understand that a fair share number is not a percentage. This is why the practice purposely does not use fair share numbers that add up to 100%. Instead, they add up to 110, which is also not a percentage. WebLogic adds up all of the fair share class numbers from all work managers and calculates a percentage based on the ratio of the fair share number in relation to the aggregate total of all fair share numbers. The formula for this is similar to (fair share/sum of all fair share classes) = percentage for next request. For this practice, this evaluates to the following:

- Fairshare90: 90/110 = 82% chance for next request
- Fairshare10: 20/110 = 18% chance for next request

Applications that are using HighPriorityWM have an 82% chance of a dispatched request versus applications that are using LowPriorityWM, which have an 18% chance of a dispatched request.

6. Redeploy both `Auction` applications by using a deployment plan to assign a work manager to each application.

   a. Locate the deployment plan files at `/practices/part2/practice10-01/resources`. Confirm that each refers to one of your work managers. For example:

```
<variable>
   <name>SimpleAuctionWeb_DispatchPolicy</name>
   <value>HighPriorityWM</value>
</variable>
```

b. Perform the following commands to redeploy the applications by using the supplied deployment plans:

```
$ ./redeploy.sh
```

c. Test that each application has deployed successfully using a browser:

Note: This drives some traffic that causes each work manager to do some processing.

- `http://host01:7011/SimpleAuctionWebAppDb1`
- `http://host01:7011/SimpleAuctionWebAppDb2`

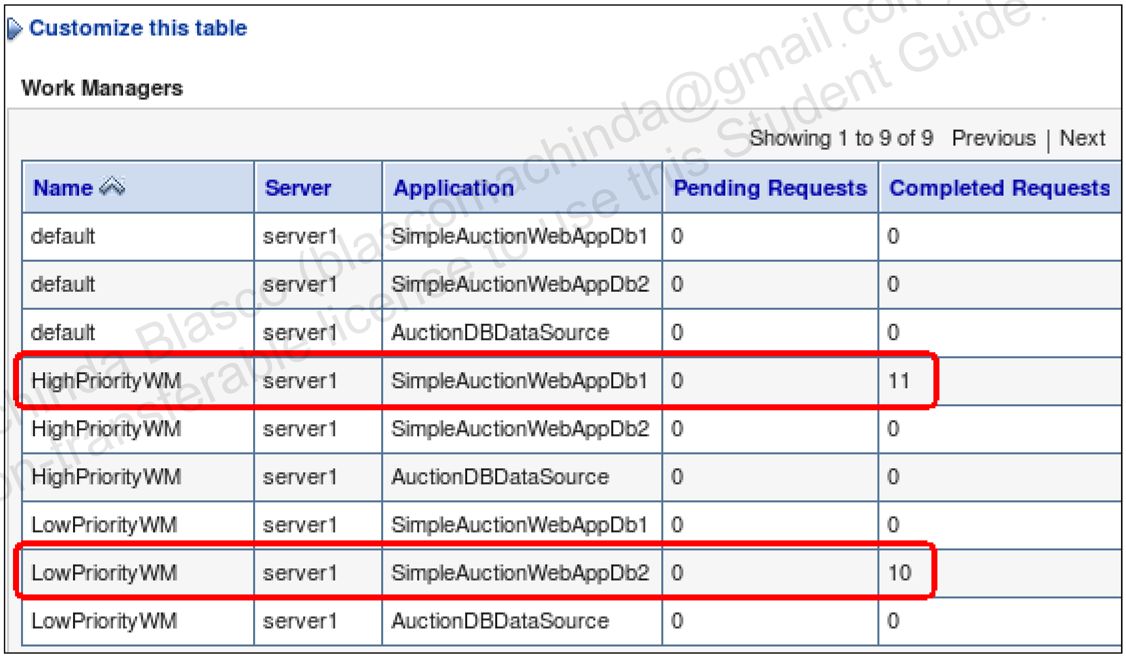
d. Return to the WebLogic administration console.

e. In the Domain Structure panel, select **Deployments**.

f. Click the **Monitoring > Workload** tab.

g. Locate the Work Managers table. Use the Completed Requests column to confirm that each application is linked to the correct work manager.

Note: You may need to use the *Customize this table* link to change the number of rows to display to a number greater than ten in order to see the same view as shown below. Keep in mind that your individual request numbers may be different based on your usage of the applications.



Customize this table

Work Managers

Showing 1 to 9 of 9 Previous | Next

Name	Server	Application	Pending Requests	Completed Requests
default	server1	SimpleAuctionWebAppDb1	0	0
default	server1	SimpleAuctionWebAppDb2	0	0
default	server1	AuctionDBDataSource	0	0
HighPriorityWM	server1	SimpleAuctionWebAppDb1	0	11
HighPriorityWM	server1	SimpleAuctionWebAppDb2	0	0
HighPriorityWM	server1	AuctionDBDataSource	0	0
LowPriorityWM	server1	SimpleAuctionWebAppDb1	0	0
LowPriorityWM	server1	SimpleAuctionWebAppDb2	0	10
LowPriorityWM	server1	AuctionDBDataSource	0	0

Note: If you see that some requests are being handled by the default work manager for either of your applications, you should check your configuration for problems. One common issue that can occur is a spelling mistake in your work manager name. When this happens, the deployment plan name does not match and its configuration does not take effect.

7. Use Grinder agents to stress test the server.

   a. Close the Grinder console and agent if they are running.

   b. Launch two command prompt terminal windows and ensure that you are in the current practice folder. You can reuse the previous prompt shell as well if it is still open.

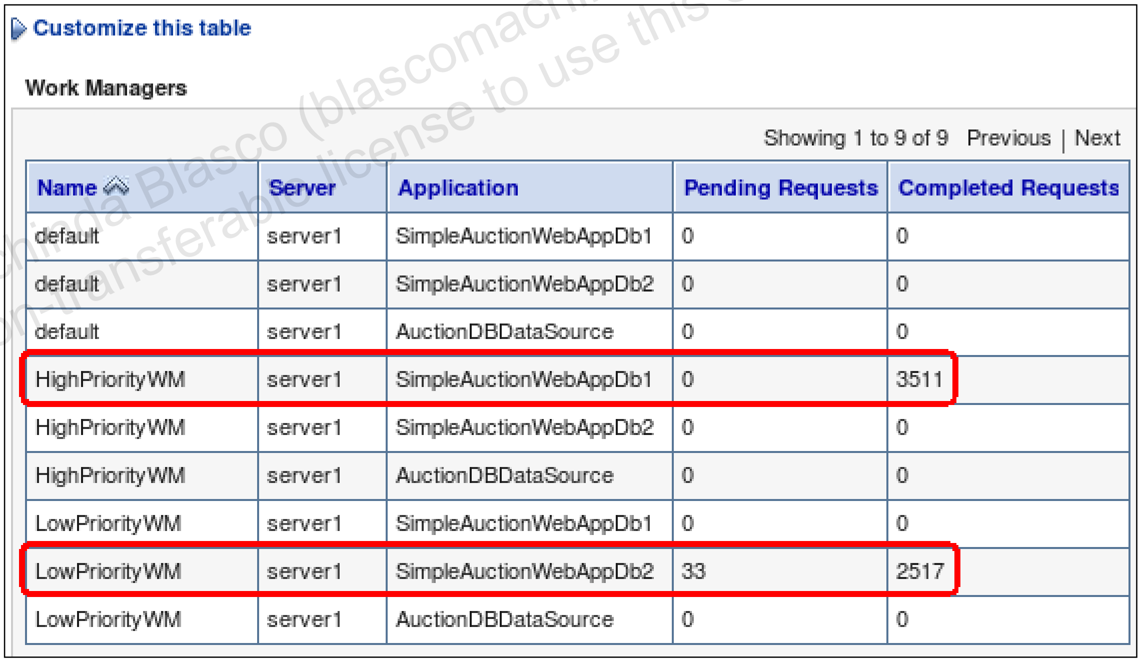
   c. Perform the following command in each window to set the environment for using Grinder:

```
$ . ./setenv.sh
```

   d. Start a Grinder agent in each command prompt at approximately the same time:

```
$ java net.grinder.Grinder grinder-db1.properties


$ java net.grinder.Grinder grinder-db2.properties
```

   e. While the client agents are running, return to the console and refresh the page to view the work load. You should see that HighPriorityWM is getting a larger percentage of the workload and LowPriorityWM is getting a lower percentage of the workload. Your numbers may not match the numbers on the screen, which is perfectly fine. What you are looking for is a higher number of requests handled by the HighPriorityWM work manager. You should also note that when both Grinder loads are finished, and all requests are handled, both servers will show a similar number of completed requests. The key is to watch the numbers during the load test. After the test, the numbers have no meaning.



Work Managers

Showing 1 to 9 of 9   Previous | Next

Name	Server	Application	Pending Requests	Completed Requests
default	server1	SimpleAuctionWebAppDb1	0	0
default	server1	SimpleAuctionWebAppDb2	0	0
default	server1	AuctionDBDataSource	0	0
HighPriorityWM	server1	SimpleAuctionWebAppDb1	0	3511
HighPriorityWM	server1	SimpleAuctionWebAppDb2	0	0
HighPriorityWM	server1	AuctionDBDataSource	0	0
LowPriorityWM	server1	SimpleAuctionWebAppDb1	0	0
LowPriorityWM	server1	SimpleAuctionWebAppDb2	33	2517
LowPriorityWM	server1	AuctionDBDataSource	0	0

   f. Wait for both agents to finish and exit.

8. Inspect the performance results.

   a. Open the `logs-db1/host01.example.com-0.log` file **(in the practice folder)**.

   b. At the end of the log file, locate the row starting with the text "Totals." Record the following performance data from this row:

      – Mean Test Time (ms): _________

      – TPS: ________

      Tip: You may want to disable line wrapping in your text editor.

   c. Repeat the previous steps on the `logs-db2/host01.example.com-0.log` file.

   d. Compare the results.

      Congratulations! You have successfully created, configured, and used work managers for different applications.

9. Shut down the environment.

   a. Press **Ctrl + C** in all terminal windows that are running the servers.

   b. Close all terminal windows used for each server.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Open a new terminal window **on host01**:
2. Change the current directory to the current practice folder.
3. Execute the solution script:

   ```
   $ ./solution.sh
   ```

   The solution script performs the following:

   - Ensures that no previous servers are running on both machines
   - Restores the domain and practice to their original state
   - Starts the `AdminServer` and `server1` servers on host01
   - Deploys the basic practice applications without deployment plans
   - Creates the work managers used for the practice
   - Redeploys the applications using deployment plans that associate each application with a work manager
4. Wait for servers **on host01** to fully start.
5. Continue starting at step number 6c.

Practices for Lesson 11: Working with the Security Realm

Chapter 11

Practices for Lesson 11: Overview

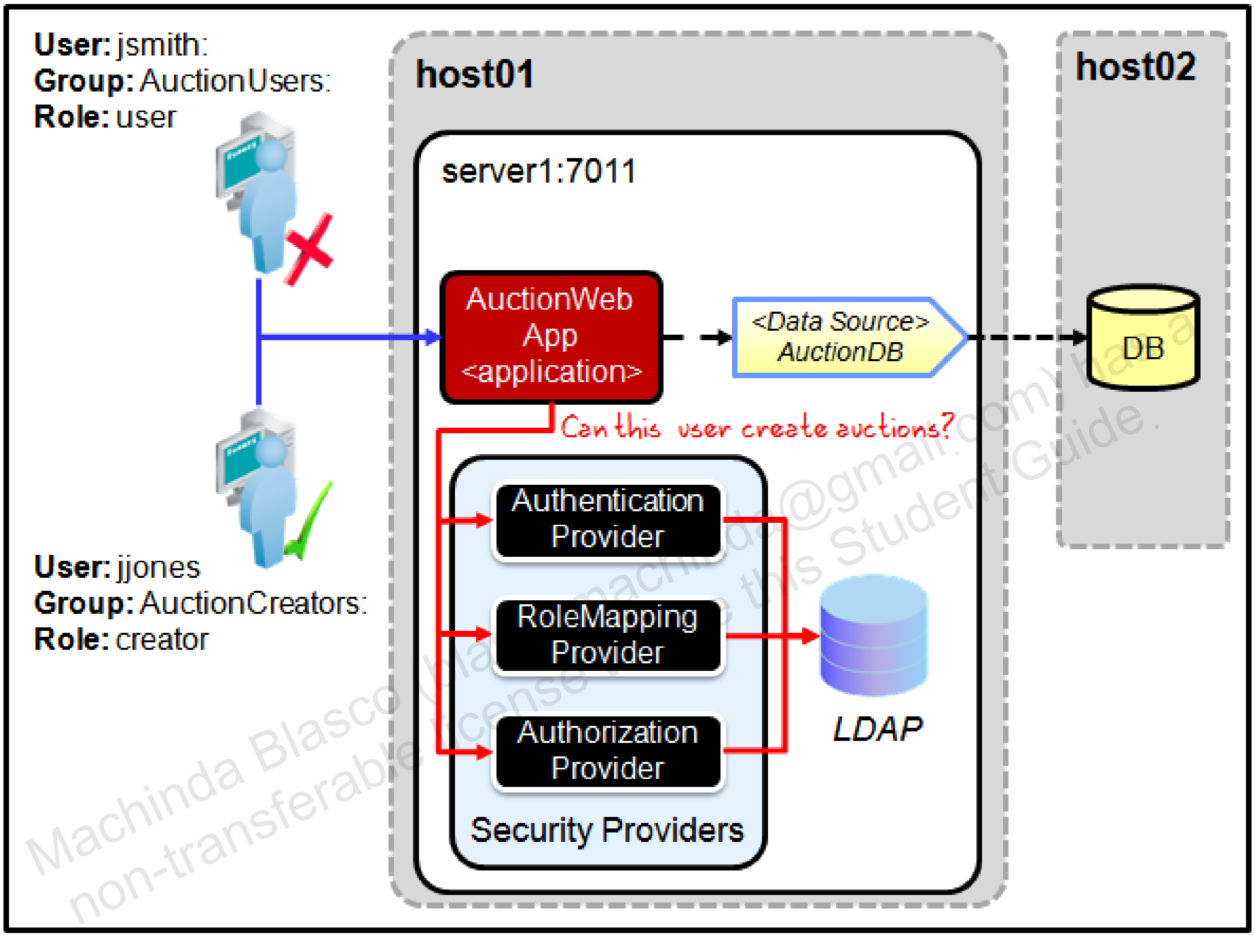
Practices Overview

In the practices for this lesson, you use the administration console to create users, groups, roles, and a policy to control access to an application resource, and configure the WebLogic default auditing provider.

Practice 11-1: Creating Users, Groups, Roles, and Policies

Overview

In this practice, you create two users and two groups. Based on the group membership of the user, each is mapped into an application role. This role is used in a policy to determine if each user can access the Create Auction feature of the Auction application.



This image depicts the architecture of the environment for this practice:

1. You create two users and groups and assign each user to one of the groups:
   a. `jsmith` is a member of the AuctionUsers group.
   b. `jjones` is a member of the AuctionCreators group.
2. You create two roles and map each user to a role based on the user's group membership:
   a. Users that are members of the AuctionUsers group are assigned the user role.
   b. Users that are members of the AuctionCreators group are assigned the creator role.
3. You create an authorization policy that protects the URL for the `createAuction.jsp` page:
   a. Users that are not assigned the creator role are denied access.
   b. Users that are assigned the creator role are granted access.
4. You use the application logged in as `jsmith`, the WebLogic security providers perform the authorization process, and you are denied access to create auctions.
5. You use the application logged in as `jjones`, the WebLogic security providers perform the authorization process, and you are granted access to create auctions.

Tasks

1. Connect to the **host01** machine.
2. Set up the practice environment.
   a. You may reuse the MAIN terminal window from previous practices. Otherwise, open a terminal window **on host01** by clicking the terminal icon in the launch panel located at the top of the screen.
   b. Set the title of the terminal window by selecting Terminal > Set Title and entering `MAIN` as the title. This makes it easier to distinguish the purpose of each window.
   c. Within the terminal window **on host01**, navigate to the `practice11-01` folder.

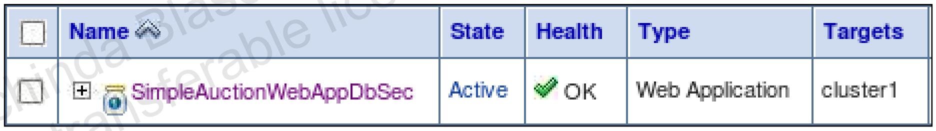
   ```
   $ cd /practices/part2/practice11-01
   ```

   d. Execute the `setup.sh` script to set up the initial practice environment:

   ```
   $ ./setup.sh
   ```

   This script performs the following:

   - Ensures that no previous servers are running on both machines
   - Restores the domain and practice to their original state
   - Starts the `wlsadmin` domain on host01 (host02 is not used in this practice.)
   - Ensures that no applications or libraries are deployed
   - Deploys the starting application used for this practice
3. Verify the domain's configuration.
   a. Launch a web browser and log in to the administration console:
   `http://host01:7001/console`
   b. Navigate to **Deployments** and verify that the `SimpleAuctionWebAppDbSec` application is deployed.



□	Name	State	Health	Type	Targets
□	⊞ SimpleAuctionWebAppDbSec	Active	✓ OK	Web Application	cluster1

4. Test the `SimpleAuctionWebAppDbSec` application using the current web browser to ensure that the application is working as expected:
   a. Browse to `http://host01:7011/SimpleAuctionWebAppDbSec`. You should see the Auction application appear.
   b. If you have already created the database, you should be able to click the *View Auction List* link and successfully get a list of auctions returned. This verifies that the `orcl` database is started and functioning properly.
5. Review the application's security declared in its deployment descriptors.
   a. Perform the following steps to review the declarative security defined in the application's `weblogic.xml` deployment descriptor:

   ```
   $ gedit resources/SimpleAuctionWebAppDbSec/WEB-INF/weblogic.xml
   ```

b. Find the configuration related to security. It should resemble the following:

```
<wls:security-role-assignment>
  <wls:role-name>user</wls:role-name>
  <wls:principal-name>AuctionUsers</wls:principal-name>
</wls:security-role-assignment>
<wls:security-role-assignment>
  <wls:role-name>auctionCreators</wls:role-name>
  <wls:principal-name>AuctionCreators</wls:principal-name>
</wls:security-role-assignment>
```

Why? This code defines two security roles:

- `user`: Any user that is mapped by the system to the AuctionUsers principal (or group) is assigned the `user` role.
- `auctionCreators`: Any user that is mapped by the system to the AuctionCreators principal is assigned the `auctionCreators` role.

c. Open the application's `web.xml` deployment descriptor to review the declarative security defined in it.

```
$ gedit resources/SimpleAuctionWebAppDbSec/WEB-INF/web.xml
```

d. Find the configuration related to security. Review here what each configuration setting is used to accomplish:

The `error-page` tag allows you to configure a page that the server displays when certain error conditions are met. In this case, when the `403 forbidden` error code is encountered, WebLogic should display the `userRequired.jsp` page to the user. This page usually contains some information about why the user is having trouble.

```
<error-page>
  <error-code>403</error-code>
  <location>/userRequired.jsp</location>
</error-page>
```

The security-constraint tag enables you to configure which principals are allowed access to resources of the application. In this case, the auctionCreators role is configured as the principal that is granted access to two URL patterns related to creating auctions.

```
<security-constraint>
  <display-name>create auction</display-name>
  <web-resource-collection>
    <web-resource-name>createAuction</web-resource-name>
    <url-pattern>/createAuction.jsp</url-pattern>
    <url-pattern>/CreateAuctionServlet</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>auctionCreators</role-name>
  </auth-constraint>
</security-constraint>
```

The security-role tag configures a role name that maps to the configuration specified in the weblogic.xml deployment descriptor. The determination of how principals are assigned these roles is defined in the weblogic.xml file.

```
<security-role>
  <role-name>user</role-name>
</security-role>
<security-role>
  <role-name>auctionCreators</role-name>
</security-role>
```

The login-config tag configures the login method used by the server to let users enter their usernames and passwords to log in to the system. In this case, form-based authentication is configured for the WebLogic security realm called myrealm. Form-based logins require you to declare which HTML-based pages to use for letting users log in and for displaying login errors. In this case, the login.jsp page is used to let users enter their credentials and the loginError.jsp page is displayed when users fail to authenticate successfully.

```
<login-config>
  <auth-method>FORM</auth-method>
  <realm-name>myrealm</realm-name>
  <form-login-config>
    <form-login-page>/login.jsp</form-login-page>
    <form-error-page>/loginError.jsp</form-error-page>
  </form-login-config>
</login-config>
```

Important! Review the code snippets of this step again. Note which snippets are in bold and which are not in bold. The snippets that are NOT in bold are related to security roles and policies. The snippets that ARE in bold are related to the general authentication process. This is important because the default behavior when an application is deployed to WebLogic is that the security roles and policies declared in deployment descriptors are automatically applied. **However**, the application for this practice was deployed using the `-securityModel CustomRolesAndPolicies` option, which instructs WebLogic to completely ignore all declarative roles and policies defined in the application's deployment descriptors.

Why? The purpose of this practice is to show you how to configure users, groups, roles, and policies. The configuration of roles and policies is supported within deployment descriptors and directly within the embedded LDAP server. You also have the ability to ignore the configuration in deployment descriptors. This practice shows you what the configuration looks like within a deployment descriptor and how to override it using the administration console and deployment options.

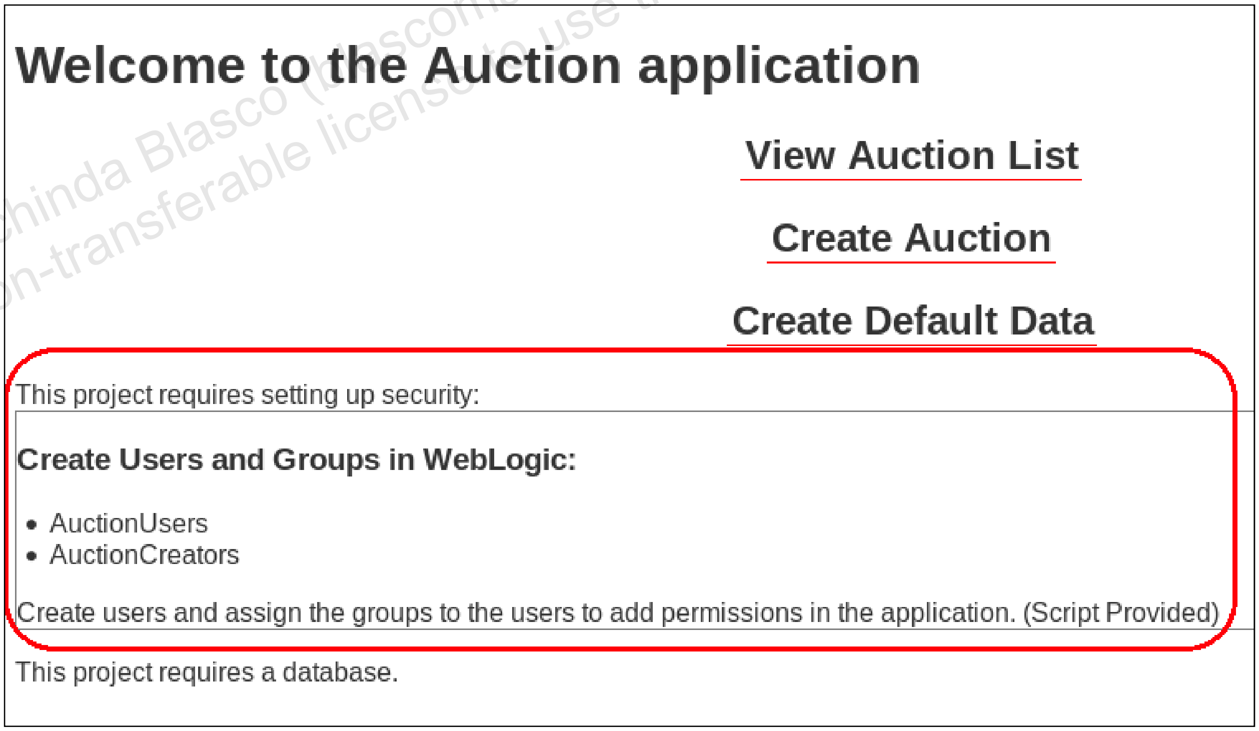
6. Test the unprotected application.

   The currently deployed application is not protected with any security policies because you ignored it during deployment. You also did not create any users or groups related to this application. If you take a few minutes and explore the users, groups, roles, and policies associated with the domain, you will see that only the default WebLogic identities are configured.

   a. Direct your web browser to the following URL:

      `http://host01:7011/SimpleAuctionWebAppDbSec`

      You should see a slightly different variation of the application that indicates it is a security-based version of the Auction application.



Welcome to the Auction application

This project requires setting up security:

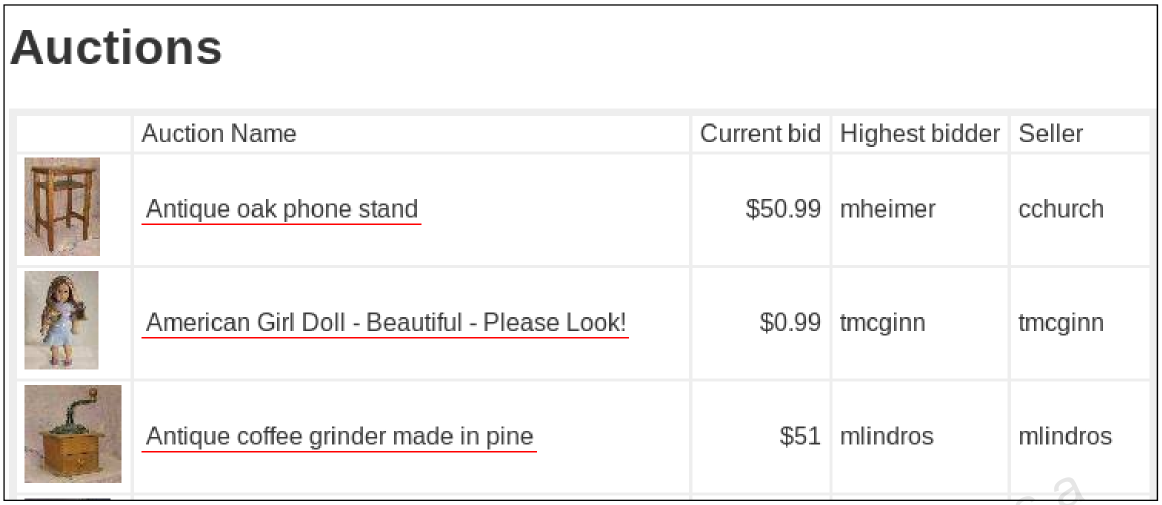
Create Users and Groups in WebLogic:

- AuctionUsers
- AuctionCreators

Create users and assign the groups to the users to add permissions in the application. (Script Provided)

This project requires a database.

b. Click the *View Auction List* link to view the list of auctions stored in the database.

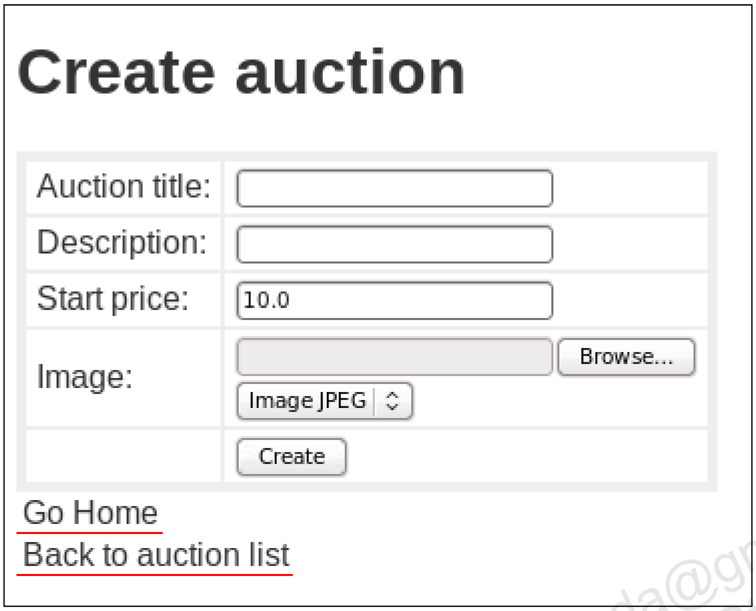


Auctions

	Auction Name	Current bid	Highest bidder	Seller
	Antique oak phone stand	$50.99	mheimer	cchurch
	American Girl Doll - Beautiful - Please Look!	$0.99	tmcginn	tmcginn
	Antique coffee grinder made in pine	$51	mlindros	mlindros

c. Click the *Go Home* link and click the *Create Auction* link to create an auction. You should see the *Create auction* page appear.

Why? This page is displayed because there is no security protecting it as a resource in WebLogic at the moment. This proves that no login was required to view the page.



Create auction

Auction title:	
Description:	
Start price:	10.0
Image:	Browse... Image JPEG
	Create

Go Home

Back to auction list

7. Create users and groups and assign users to groups by using the administration console.

a. In the **Domain Structure** panel, select **Security Realms > myrealm > Users and Groups > Groups** to display the page for configuring groups.

b. Perform the following steps to create the AuctionUsers group:

Note: These changes are not managed as part of the typical change management system of the console. There is no need to lock and edit or activate changes for these steps.

– Click **New** to create a new group.

– Enter `AuctionUsers` as the group name.

– Leave the rest of the parameters at their default values.

– Click **OK**.

c. Repeat the last step to create the `AuctionCreators` group.

d. Click the **Users** tab to display the page for configuring users.

e. Perform the following steps to create user `jsmith` and assign the user to the `AuctionUsers` group:

– Click **New** to create a new user.

– Enter `jsmith` as the username.

– Enter `Welcome1` as the password and confirmation password.

– Leave the rest of the parameters at their default values.

– Click **OK**.

– Click `jsmith` to display the user's configuration page.

– Click the Groups tab.

   - Select the `AuctionUsers` group from the Available column and use the » button to move it to the Chosen column to make `jsmith` a member of the `AuctionUsers` group.
   - Click **Save**.

f. Repeat the previous step to create user `jjones` and assign this user to the `AuctionCreators` group. Return to the Users page using the breadcrumb trail at the top of the page by selecting Users and Groups.

8. Create two roles and map them to their groups.
   a. In the **Domain Structure** panel, select **Deployments > SimpleAuctionWebAppDbSec > Security > Application Scope > Roles** to display the page for configuring roles for the application.
   b. Create the `user` role and map it to the `AuctionUsers` group:
      - Click **New** to create a new role.
      - Enter `user` as the role name.
      - Leave the rest of the parameters at their default values.
      - Click **OK**.
      - Click the `user` role to display the role's configuration page.
      - Click *Add Conditions* to add an expression that defines which users are assigned the `user` role.
      - Select Group as the *Predicate List* to map members of a group to the `user` role and click **Next**.
      - Enter `AuctionUsers` as the *Group Argument Name*, click Add, and click Finish to map members of the AuctionUsers group to the `user` role.
      - Click **Save**.
   c. Repeat the previous step to create the creator role and map it to the `AuctionCreators` group. Return to the Roles page using the breadcrumb trail at the top of the page by selecting Roles.

9. Create a policy that grants users in the creator role access to the `createAuction.jsp` page.
   a. Return to the Roles page using the breadcrumb trail at the top of the page by selecting Roles.
   b. Select the URL **Patterns > Policies** tab to display the page for configuring policies for the application's URL patterns.
   c. Click **New** to create a new URL pattern to protect.
   d. Enter `/createAuction.jsp` as the URL pattern to protect.
   e. Click **OK**.
   f. Click `/createAuction.jsp` to display its configuration page.
   g. Click **Add Conditions** to add an expression that defines the criteria by which allowed users are granted or denied access to this URL.
   h. Select **Role** as the *Predicate List* to indicate that you want users assigned to a particular role for access and click **Next**.

i. Enter `creator` as the *Role Argument Name*, click **Add**, and click **Finish** to indicate that a user must be in the `creator` role in order to be granted access.

j. Click **Save**.

10. Run the application as each user to test your security configuration.

Note: You will use Firefox's private browsing feature to switch between logins for each user. This is an easy way of invalidating a previous session, so a new user can log in.

a. In Firefox, select **File > New Private Window**.

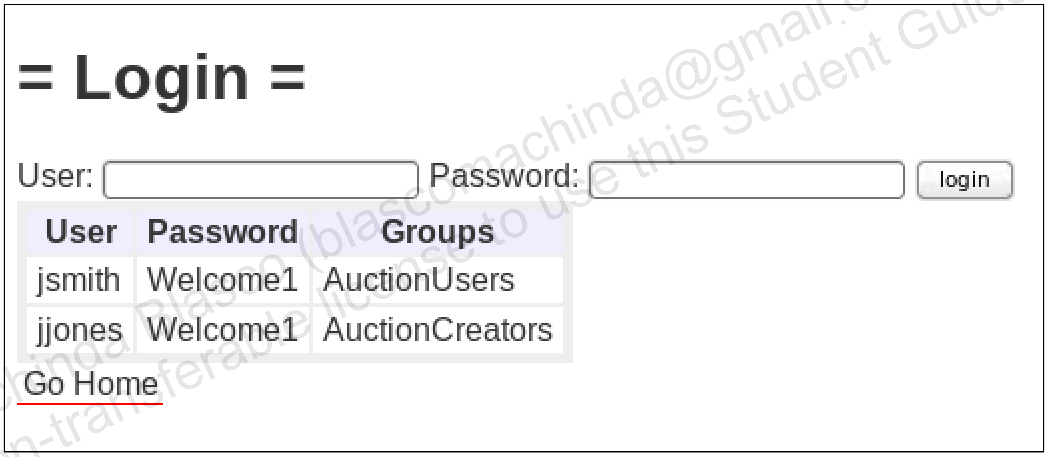
b. Click Start Private Browsing.

c. Browse to `http://host01:7011/SimpleAuctionWebAppDbSec` to use the Auction application.

d. Click the *View Auction List* link to view the list of auctions stored in the database. This link should still work because it is still not protected by any security policies.

e. Click the *Create Auction* link on the bottom of the page to create an auction. You are automatically presented with a login page so you can enter credentials.

Why? This page is displayed because now there is a security policy protecting it as a resource in WebLogic at the moment. This proves that your security configuration is working so far.

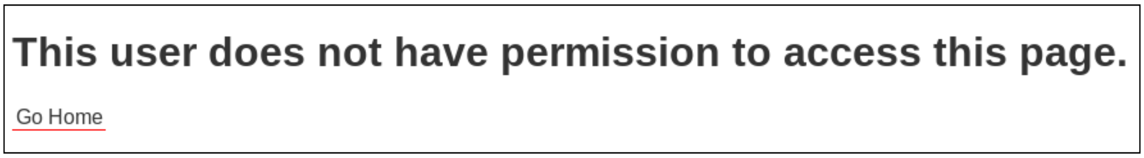


= Login =

User: Password:

User	Password	Groups
jsmith	Welcome1	AuctionUsers
jjones	Welcome1	AuctionCreators

f. Enter **`jsmith`** as the User and **`Welcome1`** as the Password and click "login." You should be denied access because user `jsmith` is not mapped to the creator role. This proves that your security policy is properly denying unauthorized access.



This user does not have permission to access this page.

g. Select **Tools > Stop** Private Browsing to end this session.

h. Repeat the previous steps and change the user from `jsmith` to `jjones`. This time you should see the Create auction page after logging in. This is because user `jjones` is granted access because this user is mapped to the creator role. This proves that your security policy is properly granting authorized access.

i. Select **Tools > Stop** Private Browsing to end this session.

Congratulations! You have successfully configured users, groups, roles, and policies for a WebLogic Server application.

11. Leave the environment running for the next practice.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Open a new terminal window **on host01**:
2. Change the current directory to the current practice folder.
3. Execute the solution script:

   ```
   $ ./solution.sh
   ```

   The solution script performs the following:

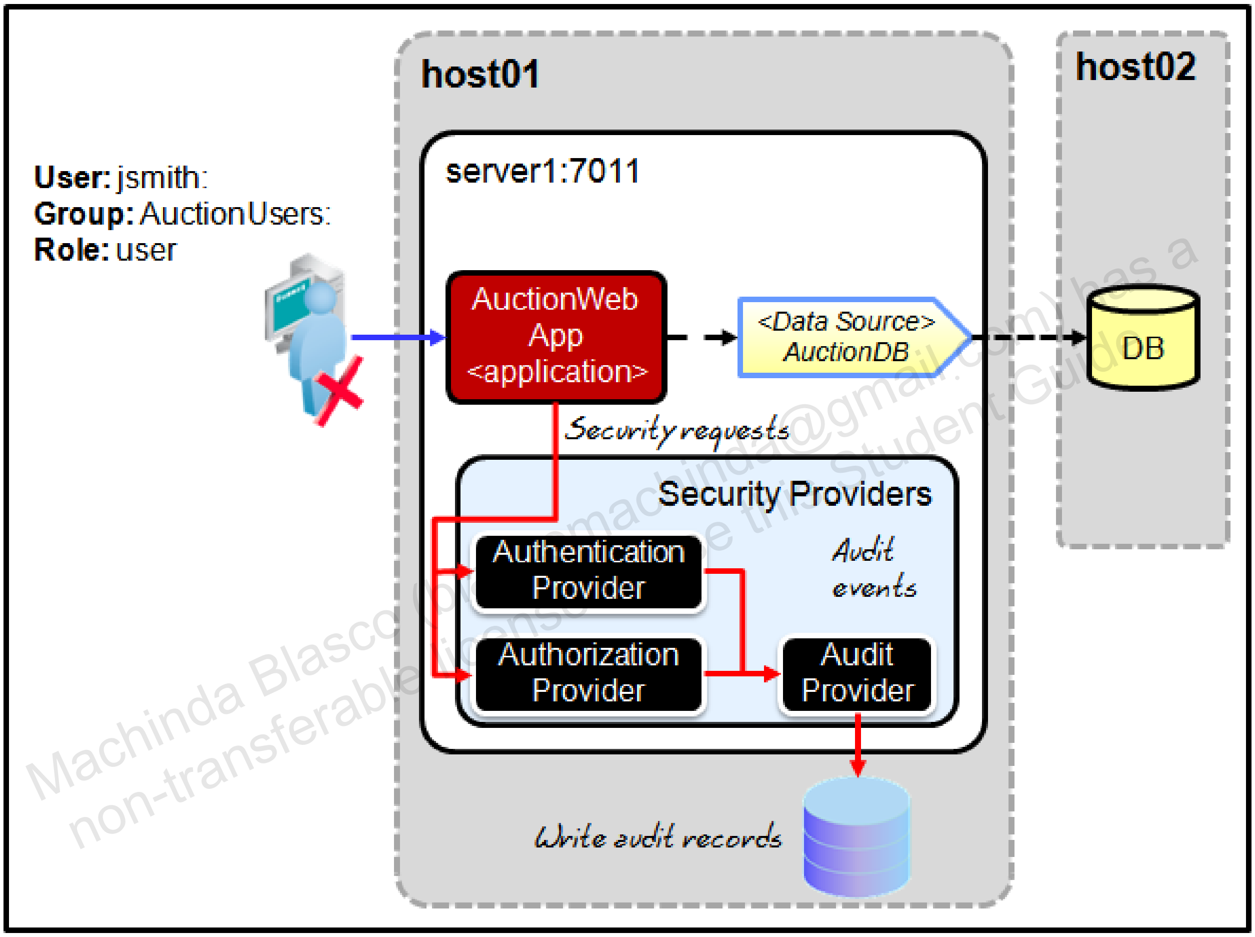
   - Ensures that no previous servers are running on both machines
   - Restores the domain and practice to their original state
   - Starts the wlsadmin domain on host01
   - Deploys the solution applications used for this practice
   - Creates all the security configuration required for this practice
4. Wait for servers **on host01** to fully start.
5. Continue starting at step number 10.

Practice 11-2: Configuring WebLogic Auditing

Overview

Using WebLogic auditing, you can keep track of what users are doing in the system. This includes failed security attempts, which is a useful tool for identifying when someone is trying to do something that they should not be doing in the first place.

This practice builds on the environment of the previous practice. In this practice, you configure the WebLogic DefaultAuditor security provider to log failed authorization requests.



This image depicts the architecture of the environment for this practice:

1. This practice uses the domain from the previous practice.
2. You create the WebLogic DefaultAuditor security provider.
3. You configure the auditing provider to log failed authorization requests.
4. You run the application as user `jsmith` and try to access the *Create auction* page.
5. You locate and review the audit log to see if the failure was logged.

Tasks

1. Connect to the **host01** machine.
2. Set up the practice environment.
   a. You may reuse the MAIN terminal window from previous practices. Otherwise, open a terminal window **on host01** by clicking the terminal icon in the launch panel located at the top of the screen.
   b. Set the title of the terminal window by selecting **Terminal > Set Title** and entering `MAIN` as the title. This makes it easier to distinguish the purpose of each window.
   c. In the terminal window **on host01**, navigate to the `practice11-02` folder.

```
$ cd /practices/part2/practice11-02
```

   d. Execute the `setup.sh` script to set up the initial practice environment:

   Optional: Execute the setup script only if you have not completed the previous practice. If you have completed the previous practice and the domain is still running, then skip this step and continue with the next step.

```
$ ./setup.sh
```

   This script performs the following:

   – Ensures that no previous servers are running on both machines
   – Starts the `wlsadmin` domain on host01 (host02 is not used in this practice.)
   – Ensures that no applications or libraries are deployed
   – Deploys the starting application used for this practice
   – Ensures that the WebLogic domain is set to the starting point of the practice

3. Create the WebLogic DefaultAuditor provider.
   a. Launch a web browser and log in to the administration console:
   `http://host01:7001/console`
   b. In the **Domain Structure** panel, navigate to **Security Realms > myrealm > Providers > Auditing** to display the configuration page for auditing providers.
   c. Click **Lock & Edit.**
   d. Click **New** to create a new provider.
   e. Enter the following values:

Field	Value
Name	`DefaultAuditor`
Type	DefaultAuditor

   f. Click **OK**.

4. Configure the auditing provider.

   Why? You created a new auditing provider, but you have not configured it to do anything yet.

   a. Click your new **DefaultAuditor** link to display its configuration page.
   b. Click the **Provider Specific** page to display the provider's configuration options.
   c. For this practice, you are going to configure only the severity level of events to audit. You can explore other options as well if you like. Set the Severity field to CUSTOM.

      Why? When you set severity to CUSTOM, WebLogic allows you to specify more than one type of event to audit for your applications.
   d. Select **Error, Success, and Failure** audit severity to enable each type.
   e. Click **Save**.
   f. Activate your changes.

5. Restart the servers.

   Why? Whenever a security provider is created, all servers must be restarted to realize the changes.

   a. In the terminal windows for the AdminServer and server1 servers, press **Ctrl + C** to kill the servers and close the terminal windows.
   b. Execute the following commands to start the servers again and wait for the servers to start completely before moving to the next step.

```
$ startAdmin.sh
$ startServer1.sh
```

6. Test the application using user `jsmith`:

   a. Browse to the following URL and click the *Create Auction* link.
      - `http://host01:7011/SimpleAuctionWebAppDbSec`
   b. Log in as `jsmith`. Remember that the password is `Welcome1`. You should be denied access, which is good because you want a failure that gets logged by the auditing provider.

7. Locate and review auditing logs.

   a. In a terminal window, navigate to the logs folder for `server1` and open the audit log for viewing:

```
$ cd /u01/domains/part2/wlsadmin/servers/server1/logs
$ gedit DefaultAuditRecorder.log
```

   b. Perform a search for `jsmith` in the file to find the audit record associated with the authorization failure. That record should look similar to the following:

```
#### Audit Record Begin
<Apr 11, 2013 9:10:10 PM>  <Severity =SUCCESS>  <<<Event Type =
Authentication Audit Event><jsmith><AUTHENTICATE>>>
Audit Record End ####
#### Audit Record Begin
<Apr 11, 2013 9:10:10 PM>  <Severity =FAILURE>  <<<Event Type =
Authorization Audit Event V2 ><Subject: 2
Principal = class
weblogic.security.principal.WLSUserImpl("jsmith")
Principal = class
weblogic.security.principal.WLSGroupImpl("AuctionUsers")
```

```
><ONCE><<url>><type=<url>, application=SimpleAuctionWebAppDbSec,
contextPath=/SimpleAuctionWebAppDbSec, uri=/createAuction.jsp,
httpMethod=GET>>>
Audit Record End ####
```

Congratulations! You have successfully configured the WebLogic default auditing provider.

8. Shut down the environment.
   a. Press **Ctrl + C** in all terminal windows that are running servers.
   b. Close all terminal windows used for each server.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Open a new terminal window **on host01**:
2. Change the current directory to the current practice folder.
3. Execute the solution script:

   ```
   $ ./solution.sh
   ```

   The solution script performs the following:

   - Executes the solution for practice11-01, which sets the initial environment for the practice
   - Creates the WebLogic default auditing provider and configures it
   - Restarts the servers to realize the audit provider configuration
4. Wait for servers **on host01** to fully start.
5. Continue starting at step number 6.

Practices for Lesson 12: Disaster Recovery and Migration

Chapter 12

Practices for Lesson 12: Overview

Practices Overview

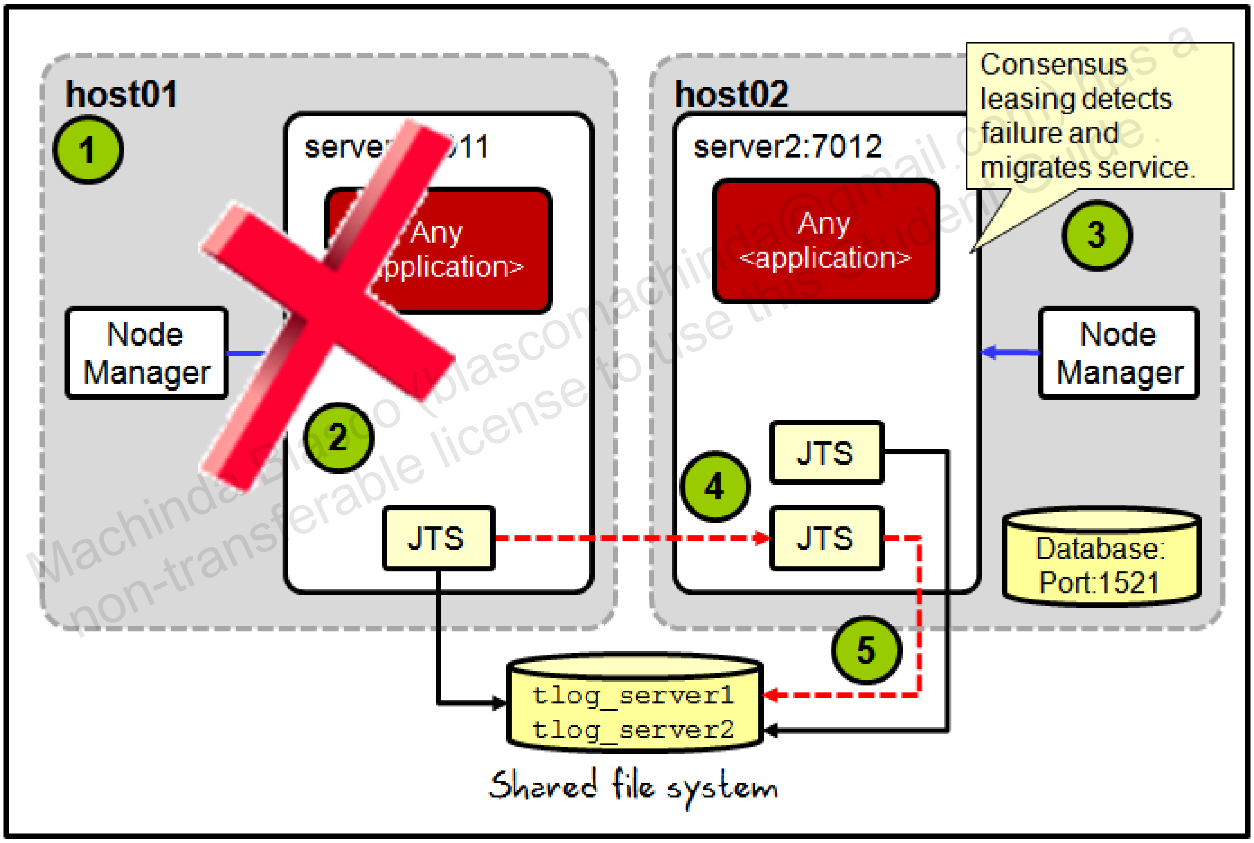
In the practices for this lesson, you use the administration console to configure the `wlsadmin` domain to automatically migrate the JTA service from one server to another server in the cluster.

Practice 12-1: Configuring JTA Service-Level Migration

Overview

In this practice, you configure the `wlsadmin` domain to automatically migrate the Java Transaction Service (JTS) from one server in a cluster to another server in the cluster. You will perform the following steps:

1. Review the existing machine and Node Manager configuration.
2. Configure consensus leasing for a cluster.
3. Enable automatic JTA migration for each server in the cluster.
4. Configure candidate machines for automatic migration.
5. Configure a persistent store for each server, so they are accessible to all servers in the cluster on a shared file system.
6. Test automatic migration.



This image depicts the architecture of the environment for this practice:

1. The domain, which is running normally, comprises two managed servers in a cluster. The Java Transaction Service (JTS) is configured for automatic migration in the event of a failure. Each server in the cluster uses a transaction log (TLOG) in a persistent store that is located on a shared file system so it is available to all server members in the cluster. Server1's TLOG is called `tlog_server1` and server2's TLOG is called `tlog_server2`.
2. `Server1` fails on host01.
3. WebLogic's consensus leasing mechanism detects the failure and initiates the migration of any services that can migrate.
4. The Java Transaction Service (JTS) is migrated from `server1` to `server2`.

5. The migrated service recovers any outstanding transactions for `server1` from `server2`. After all work is recovered, the service automatically fails back to `server1` so it is available when the server starts up again.

Tasks

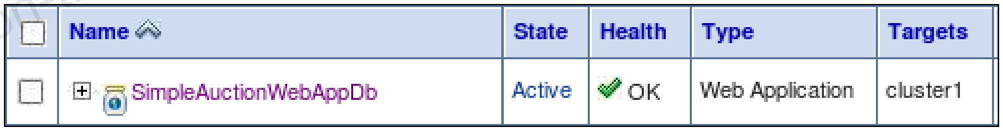
1. Connect to the **host01** machine.
2. Set up the practice environment.
   a. You can reuse the MAIN terminal window from previous practices. Otherwise, open a terminal window **on each machine** by clicking the terminal icon in the launch panel located at the top of the screen.
   b. Set the title of the terminal window by selecting Terminal > Set Title and entering `MAIN` as the title. This makes it easier to distinguish the purpose of each window.
   c. In the terminal window **on host01**, navigate to the `practice12-01` folder:

   ```
   $ cd /practices/part2/practice12-01
   ```

   d. Execute the `setup.sh` script to set up the initial practice environment:

   ```
   $ ./setup.sh
   ```

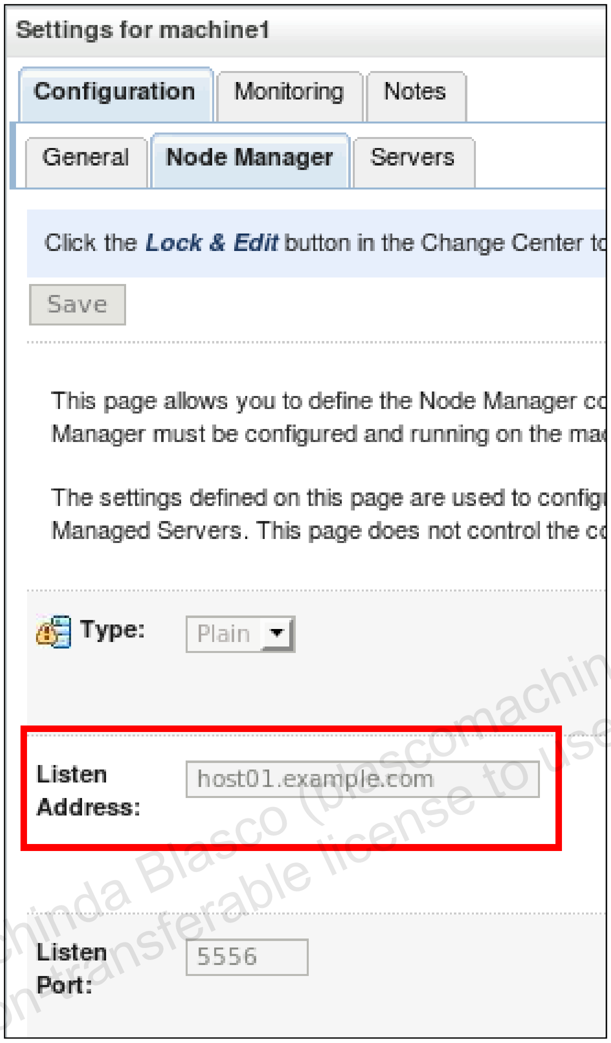
   This script performs the following:
   - Ensures that no previous servers are running on both machines
   - Restores the domain and practice to their original state
   - Starts the `wlsadmin` domain on host01 and host02
   - Starts the Node Manager on host01 and host02
   - Ensures that no applications or libraries are deployed
   - Deploys the starting application used for this practice
3. Verify the domain's configuration.
   a. Launch a web browser and log in to the administration console:
   `http://host01:7001/console`
   b. Verify that the `SimpleAuctionWebAppDb` application is deployed.



□	Name	State	Health	Type	Targets
□	⊞ SimpleAuctionWebAppDb	Active	✓ OK	Web Application	cluster1

4. Test the `SimpleAuctionWebAppDb` application using the current web browser to ensure that the application is working as expected:
   a. Browse to `http://host01:7011/SimpleAuctionWebAppDb`. You should see the Auction application appear.
   b. If you have already created the database, you should be able to click the *View Auction List* link and successfully get a list of auctions returned. This verifies that the `orcl` database is started and functioning properly.
5. Review the machine configuration.
   a. In the **Domain Structure** pane, expand **Environment** and select **Machines**.
   b. You should see two machines configured: *machine1* and *machine2*.
   c. Click **machine1** to view its settings.

d. Click the **Node Manager** tab to view the settings for this machine. You should see that the Node Manager is already configured and that this machine is associated with *host01*.



Settings for machine1

Click the ***Lock & Edit*** button in the Change Center to

This page allows you to define the Node Manager co
Manager must be configured and running on the mac

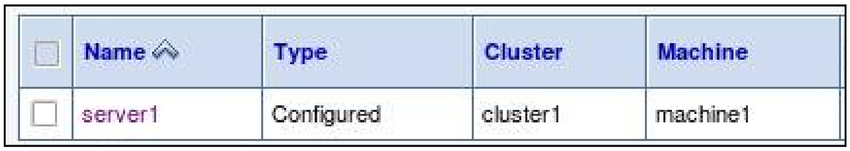
The settings defined on this page are used to config
Managed Servers. This page does not control the co

Type: Plain

Listen Address: host01.example.com

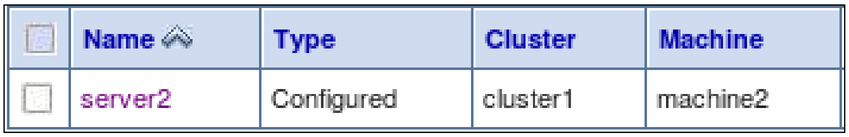
Listen Port: 5556

e. Click the **Servers** tab to view the servers that are assigned to *machine1*. You should see that the `server1` is configured for machine1.



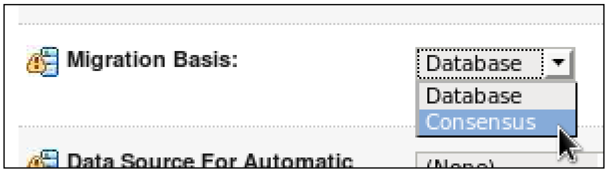
□	Name	Type	Cluster	Machine
□	server1	Configured	cluster1	machine1

f. Repeat these steps. However, select *machine2* instead of *machine1* to view the Node Manager settings and servers configured for that machine. You should see that the Node Manager is running on host02 and `server2` is configured for machine2.



□	Name	Type	Cluster	Machine
□	server2	Configured	cluster1	machine2

6. Configure consensus leasing.
   a. In the **Domain Structure** pane, expand **Environment** and select **Clusters** to list the configured clusters in the domain.
   b. Select *cluster1* to view its settings.
   c. Click the **Configuration > Migration** tab to display migration settings for the cluster.
   d. Click **Lock & Edit**.
   e. Find the **Migration** Basis field and set it to *Consensus*.

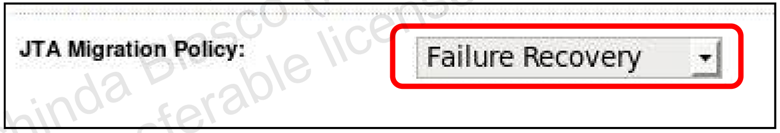


Migration Basis: Database

Note: By setting the cluster to consensus leasing, you allow the cluster migration mechanism to manage cluster leasing in-memory. This requires using the Node Manager for migration, but does not require configuring a database table for database leasing.

   f. Save your changes.

7. Enable automatic JTA migration on each server.
   a. In the **Domain Structure** pane, expand **Environment** and select **Servers** to list the configured servers in the domain.
   b. Select `server1` to view its settings.
   c. Click the **Configuration > Migration** tab to display migration settings for the server.
   d. Find the *JTA Migration Configuration* section and set the **JTA Migration Policy** to *Failure Recovery*.



JTA Migration Policy: Failure Recovery

   e. Review the *JTA Candidate Servers* field. It shows a list of Available and Chosen servers. This field is optional. If no servers in the cluster are set as Chosen, all servers in the cluster are candidates for migration. This setting is useful for scenarios where some server members in a cluster have access to a shared persistent store while other server members may not. Because all servers in your domain will have access to the shared persistent store, you do not configure any candidate servers to let it default to all servers.
   f. Save your changes.
   g. Repeat these steps again for `server2`.

8. Configure a new default persistent store for each server.
   a. In the Domain Structure panel, expand **Environment** and select **Servers > server1**.
   b. Click the **Configuration > Services** tab to display the settings for this server's services. **Do not select the top-level Services tab because it is for other purposes.**
   c. Find the *Default Store* section and the **Directory** field in that section. Set the directory to `/practices/part2/practice12-01`. As you know, the `/practices` folder is on an NFS-shared file system that is accessible by host01 and host02.
   d. Find the *Transaction Log Store* section and the *Type* field in that section. Note that it is set to *DefaultStore*.

e. Save your changes.

f. Repeat these steps for server2.

g. Activate your changes.

9. Restart domain and test migration.

   The configuration changes you made require restarting all the servers. You must also start the servers using the Node Manager for consensus leasing to work.

   a. Press **Ctrl + C** in all terminal windows that are running servers to stop the servers.

   b. Close the `AdminServer` terminal window but keep the `server1` and `server2` terminal windows open because you will reuse them for the remainder of the practice. You may want to increase the size of these terminal windows to make it easier to read log messages.

   c. Perform the following commands in the `server1` terminal window to monitor its log file:

```
$ cd /u01/domains/part2/wlsadmin/servers/server1/logs
$ tail -f server1.log
```

   d. Perform the following commands in the `server2` terminal window to monitor its log file:

```
$ cd /u01/domains/part2/wlsadmin/servers/server2/logs
$ tail -f server2.log
```

   e. Perform the following command in the main terminal window **on host01** to start the domain using the Node Manager:

```
$ wlst.sh startDomain.py
```

   f. Observe the `server1` and `server2` terminal windows for log messages as the servers start. You should look for a line similar to the following to confirm that your configuration is working **(after the server is RUNNING message)**:

   Server1: `<BEA-000189> <The Singleton Service server1 is now active on this server...>`

   Server2: `<BEA-000189> <The Singleton Service server2 is now active on this server. ...>`

   g. Press the **Enter** key in both the `server1` and `server2` terminal windows repeatedly in order to create a lot of whitespace between the start up log messages and new log messages that will appear during migration. Ideally, this will clear the screen of all log messages in each window.

   h. Confirm that the two default file stores containing the transaction logs for `server1` and `server2` are created in the practice folder. They are named as follows and should be found in the practice directory:

```
_WLS_SERVER1000000.DAT
_WLS_SERVER2000000.DAT
```

i. Perform the following command to get the process ID of the `server1` process:

```
$ ps -ef | grep server1
```

```
oracle   12600 12532  4 22:28 ?        00:01:44
/u01/app/fmw/jdk/bin/java -server -Xms256m -Xmx512m -
XX:MaxPermSize=256m -Dweblogic.Name=server1 -
Djava.security.policy=/u01/app/fmw/wlserver/server/lib/weblogic.
policy -Dweblogic.ProductionModeEnabled=true -
Dweblogic.system.BootIdentityFile=/u01/domains/part2/wlsadmin/se
rvers/server1/data/nodemanager/boot.properties -
Dweblogic.nodemanager.ServiceEnabled=true -
Dweblogic.security.SSL.ignoreHostnameVerification=false -
Dweblogic.ReverseDNSAllowed=false -Xverify:none -
Djava.endorsed.dirs=/u01/app/fmw/jdk/jre/lib/endorsed:/u01/app/f
mw/oracle_common/modules/endorsed -da -
Dwls.home=/u01/app/fmw/wlserver/server -
Dweblogic.home=/u01/app/fmw/wlserver/server -
Dweblogic.management.server=http://host01:7001 weblogic.Server
```

j. Perform the following command to kill server1 to trigger the automatic migration of its JTA service:

```
$ kill -9 12600
```

k. Quickly look in the server2 terminal window and you should see messages similar to the following, which show that server1's JTA service has migrated to server2:
Note: If you miss the messages consider using the grep or less commands to find the message numbers, shown in bold below.

```
<BEA-280008> <Opening the persistent file store "_WLS_server1"
for recovery: directory=/practices/part2/practice12-01
requestedWritePolicy="Direct-Write" fileLockingEnabled=true
driver="wlfileio3".>
```

```
####<Apr 29, 2016 11:08:35 PM UTC> <Info> <Store>
<host02.example.com> <server2> <[ACTIVE] ExecuteThread: '12' for
queue: 'weblogic.kernel.Default (self-tuning)'> <<WLS Kernel>>
<> <> <1367276915018> <BEA-280009> <The persistent file store
"_WLS_server1" (77b28bcf-2df4-4d6c-97ed-12354385bdc8) has been
opened: blockSize=512 actualWritePolicy="Direct-Write(read-
buffered)" explicitIOEnforced=false records=13.>
```

```
####<Apr 29, 2016 11:08:35 PM UTC> <Info> <Cluster>
<host02.example.com> <server2> <[ACTIVE] ExecuteThread: '12' for
queue: 'weblogic.kernel.Default (self-tuning)'> <<WLS Kernel>>
<> <> <1367276915359> <BEA-000189> <The Singleton Service
server1 is now active on this server.>
```

l. After server1 starts again, or perhaps even before, you should see the service fail back to server1. Check both log files for messages similar to the following:

Server1: `<BEA-000189> <The Singleton Service server1 is now active on this server.>`

Server2: `<BEA-000190> <The Singleton Service server1 has been deactivated on this server.>`

Note

- Because the log files may be verbose, you may have better results if you use gedit to open the log files and search for the text directly.
- You can ignore the messages regarding variables not being serializable as long as the application works.
- If migration failback fails and server1 will not start again, you have experienced an issue with using the consensus leasing feature with too few servers running, which is ok. You can perform the following tasks to manually migrate the service back:
  - Use the administration console to shut down server1 to clear its current state.
  - Start server1 again.

Congratulations! You have successfully configured automatic JTA migration.

10. Shut down the environment.

a. Perform the following command in the main terminal window to shut down the domain:

```
$ wlst.sh stopDomain.py
```

b. Close all terminal windows used for each server.

Practice Solution

1. Open a new terminal window **on each machine**. Set the title of the terminal window by selecting Terminal > Set Title and entering `server1log` **on host01** and `server2log` **on host02** as the title. This makes it easier to distinguish the purpose of each window.
2. Change the current directory to the current practice folder.
3. Execute the solution script:

   ```
   $ ./solution.sh
   ```

   This script performs the following:

   - Ensures that no previous servers are running on both machines
   - Creates the solution domain for this practice
   - Starts the Node Managers on host01 and host02
4. Continue starting at step number 9e. Once the domain has started then perform the tasks from step 9c and 9d (tail of server log files) and continue.

Practices for Lesson 13: Domain Partitions

Chapter 13

Practices for Lesson 13: Overview

Practices Overview

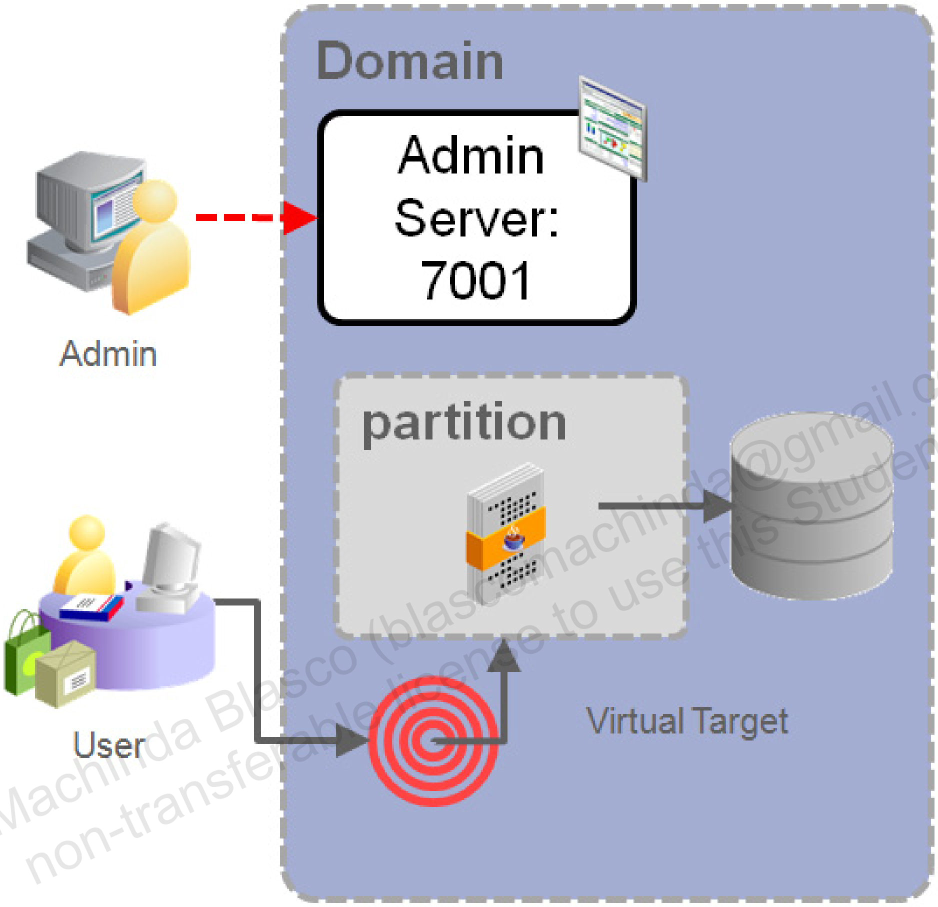
WebLogic Server provides the concepts of Domain Partitions. Domain partitions allow various artifacts such as servers, clusters, applications and other JEE components to share an environment. Such shared environments provide considerable benefits such as maximizing resource use, minimizing configuration overhead and so on.

In this practice, you will learn how to create virtual targets, create partitions, create related components, and deploy applications into a single partition environment.

Practice 13-1: Creating and managing Domain Partitions

Overview

This practice shows you how to create, configure, and deploy applications to WebLogic partitions.



This practice involves the following:

1. Creating a virtual target
2. Creating a domain partition and assigning a virtual target
3. Deploying and testing an application against a domain partition.

Tasks

1. Connect to the **host01** and **host02** machines.
2. Set up the practice environment.
   a. You may reuse the MAIN terminal windows from previous practices. Otherwise, open a terminal window **on each machine** by clicking the terminal icon in the launch panel located on the top of the screen.
   b. Set the title of each terminal window by selecting Terminal > Set Title and entering `MAIN` as the title. This makes it easier to distinguish the purpose of each window.
   c. Within the terminal window **on host01**, navigate to the `practice06-01` folder and execute the `setup.sh` script to reset the initial practice environment:

```
$ cd /practices/part2/practice13-01
$ ./setup.sh
```

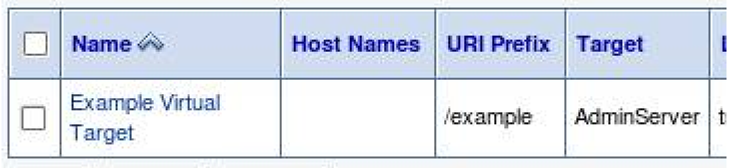
   This script performs the following tasks:

   – Ensures that no previous servers are running on both machines

   – Restores any domains and the practice to their original state

   – Starts an administration server instance

3. Configure a new virtual target:
   a. Log in to the administration console.
   `http://host01:7001/console`
   b. In the Domain Structure pane, under the *wlsadmin* domain, expand **Environment**.
   c. Select **Virtual Targets**.
   d. In Change Center, click **Lock & Edit.**
   e. In the *Summary of Virtual Targets* pane, click **New**.
   f. In the *Create a New Virtual Target* pane, enter:
   Name: `Example Virtual Target`
   Target: `AdminServer`
   URI Prefix `/example`
   Leave all other fields unchanged.
   g. Click **OK**.
   The result should resemble :

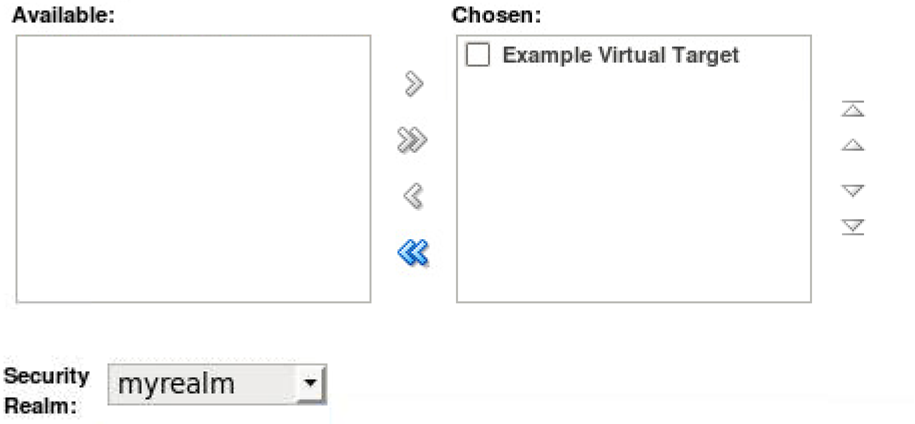


□	Name	Host Names	URI Prefix	Target
□	Example Virtual Target		/example	AdminServer

   h.
   i. In Change Center, click **Activate Changes**
4. Configure a new domain partition:
   a. In Change Center, click **Lock & Edit.**
   b. In the Domain Structure pane, under the **wlsadmin** domain, click **Domain Partitions.**
   c. In the *Summary of Domain Partitions* pane, click **New**.
   d. Name the partition `ExampleDP` and click **Next**.
   e. From the Virtual Targets list, select **Example Virtual Target** and click **Next**.
   f. From the available list, select the `Example Virtual Target.`

g. Use the blue right arrow to move the virtual target from the Available list to the Chosen list.

h. From the *Security Realm* drop-down list, select **myrealm**.
The result should resemble:



Available:

Chosen:

□ Example Virtual Target

Security Realm: myrealm

i. Leave all other fields unchanged and click **Finish**.

j. Click **Activate Changes**.

5. Restart the Administration Server:

a. On Host01, Click the [X] in the Administration Server window, or select the window and press [ctrl] + [c] to shut down the administration Server.

b. In the Host01 main terminal window, enter

```
$ startAdmin.sh
```

c. Wait for the Administration Server to restart.

6. Start the Domain Partition.

a. Log in to the administration console.

```
http://host01:7001/console
```

b. In the Domain Structure pane, under the *wlsadmin* domain, click **Domain Partitions**.

c. In the *Summary of Domain Partitions* pane, click the **Control** tab.

d. Select the newly created **exampleDP** domain partition and click **Start**.

e. In the confirmation pane, click **Yes**.

f. Use the refresh button (↻) to refresh the pane until the partition shows state RUNNING.

7. Deploy an application to the domain partition.

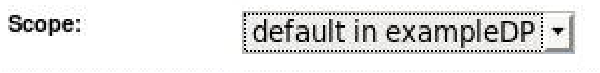
a. In the Domain Structure pane, select **Deployments**.

b. In Change Center, click **Lock & Edit**.

c. In the Summary of Deployments pane, click **Install**.

d. In the Install application wizard, enter a path of */practices/part2/apps/ShoppingCart.war* and click **Next**. .

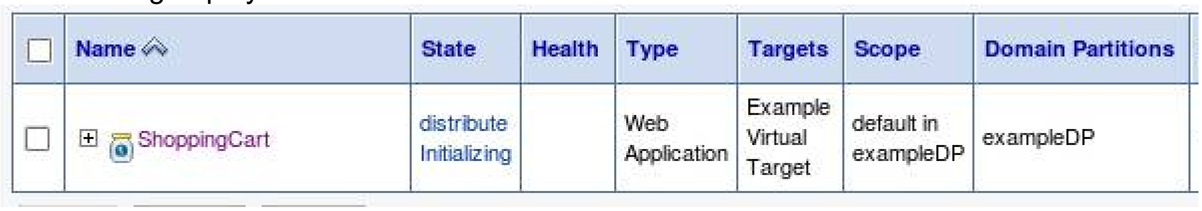
e. In the scope section, from the drop-down list, select **default in exampleDP**



Scope: default in exampleDP

f. Click **Next.**

g. Click **Finish.**
The resulting deployment should resemble:



□	Name	State	Health	Type	Targets	Scope	Domain Partitions
□	⊞ ShoppingCart	distribute Initializing		Web Application	Example Virtual Target	default in exampleDP	exampleDP

h. In Change Center, click **Activate Changes**.

i. In the Summary of Deployments pane, select the **Control** tab.

j. Select the newly deployed shopping cart application and click **Start > Start serving all requests**.

k. In the confirm pane, click **Yes**.

8. Test the application.

   a. In Firefox, create a new tab.

   b. Enter URL
   `http://host01:7001/example/ShoppingCart`

   c. The shopping cart application should display, running under the domain partition.

9 Shut down the environment.

   a. Perform the following command in the main terminal window to shut down the domain:

```
$ /practices/part2/bin/stopDomain.sh
```

   b. Close all terminal windows used for each server.

This completes the exercise.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Open a new terminal window on **host01**.
2. Change the current directory to the current practice folder.
3. Execute the solution script:

   ```
   $ ./solution.sh
   ```

   The solution script performs the following:
   - Resets the environment and starts the Administration Server
   - Creates a virtual target
   - Creates a domain partition
   - Deploys the shopping cart application to the domain
   - Restarts the administration server and the domain partition
4. Wait for Administration server to fully restart.
5. Test the application by opening a browser and entering URL **http://host01:7001/example/ShoppingCart**.

Practices for Lesson 14: Managing Data Sources

Chapter 14

Practices for Lesson 14: Overview

Practices Overview

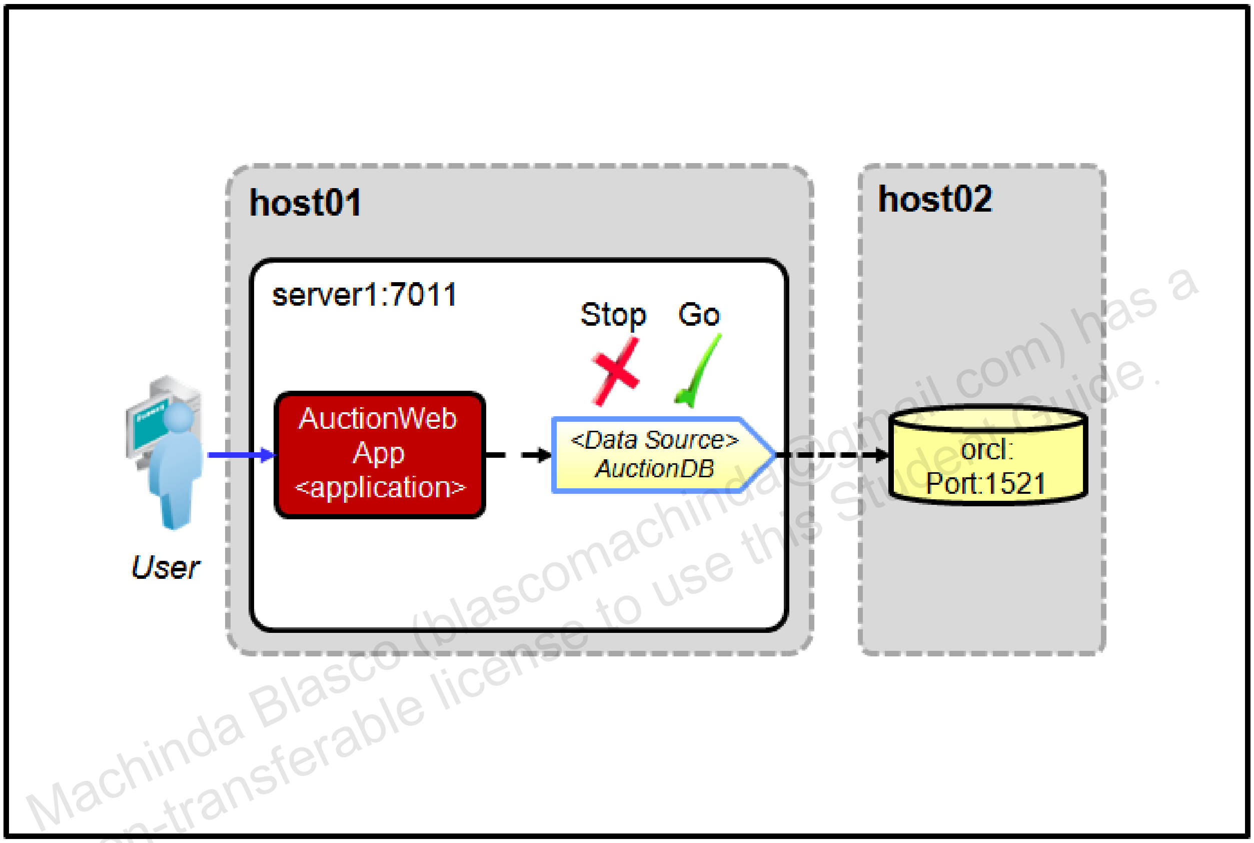
In the practices for this lesson, you use the administration console to control the `jdbc.AuctionDB` data source and update your existing WebLogic domain to support database clustering.

Practice 14-1: Controlling a Data Source

Overview

WebLogic data sources provide controls for starting, stopping, and running in administration mode. This practice shows you how to start and stop the `AuctionDBDataSource` data source.

The following image shows the architecture for controlling the data source used with the Auction application:



This image depicts the architecture of the environment for this practice:

1. The `orcl` database instance is running on port `1521` on host02.
2. The Auction application servers are running on host01.
3. You stop and start the data source called `AuctionDBDataSource` that is associated with the `orcl` database to see how it affects the application's functionality.

Tasks

1. Connect to the **host01** machine.
2. Set up the practice environment.
   a. You can reuse the MAIN terminal window from previous practices. Otherwise, open a terminal window **on host01** by clicking the terminal icon in the launch panel located at the top of the screen.
   b. Set the title of the terminal window by selecting Terminal > Set Title and entering MAIN as the title. This makes it easier to distinguish the purpose of each window.
   c. In the terminal window **on host01**, navigate to the `practice14-01` folder.

```
$ cd /practices/part2/practice14-01
```

d. Execute the `setup.sh` script to set up the initial practice environment:

```
$ ./setup.sh
```

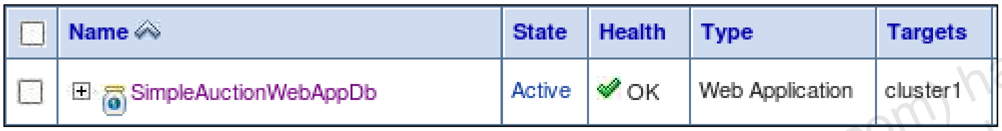
This script performs the following:

- Ensures that no previous servers are running on both machines
- Restores the domain and practice to their original state
- Starts the `wlsadmin` domain on host01 (host02 is not used in this practice.)
- Ensures that no applications or libraries are deployed
- Deploys the starting application used for this practice

3. Verify the domain's configuration.

   a. Launch a web browser and log in to the administration console:
   `http://host01:7001/console`

   b. Verify that the `SimpleAuctionWebAppDb` application is deployed.



□	Name	State	Health	Type	Targets
□	⊞ SimpleAuctionWebAppDb	Active	✓ OK	Web Application	cluster1

4. Test the `SimpleAuctionWebAppDb` application using the current web browser to ensure that the application is working as expected:

   a. Browse to `http://host01:7011/SimpleAuctionWebAppDb`. You should see the Auction application appear.

   b. If you have already created the database, you should be able to click *View Auction List* link and successfully get a list of auctions returned. This verifies that the `orcl` database is started and functioning properly.

5. Stop the data source and test the application.

   a. In the Domain Structure panel, **select Services > Data Sources > AuctionDBDataSource**.

   b. Click the **Control** tab.

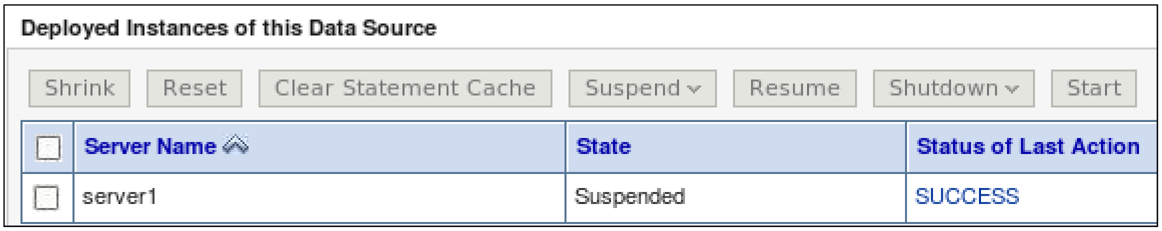
   c. Select the `server1` check box to select the data source target and to enable the control buttons.

   d. You should notice that there are several different buttons available to you:

   - **Shrink:** Shrinks the number of available connection pool connections
   - **Reset:** Closes and re-creates all available database connections for the data source
   - **Clear Statement Cache:** Clears the cached statements in memory used to improve performance
   - **Suspend:** Marks the data source as disabled and keeps reserved connections allocated. If a forced suspend is performed, all connections are removed and clients must reserve a new connection to continue.
   - **Resume:** Resumes a previously suspended data source
   - **Shutdown:** Closes all database connections in the data source and disables it
   - **Start:** Reinitializes the data source for use

   e. Select **Suspend > Suspend** to disable the data source.

   f. Confirm **Yes**.



□	Server Name	State	Status of Last Action
□	server1	Suspended	SUCCESS

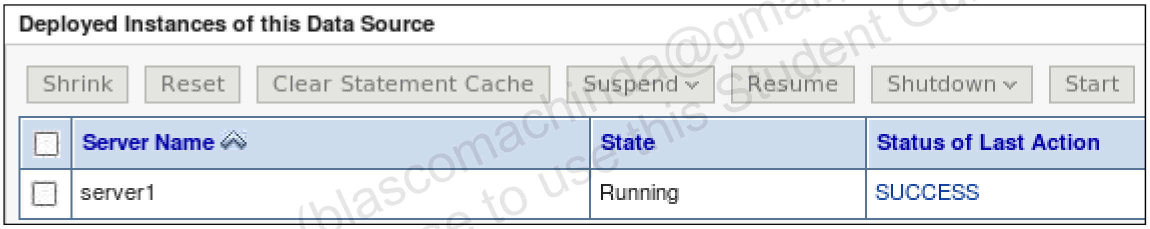
g. Test the application again by trying to view the list of auctions. You should get a 500 Internal Server Error, which causes WebLogic to display the configured System Error page. After a thorough examination of the console log for `server1`, you should see an exception similar to the following:

```
Caused by: java.sql.SQLException:
weblogic.common.resourcepool.ResourceDisabledException: Pool
AuctionDBDataSource is Suspended, cannot allocate resources to
applications
```

6. Start the data source and test the application.

a. Select the check box next to `server1` and click **Resume** to start the data source again.

b. Click **Yes** to confirm.



□	Server Name	State	Status of Last Action
□	server1	Running	SUCCESS

c. Test the application again by trying to view the list of auctions. You should once again see a list of auction records.

Congratulations! You have successfully controlled a data source. Feel free to play around with some of the other controls for the data source.

7. Shut down the environment.

a. Press **Ctrl + C** in all terminal windows that are running servers.

b. Close all terminal windows used for each server.

Practice Solution

There is no solution for this practice.

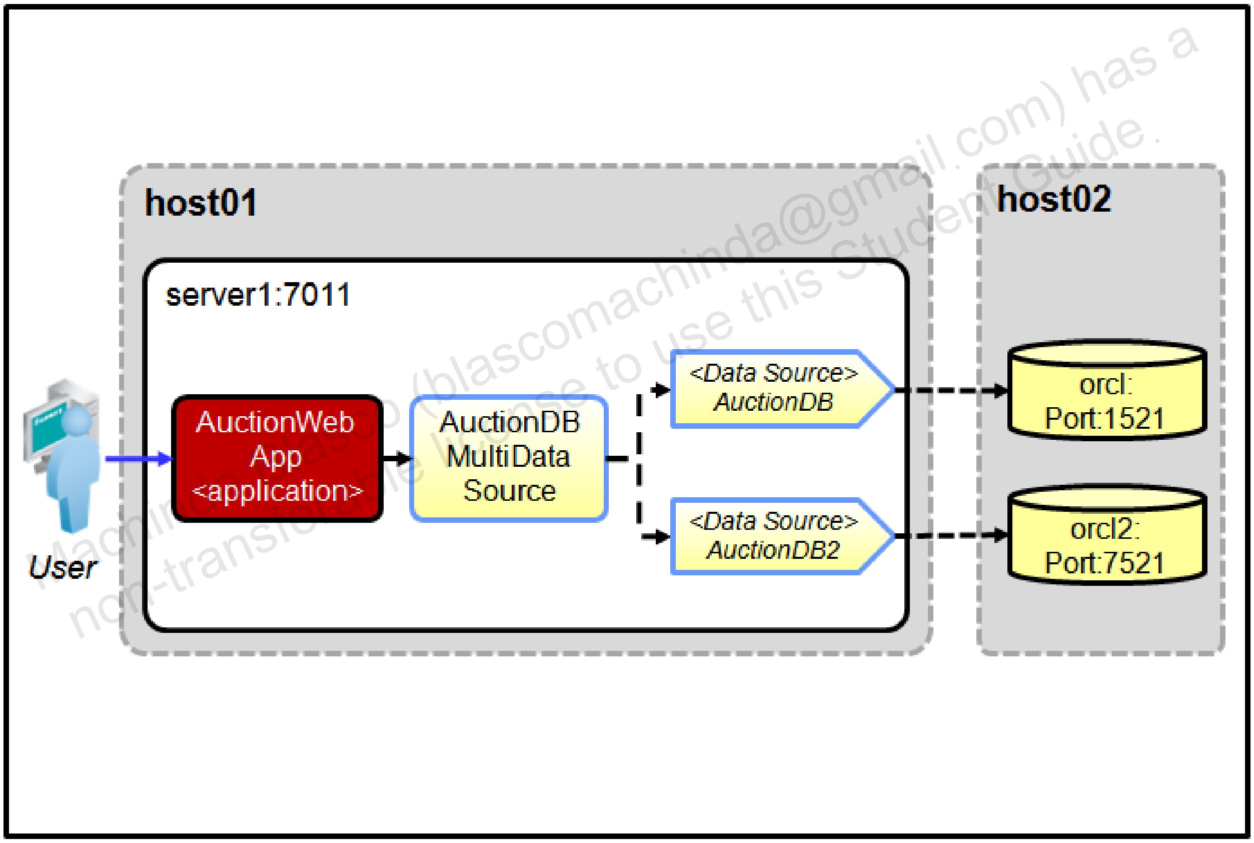
Practice 14-2: (optional) Creating and Using a Multi Data Source

Overview

Using WebLogic multi data sources, applications transparently connect to a grid of redundant database instances. Multi data sources are used to distribute database connection requests across these instances, or they can simply provide failover when one instance becomes unavailable.

The Auction application environment has been upgraded to include a backup database instance to provide a higher level of availability. If the primary database becomes unavailable and the Auction application requests a database connection, WebLogic will begin using the backup instance. The backup database is not synchronized with the primary database for the purposes of this practice to make it easier to see that failover has occurred.

The following image shows the architecture for multi data source use with the Auction application:



This image depicts the architecture of the environment for this practice:

1. There are two database instances: the existing `orcl` database running on port `1521` and the newly introduced `orcl2` database running on port `7521`.
2. You start both databases and the Auction application WebLogic servers on host01.
3. You configure a new data source called `AuctionDBDataSource2` that is associated with the `orcl2` database.
4. You configure a new multi data source called `AuctionDBMultiDataSource` that primarily uses the `AuctionDBDataSource` data source and fails over to the `AuctionDBDataSource2` data source.
5. You modify the application's data source from `jdbc.AuctionDB` to use the multi data source and redeploy the application.

6. You run the application, shut down the `orcl` database, and run the application again to see the application fail over to the `orcl2` database.
7. You shut down the application and reset the environment so only the `orcl` database is running.

Tasks

1. Connect to the **host01** and **host02** machines.
2. Set up the practice environment.
   a. You can reuse the MAIN terminal window from previous practices. Otherwise, open a terminal window **on host01** by clicking the terminal icon in the launch panel located at the top of the screen.
   b. Set the title of the terminal window by selecting Terminal > Set Title and entering `MAIN` as the title. This makes it easier to distinguish the purpose of each window.
   c. Within the terminal window **on host01**, navigate to the `practice14-02` folder.

```
$ cd /practices/part2/practice14-02
```

   d. Execute the `setup.sh` script to set up the initial practice environment:

```
$ ./setup.sh
```

   This script performs the following:

   - Ensures that no previous servers are running on both machines
   - Restores the domain and practice to their original state
   - Starts the `wlsadmin` domain on host01 (server2 is not used in this practice.)
   - Ensures that no applications or libraries are deployed
   - Deploys the starting application used for this practice
3. Start and create the backup database.
   a. Run the following scripts to start the `orcl2` database instance **on host02**:

```
$ startDB2.sh
```

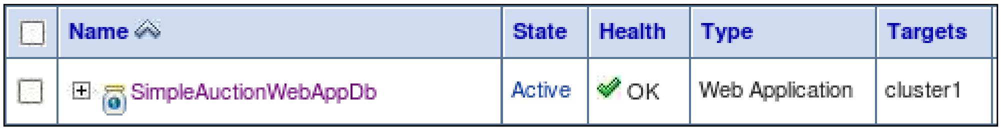
   b. Wait for the database to start.
   c. Open a new terminal window and execute the following script to create the `oracle` user and Auction tables:

```
$ createDatabase2.sh
```

   Note: This script is the same as the script that creates the tables for the Auction application on the `orcl` database instance. The only difference is that it calls a SQL script to create the `oracle` user and exports `ORACLE_SID=orcl2` to cause all database commands to use the backup `orcl2` database instance.
4. Verify the domain's configuration.
   a. Launch a web browser and log in to the administration console:

   `http://host01:7001/console`

   b. Verify that the `SimpleAuctionWebAppDb` application is deployed.



□	Name	State	Health	Type	Targets
□	⊞ SimpleAuctionWebAppDb	Active	✓ OK	Web Application	cluster1

5. Test the `SimpleAuctionWebAppDb` application by using the current web browser to ensure that the application is working as expected:
   a. Browse to `http://host01:7011/SimpleAuctionWebAppDb`. You should see the Auction application appear.
   b. If you have already created the database, you should be able to click the *View Auction List* link and successfully get a list of auctions returned. This verifies that the `orcl` database is started and functioning properly.

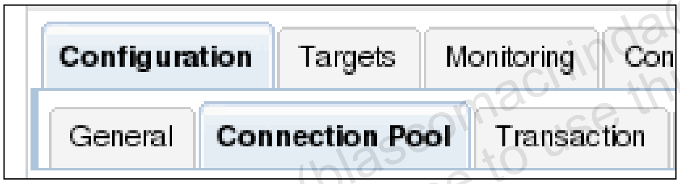
6. Configure connection testing on the original data source.

   Why? Connection testing is a feature that tests getting a JDBC connection pool connection from a data source prior to getting and using the connection. This causes a slight delay for an application for the test but allows WebLogic to fail over from the unavailable data source to a working data source. You are configuring this on the original data source, so WebLogic can detect it is unavailable and fail over to the backup data source.

   a. Execute the following command to configure the data source to test connections on reserve.

```
$ wlst.sh createTestTable.py
```

   b. In the Domain Structure panel, select **Services > Data Sources**.
   c. Click AuctionDBDataSource to edit its configuration.
   d. Click the **Configuration > Connection Pool** tab:



   e. Click the **Advanced** options.
   f. Verify that the following values are set:

Field	Value
Test Connections on Reserve	<selected>
Test Table Name	`SQL SELECT 1 FROM DUAL`

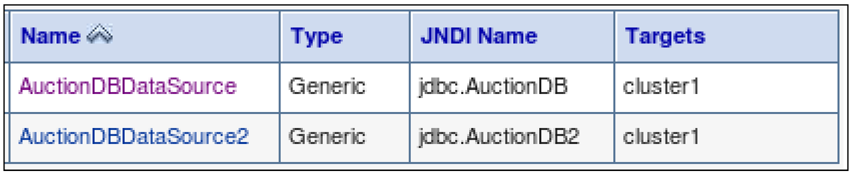
7. Create a new backup data source:
   a. Execute the following script **on host01** to create the backup data source:

```
$ wlst.sh createDataSource2.py
```

   b. Confirm that the script executed successfully:

```
Data Source created successfully.
```

   c. Return to the console and inspect the new data source:



Name	Type	JNDI Name	Targets
AuctionDBDataSource	Generic	jdbc.AuctionDB	cluster1
AuctionDBDataSource2	Generic	jdbc.AuctionDB2	cluster1

8. Create the multi data source.
   a. Click **Lock & Edit**.
   b. In the Domain Structure panel, navigate to **Services > Data Sources**.
   c. Click **New > Multi Data Source**.

d. Enter the following values:

Field	Value
Name	`AuctionDBMultiDataSource`
JNDI Name	`AuctionDBMultiDataSource`
Algorithm Type	Failover

e. Click **Next**.

f. Target the resource to `cluster1` and click **Next**.

g. Select the *XA Driver* option and click **Next**.

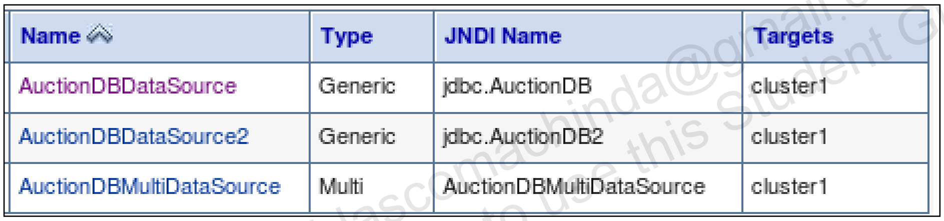
h. Use the » button to add the following data sources to the Chosen list:

- AuctionDBDataSource (Ensure that it is listed first.)
- AuctionDBDataSource2

i. Click **Finish**.

j. Activate your changes.

k. Verify that the multi data source was created in the console:



Name	Type	JNDI Name	Targets
AuctionDBDataSource	Generic	jdbc.AuctionDB	cluster1
AuctionDBDataSource2	Generic	jdbc.AuctionDB2	cluster1
AuctionDBMultiDataSource	Multi	AuctionDBMultiDataSource	cluster1

9. Modify and redeploy the `Auction` application to use your new multi data source.

Why? The Auction application uses the Java Persistence API (JPA) to provide the database model for the application. The data source used by JPA is configured in the `persistence.xml` file packaged with the application. This data source currently points to the AuctionDBDataSource data source.

a. Perform the following commands to open the `persistence.xml` file for editing:

```
$ cd /practices/part2/practice14-02/resources/SimpleAuctionWebAppDb/WEB-INF/classes/META-INF
$ gedit persistence.xml
```

b. Change the following line:

```
<jta-data-source>jdbc.AuctionDB</jta-data-source>
```

to

```
<jta-data-source>AuctionDBMultiDataSource</jta-data-source>
```

c. Save the file.

d. Perform the following commands to redeploy the application with the updated configuration:

```
$ cd /practices/part2/practice14-02
$ ./redeploy.sh
```

e. Verify that the application is now configured to use the multi data source by navigating to **Deployments > SimpleAuctionWebAppDb > Configuration > Persistence > AuctionPU > Data Sources**. View the *Jta Data Source JNDI Name* field and verify that it is set to `AuctionDBMultiDataSource`.

10. Test the application to ensure that it is working:

a. If you shut server1 down previously, you must start it again now **on host01**:

```
$ startServer1.sh
```

b. Browse to the following URL and click the *View Auction List* link. You should see a list of auctions. This list should have been retrieved from the original database.

– `http://host01:7011/SimpleAuctionWebAppDb`

c. Return to the administration console.

d. In the Domain Structure panel, select Services > Data Sources.

e. Click the Monitoring tab and scroll to the right. You should see that the *AuctionDBDataSource* data source has a positive *Active Connections High Count* figure.

11. Test data source failover.

a. Perform the following steps to stop the original `orcl` database instance **on host02**:

```
$ stopDB1.sh
```

b. Wait for the database to completely shut down.

c. Wait for about 10-20 seconds (maybe longer) until you see the following message in the console for `server1`.

Why? A data source is configured by default to trust an existing JDBC connection for 10 seconds. During this time period, the application may fail because the failover has not occurred yet.

```
<Mar 25, 2013 4:13:29 PM UTC> <Warning> <JDBC> <BEA-001129>
<Received exception while creating connection for pool
"AuctionDBDataSource": IO Error: The Network Adapter could not
establish the connection.>
```

d. Repeat all of the instructions in step 10 again. At first, you should notice that there are no database entries. This is because the databases are not actually synched in this practice environment. This shows you that you are connected to the `orcl2` database.

Why? The application is the same, but it is now configured to use your multi data source. You just shut down the original `orcl` database instance so it is no longer available. New database requests made by the application should now failover to the `orcl2` backup database instance.

e. Create the auction data records and test again:
   - Click the *Go Home* link.
   - Click the *Create Default Data* link.
   - Click *Yes* to confirm.
   - Click the *Go Home* link.
   - Click the *View Auction List* link again and you should see database results again.

f. Within the administration console, view the data source monitoring page again to verify that there are statistics for the AuctionDBDataSource2 data source. You should also see that the first database connection is suspended, while the current connection is running.

g. Perform the following steps **on host02** to start the `orcl` database instance:

```
$ startDB1.sh
```

Congratulations! You have successfully created, configured, and used a multi data source.

12. Shut down environment.

a. Press **Ctrl + C** in all terminal windows that are running servers.

b. Close all terminal windows used for each server.

c. Perform the following steps to stop the `orcl2` database instance **on host02**:

```
$ stopDB2.sh
```

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Open a new terminal window **on host01**:
2. Change the current directory to the current practice folder.
3. Execute the solution script:

```
$ ./solution.sh
$ Enter the Oracle WebLogic password, followed by [ENTER]:

Password updated successfully
```

   The solution script performs the following:

   - Ensures that no previous servers are running on both machines
   - Restores the domain and practice to their original state, which includes shutting down the backup database
   - Starts the wlsadmin domain on host01
   - Adds JDBC connection testing parameters to the primary data source for failover purposes
   - Starts the backup database
   - Creates the backup data source
   - Creates the multi data source
   - Creates the backup database
   - Deploys the practice application that uses the multi data source

4. Wait for servers on `host01` to fully start.
5. Continue starting at step number 10.

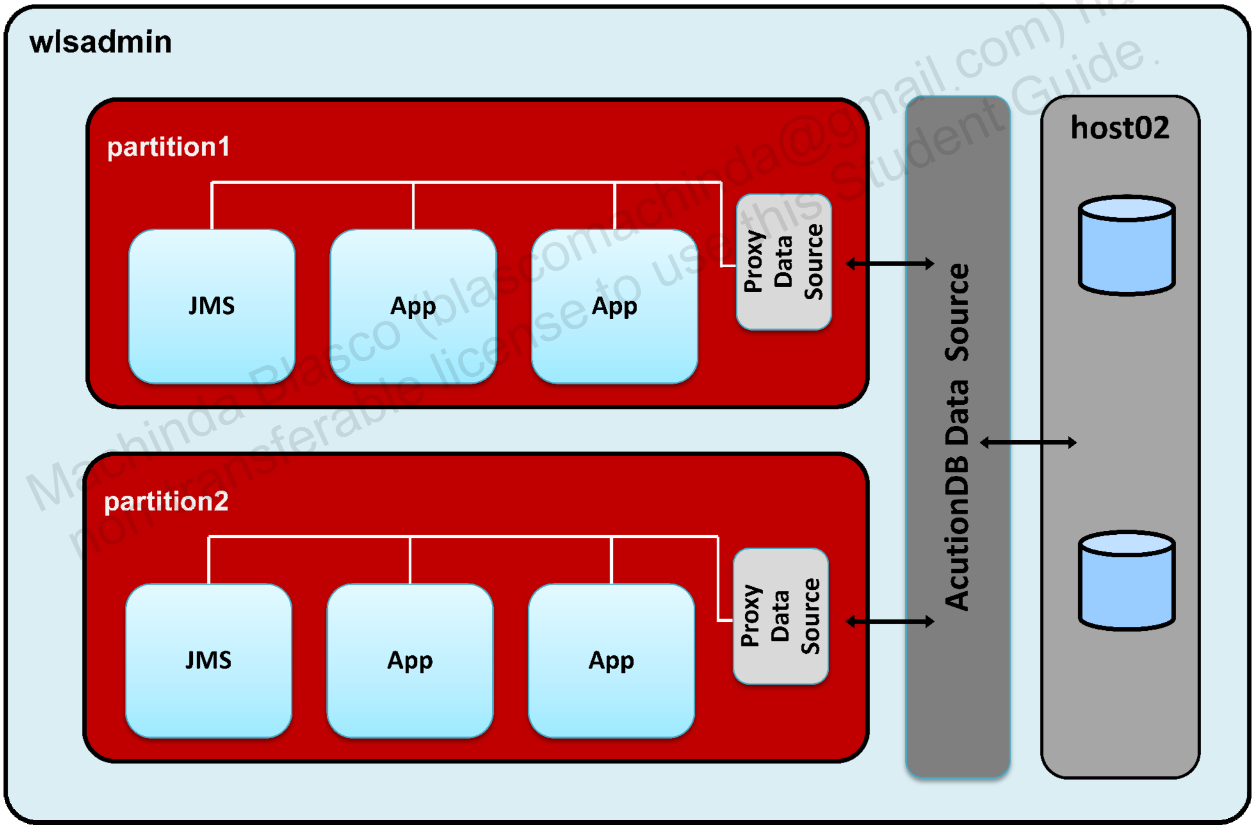
Practice 14-3: (Optional) Creating and Using Proxy Data Sources

Overview

Proxy Data Sources simplifies the administration of multiple data sources by providing a light-weight mechanism for accessing a data source associated with a partition or tenant.

The Auction application environment has been deployed into two different domain partitions, `partiton1` and `partiton2`. Originally the `Auction` application connects into a primary database using the `AuctionDB Data Source`. In partition domains is required use a Proxy Data Source to use JDBC resources that are out of the partition scope.

The following image shows the architecture for multi data source use with the Auction application:



Tasks

1. Connect to the **host01** machine.
2. Set up the practice environment.
   You can reuse the MAIN terminal window from previous practices. Otherwise, open a terminal window on host01 by clicking the terminal icon in the launch panel located at the top of the screen.

   a. Set the title of the terminal window by selecting Terminal > Set Title and entering MAIN as the title. This makes it easier to distinguish the purpose of each window.

3. Within the terminal window on host01, navigate to the practice14-03 folder.

```
$ cd /practices/part2/practice14-03
```

4. Execute the setup.sh script to set up the initial practice environment:

```
$ ./setup.sh
```

   This script performs the following:

   a. Ensures that no previous servers are running on both machines.
   b. Restores the domain and practice to their original state.
   c. Starts the **wlsadmin** admin server on host01 (server1 & 2 are not used in this practice.)
   d. Creates a Virtual Target (`target1`)
   e. Create a Partition (`partition1`)
   f. Creates a Data Source (`AuctionDBDataSource`)
   g. Ensures that no applications or libraries are deployed.

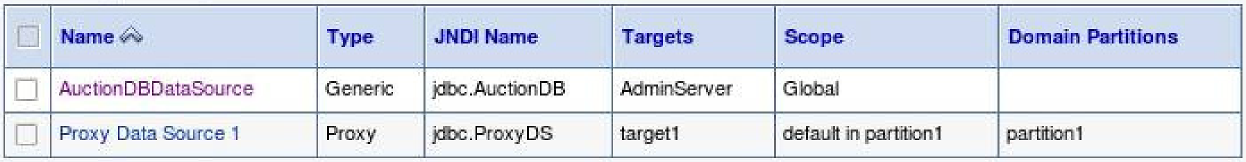
5. Start Domain Partitions
   a. Launch a web browser and log in to the administration console: `http://host01:7001/console`
   b. In the Domain Structure panel, select **Domain Partitions**.
   c. Confirm that the partition1 is in state RUNNING,

6. Create the Proxy Data Source for `partition1`
   a. In the Domain Structure panel, select **Services > Data Sources**.
   b. Click **Lock & Edit**.
   c. Click **New > Proxy Data Sources**.
   d. Enter the following values and click **Next**.

Field	Value
Name	`Proxy Data Source 1`
JNDI Name	`jdbc.ProxyDS`
Scope	`default in partition1`
Switching Properties	`default=domain:jdbc.AuctionDB`

   e. Select *AdminServer* as target and click **Finish**.

7. Verify that the `Proxy Data Source` was created in the console and activate
    a. Examine the new data source, which should resemble that shown below.



☐	Name	Type	JNDI Name	Targets	Scope	Domain Partitions
☐	AuctionDBDataSource	Generic	jdbc.AuctionDB	AdminServer	Global	
☐	Proxy Data Source 1	Proxy	jdbc.ProxyDS	target1	default in partition1	partition1

    b. In Change Center click **Activate Changes**.

8. Modify the application to use the new proxy datasource.

    a. Return to the practice 14-03 terminal on host01.

```
$ cd /practices/part2/practice14-03
```

    b. Using gedit open the application’s persistence.xml file.

```
$ gedit resources/SimpleAuctionWebAppDb/WEB-
INF/classes/META-INF/persistence.xml
```

    c. Modify the datasource name to use the newly created proxy data sources:
    Find:

```
<jta-data-source>jdbc.AuctionDB</jta-data-source>
```

    And replace **AuctionDB** with **ProxyDS** as shown below.

```
<jta-data-source>jdbc.ProxyDS</jta-data-source>
```

    d. Save the file and exit gedit.

9. Deploy the application in the `partition1`:
    a. Click on **Lock & Edit**
    b. Select **Deployments.**
    c. Click on **Install.**
    d. Select Navigate to `/practices/part2/practice14-03/resources`
    e. Select `SimpleAuctionWebAppDb` and click on **Next.**
    f. Select the scope as `default in partition1` to install the deployment and click on **Next**.
    g. Click on **Finish.**

10. Activate the application
    a. In Change Center click **Activate Changes**.
    b. In the Summary of Deployments pane select the **Control** tab.
    c. Select the newly deployed application and click **Start > Start Servicing all request**.
    d. In the confirmation dialog click **Yes**.

11. Test the application Test the `SimpleAuctionWebAppDb` application in the partition using the current web browser to ensure that the application is working as expected:
    a. Browse to **`http://host01:7001/target1/SimpleAuctionWebAppDb`**. You should see the Auction application appear.
    b. If you have already created the database, you should be able to click ***View Auction List*** link and successfully get a list of auctions returned. This verifies that the ***`proxy_ds`*** **`Proxy Data Source`** is working properly in **`partition1`**.
    c. Browse to **`http://host01:7001/target1/SimpleAuctionWebAppDb`**. **Note** that steps d and e are only required if no data was previously created
    d. Click **Create Default Data**
    e. Click **Yes, Create Default Data**
    f. Click **Go Home**.
    g. Click **View Auction List**
    h. If auctions are displayed you have successfully completed the exercise.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Open a new terminal window **on host01**:
2. Change the current directory to the current practice folder.
3. Execute the solution script:

```
$ ./solution.sh
$ Enter the Oracle WebLogic password, followed by [ENTER]:

Password updated successfully
```

   The solution script performs the following:

   a. Ensures that no previous servers are running on both machines.
   b. Restores the domain and practice to their original state.
   c. Starts the **wlsadmin** domain on host01 (server2 is not used in this practice.)
   d. Creates Virtual Target `target1`
   e. Create Partition `partition1`
   f. Creates a Data Source (`AuctionDBDataSource`)
   g. Creates a Proxy Data Source (`Proxy Data Source 1`)
   h. Ensures that no applications or libraries are deployed.
   i. Deploy the application `SimpleauctionWebAppDB` into **partition1** using an updated **persistence.xml** file specifying the proxy JDBC data source

4. Wait for Admin server on `host01` to fully start.
5. Continue starting at step number 11.

Practices for Lesson 15: Diagnostic Framework

Chapter 15

Practices for Lesson 15: Overview

Practices Overview

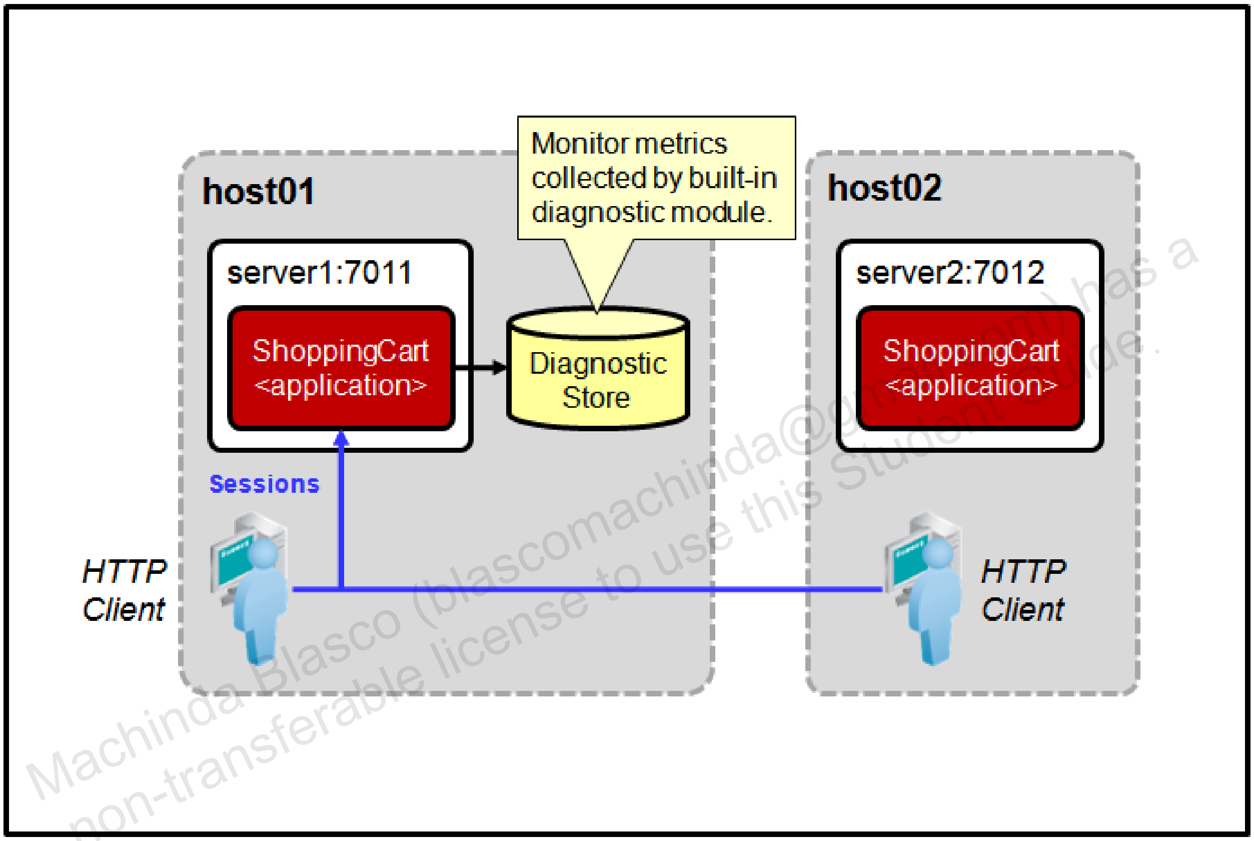
In the practices for this lesson, you use the administration console to configure WebLogic built-in diagnostic modules and metrics collected by them.

Practice 15-1: Using a Built-in Diagnostic Module

Overview

WebLogic provides built-in diagnostic modules that collect important information about the running server environment. You configure new settings for a built-in module and monitor the collected metrics. You then create a custom diagnostic module, based on a built-in module, and customize it for your environment.

The following image depicts the architecture of the environment for this practice:


host01
server1:7011
ShoppingCart <application>
Diagnostic Store
Monitor metrics collected by built-in diagnostic module.
Sessions
HTTP Client
host02
server2:7012
ShoppingCart <application>
HTTP Client

Tasks

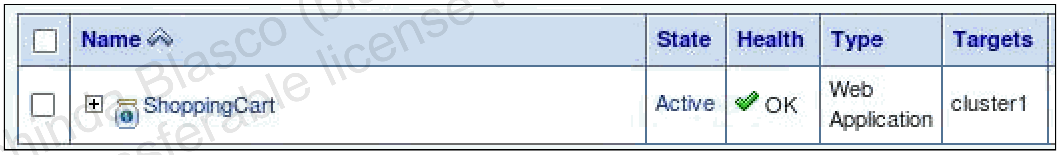
1. Connect to the **host01** machine.
2. Set up the practice environment.
   a. You can reuse the MAIN terminal window from previous practices. Otherwise, open a terminal window **on host01** by clicking the terminal icon in the launch panel located at the top of the screen.
   b. Set the title of the terminal window by selecting Terminal > Set Title and entering `MAIN` as the title. This makes it easier to distinguish the purpose of each window.
   c. In the terminal window **on host01**, navigate to the `practice15-01` folder.

   ```
   $ cd /practices/part2/practice15-01
   ```

   d. Execute the `setup.sh` script to set up the initial practice environment:

   ```
   $ ./setup.sh
   ```

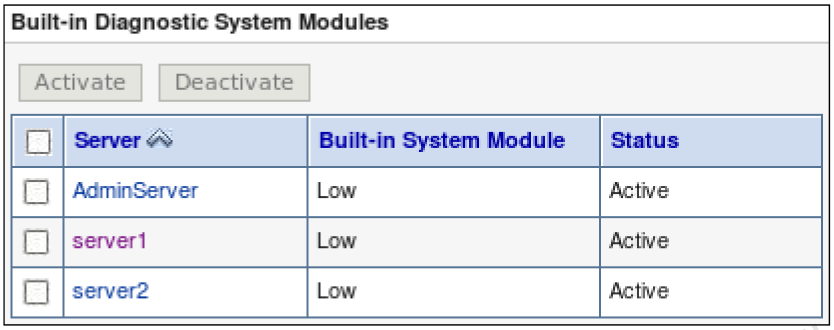
   This script performs the following:
   - Ensures that no previous servers are running on both machines
   - Restores the domain and practice to their original state
   - Starts the `wlsadmin` domain on host01 and host02
   - Deploys the starting application used for this practice
3. Verify the domain's configuration.
   a. Launch a web browser and log in to the administration console:
   `http://host01:7001/console`
   b. Verify that the `ShoppingCart` application is deployed.



□	Name	State	Health	Type	Targets
□	⊞ ShoppingCart	Active	✓ OK	Web Application	cluster1

4. Test the `ShoppingCart` application by using the current web browser to ensure that the application is working as expected:
   a. Browse to `http://host01:7011/ShoppingCart/welcome.jsp`. You should see the application appear. This shows that the application is deployed and working properly.

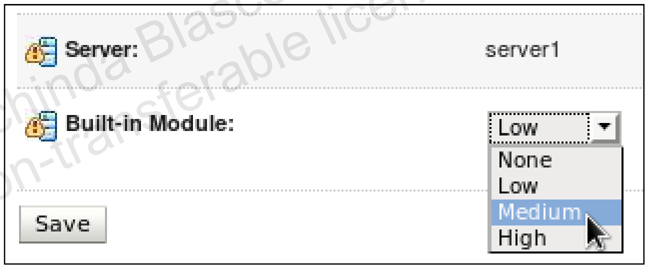
5. View the default built-in modules of the domain.
   a. In the Domain Structure panel, expand **Diagnostics** and select “**Built-in Diagnostic Modules**.”
   b. You should see a list of the configured built-in diagnostic modules for the domain. You may recall that a production mode domain enables a built-in diagnostic module for each server in the domain. This page allows you to dynamically activate or deactivate any built-in module. The image below shows that each module is currently active. You can select a check box next to any server and use the Activate or Deactivate buttons to control your modules.



Built-in Diagnostic System Modules

□	Server	Built-in System Module	Status
□	AdminServer	Low	Active
□	server1	Low	Active
□	server2	Low	Active

6. Configure a new setting for a server’s built-in module.
   a. All of the default built-in modules are set to Low. WebLogic allows for Low, Medium, or High settings for each module. Use the console to set the built-in module for `server1` from Low to Medium.
   b. Click `server1` to show the settings for its built-in module.
   c. Click **Lock & Edit**.
   d. Change its setting from Low to **Medium**.

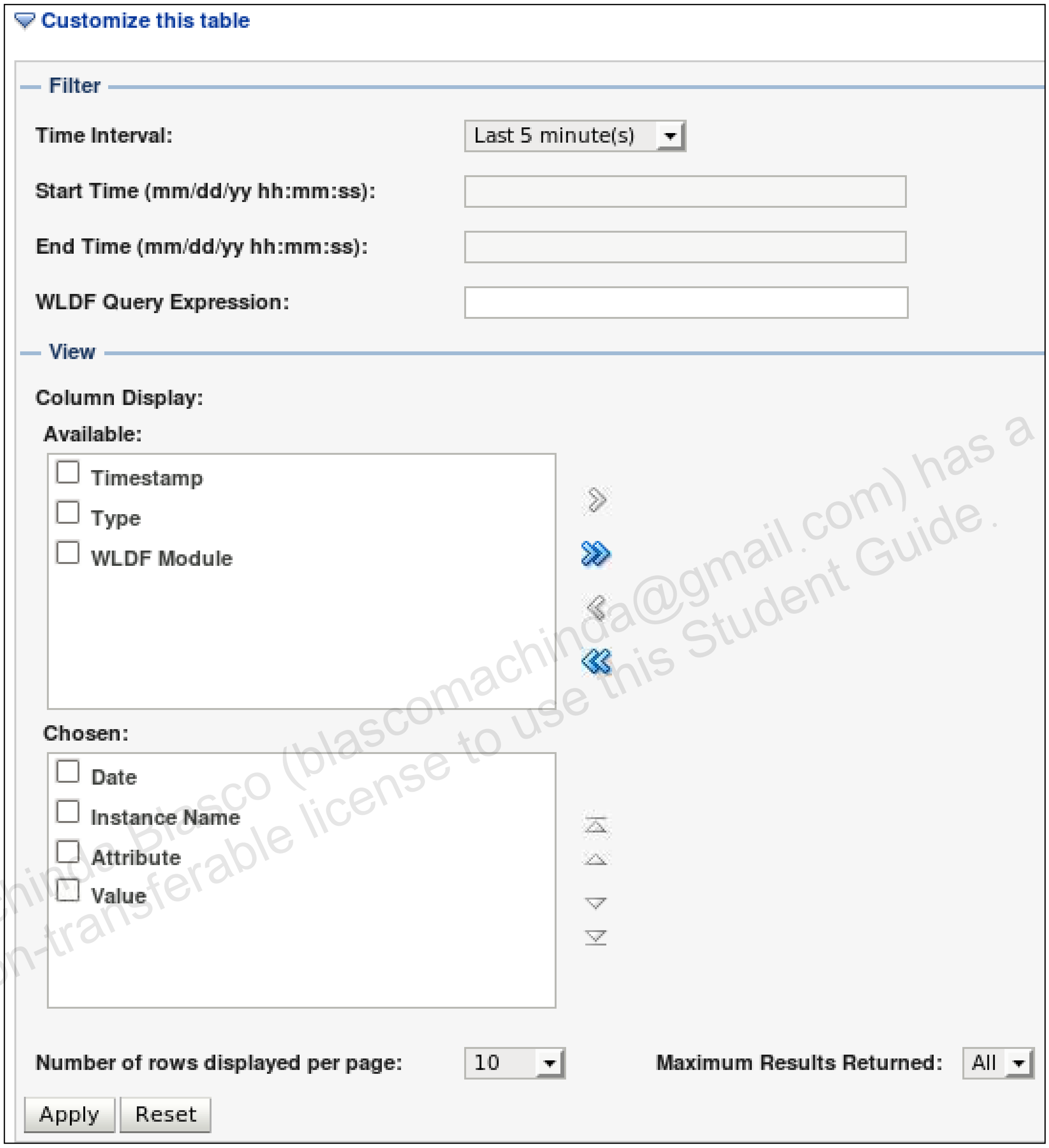


Server: server1

Built-in Module: Low

   e. Save your changes.
   f. Activate your changes.
   g. Run the application again to cause the built-in module to collect more metrics.
7. View the collected metrics with the administration console.
   a. In the **Domain Structure** pane, expand **Diagnostics** and select **Log Files**. You should see a list of the available log files for the domain. This list is populated based on which servers are currently active. If the entire domain is running, you should see several different log files for each server of the domain.
   b. Select the **HarvestedDataArchive** log for `server1` and click **View**. This displays a paginated list of the captured metrics for the server. All the metrics represent metrics collected by the built-in module because it is the only module configured for the server. Take a moment and explore the metrics that have been collected.

c. Click **Customize this table** to configure what this page displays. This shows a set of fields you can use to control which metrics to display. Your screen may be slightly different.



Customize this table

Filter

Time Interval: Last 5 minute(s)

Start Time (mm/dd/yy hh:mm:ss):

End Time (mm/dd/yy hh:mm:ss):

WLDF Query Expression:

View

Column Display:

Available:

- □ Timestamp
- □ Type
- □ WLDF Module

Chosen:

- □ Date
- □ Instance Name
- □ Attribute
- □ Value

Number of rows displayed per page: 10

Maximum Results Returned: All

The following table describes what each field setting provides:

Field	Description
Filter: Time Interval	A configurable period of time used to show only metrics collected during that period
Filter: Start Time	The start time of historic metrics to display
Filter: End Time	The end time of historic metrics to display
Filter: WLDF Query Expression	Provides a way to use the WLDF expression language to exact greater control over the displayed metrics on the page
View: Column Display: Available	Allows you to control which columns are displayed for each metric on the page. This field represents the available columns that are not currently being displayed.
View: Column Display: Chosen	Represents the columns that are currently being displayed
View: Number of rows displayed per page	Controls the number of rows to display on a single page
View: Maximum Results Returned	Controls how many total results are displayed

d. In the *Column Display:* Ensure that the *WLDF Module* column name is in the Chosen field. This column shows which module collected the metric in each displayed row.

e. Click **Apply** to realize your settings if you made any changes.

f. Review the displayed metrics table again. This time scroll to the right to view the *WLDF Module* column.
Hint: The horizontal scroll bar is at the bottom of the page. This shows you that the module that collected each metric is the *wldf-server-medium* built-in module.

8. Create a custom diagnostic module based on a built-in module.

a. In the **Domain Structure** pane, expand **Diagnostics** and select **Diagnostic Modules**.

b. Click **Lock & Edit**.

c. Click **New** to create a new diagnostic module.

d. Enter values from the following table to configure your module and click **OK**. Your module will be based on the Low built-in diagnostic module.

Field	Value
Name	MyCustomModule
Description	My new built-in based module
Use a built-in diagnostic system module as template	Checked
Built-in diagnostic system module	Low

e. Select your newly configured module to view its settings page.

f. Click the **Collected Metrics** tab to display the metrics that are collected already by your module. These are derived from the Low built-in module. You can now customize which metrics your module collects.

g. Select every entry in the *Collected Metrics in this Module* table except for the *WebAppComponentRuntimeMBean* entry and click Delete. You are removing all the unwanted metrics in order to keep this example simple.

h. Click **Yes** to confirm.

i. Save your changes.

j. Click the **Targets** tab.

k. Select `server1` (and only `server1`) as the target for this module.

l. Save your changes.

m. Activate your changes.

n. Return to and run the shopping cart application to create more metric data.

o. Return to the console to view metric entries in the console.

p. Click the *Customize this table* link again.

q. Enter the following data in the WLDF Query Expression field to limit the display to show entries only from your custom module:

```
WLDFMODULE = 'MyCustomModule'
```

r. Click **Apply** to realize your changes.

s. The page should change to display entries created by your custom module. Find the entries related to `server1` and the ShoppingCart application to view how many sessions are created on the server. The current value should be 1.

t. Run the application again from a web browser **on host02** to generate more metrics. Remember to use host01 as the host name in the URL when accessing the application from host02; otherwise, the new session is created on `server2` instead of `server1`.

u. Return to the administration console and refresh the display to view the same metric again. Because you ran the application with a new client, the value should now be changed to 2.
Hint: You need to go to the entry that corresponds with the time that you ran the application again. If not, you will still see a value of 1 for session count.

Congratulations! You have successfully worked with WebLogic built-in diagnostic modules and created a custom diagnostic module that is based on a built-in module.

9. Shut down the environment.

a. Perform the following command in any terminal window **on host01** to shut down the domain:

```
$ stopDomain.sh
```

Practice Solution

There is no solution for this practice. No practices depend on this practice.

Practices for Lesson 16: WebLogic and Coherence Integration

Chapter 16

Practices for Lesson 16: Overview

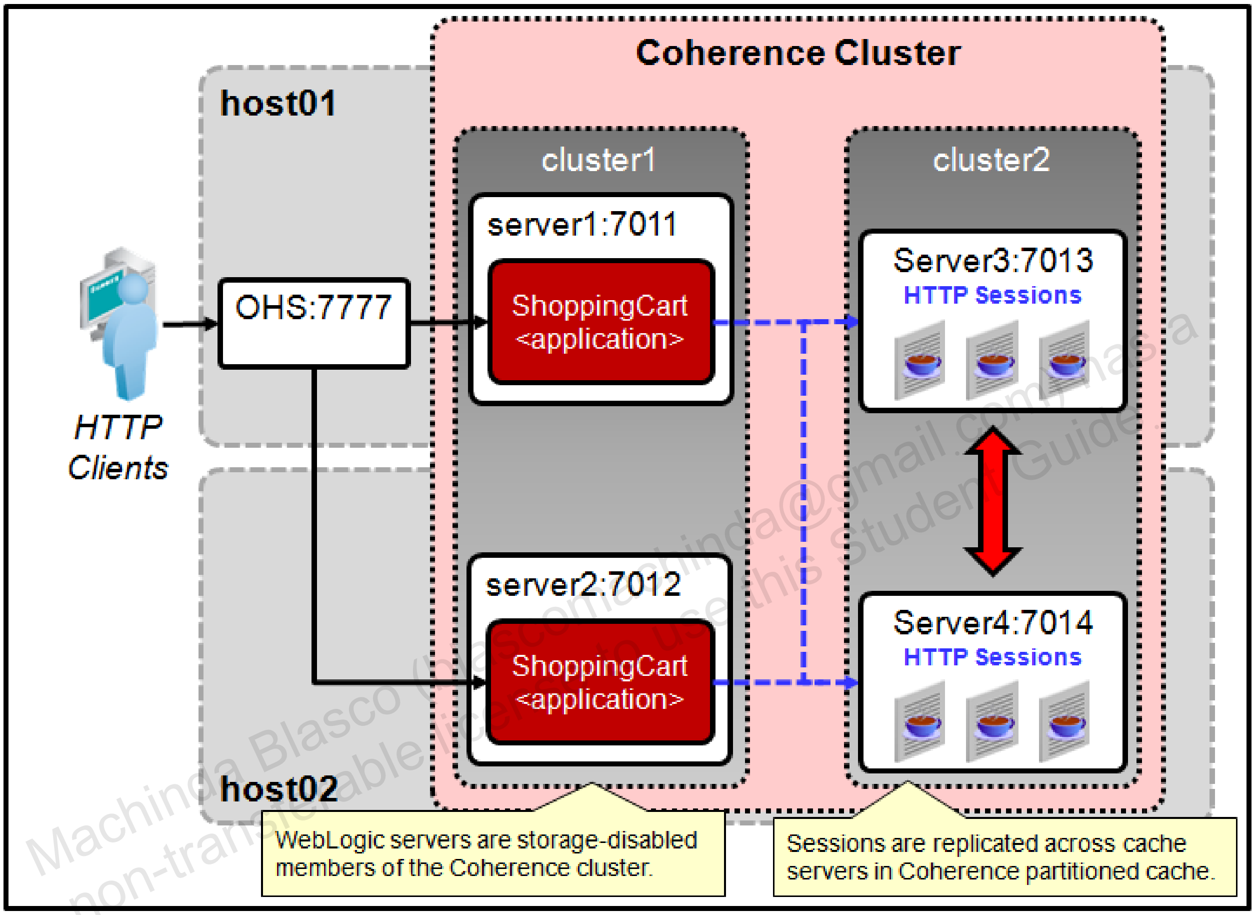
Practices Overview

In the practices for this lesson, you configure the ShoppingCart application to use Coherence*Web session replication. Then you create managed Coherence servers, deploy a grid archive to them, and run a Coherence application.

Practice 16-1: Configuring Coherence*Web

Overview

WebLogic allows you to use Coherence caching to store HTTP sessions. This practice shows you how to configure an application to use Coherence*Web session replication. You then run the application to test your configuration.


Coherence Cluster
host01
cluster1
cluster2
server1:7011
OHS:7777
ShoppingCart <application>
Server3:7013
HTTP Sessions
HTTP Clients
server2:7012
ShoppingCart <application>
Server4:7014
HTTP Sessions
host02
WebLogic servers are storage-disabled members of the Coherence cluster.
Sessions are replicated across cache servers in Coherence partitioned cache.

This image depicts the architecture of the environment for this practice:

1. The ShoppingCart application is deployed to a cluster of two servers: server1 and server2. These servers form WebLogic cluster1.
2. Server1 runs on host01 on port 7011.
3. Server2 runs on host02 on port 7012.
4. The ShoppingCart application is configured to use Coherence*Web HTTP session persistence. Sessions are stored in Coherence*Web storage-enabled servers: server3 and server4. These servers form WebLogic cluster2 and are dedicated to storing HTTP sessions with no applications deployed to them.
5. Server3 runs on host01 on port 7013.
6. Server4 runs on host02 on port 7014.
7. All managed servers are members of the Coherence cluster.
   - Cluster1 servers have **disabled** session local storage, so they **cannot** store HTTP sessions using Coherence*Web.
   - Cluster2 servers have **enabled** session local storage, so they **can** store HTTP sessions using Coherence*Web.

8. Coherence manages replicating sessions across server3 and server4 using a primary and backup mechanism.
9. You run the ShoppingCart application to test your newly configured replication settings.
10. You restart cluster1 servers and test the application again to demonstrate that the ShoppingCart application session is still intact because it is stored in a Coherence*Web cache.

Tasks

1. Connect to the **host01** and **host02** machines.
2. Set up the practice environment.
   a. You can reuse the MAIN terminal window from previous practices. Otherwise, open a terminal window **on each machine** by clicking the terminal icon in the launch panel located at the top of the screen.
   b. Set the title of the terminal window by selecting Terminal > Set Title and entering MAIN as the title. This makes it easier to distinguish the purpose of each window.
   c. In the terminal window **on host01**, navigate to the `practice16-01` folder.

```
$ cd /practices/part2/practice16-01
```

   d. Execute the `setup.sh` script to set up the initial practice environment:

```
$ ./setup.sh
```

   This script performs the following:

   – Ensures that no previous servers are running on both machines
   – Restores the domain and practice to their original state
   – Creates the required Coherence Cluster
   – Configures two new servers, server3 on host01 and server4 on host2
   – Configures a new cluster, cluster2, and adds server3 and server4 to it
   – Starts the `wlsadmin` domain on host01 and host02, not including cluster2
   – Ensures that no applications or libraries are deployed

   e. Start OHS by executing the following command in the MAIN terminal window **on host01**. OHS is already configured for this course.

   Note: Skip this step if OHS is already running on host01.

```
$ startohs.sh
```

   f. Ensure that OHS has started properly.
3. Configure Coherence*Web session persistence for the application.

   First you configure the application to support in-memory session replication. This allows your session to persist after you shut down the server handling your requests. Replication for this part of the practice is done using the servers' default channel.

   a. In a terminal window **on host01**, navigate to the application's `WEB-INF` folder and modify its `weblogic.xml` descriptor.

```
$ cd resources/ShoppingCart/WEB-INF
$ gedit weblogic.xml
```

b. Find the following lines in the file. In the current configuration, WebLogic uses in-memory replication to replicate sessions for this application if it is deployed to a cluster.

```
<wls:session-descriptor>
  <wls:persistent-store-type>
    replicated_if_clustered
  </wls:persistent-store-type>
</wls:session-descriptor>
```

c. Change the `replicated_if_clustered` entry to **`coherence-web`**. This causes the deployed application to use Coherence*Web session persistence.

d. Save the file.

4. Examine Coherence installation.

   a. This step is optional, however if you took the time to copy the `coherence-web.jar` file to a temporary folder and extract its contents, you would find the `default-session-cache-config.xml` file. This file contains the Coherence cache configuration for storing HTTP sessions. It configures the WebLogic server as a storage-disabled member of the cluster, and configures a two-tier Coherence cache to store HTTP sessions. The cache is a front-end local near cache backed by a back-end distributed (or partitioned) cache that stores and replicates the actual HTTP session objects. When a session is updated in the distributed cache, the copy in the local near cache is invalidated so subsequent requests for the session are sent to the distributed cache, and the local near cache is updated with a copy of the new data.

   b. Again, you do not have to do this part, but if you took the time to copy the `coherence.jar` file to a new folder and extracted its contents, you would find the default `tangosol-coherence.xml` configuration file. This file contains the default Coherence configuration for setting the cluster name, server name, multicast address and port, authorized hosts, and many other settings that determine which Coherence clusters with which this particular instance can connect. By default, Coherence is configured to automatically detect and connect to any Coherence clusters it finds. So by default, Coherence*Web members automatically detect and connect to any Coherence cluster they find. An instance can connect only to a single cluster. Multiple Coherence instances running the same configuration will automatically detect each other and form a single cluster that comprises each individual instance. You will use this default behavior to create a Coherence cluster for this practice. In your corporate environment, you would change the cluster name and other settings to ensure proper behavior for your environment's needs.

5. Add cluster2 servers to CoherenceCluster1.

   A WebLogic domain creates a default Coherence cluster. This cluster is configured to use unicast communication for connecting to other cluster members. This cluster is also configured to form a cluster with a specific name that all the servers in the cluster use to form a Coherence cluster. Servers can be added and removed from a Coherence cluster, regardless of WebLogic cluster configuration.

   a. Open the WebLogic administration console.
   b. In the **Domain Structure** pane, expand **Environment**, and click **Coherence Clusters**.
   c. Click **CoherenceCluster1** to view its settings.
   d. Click the **Members** tab.
   e. Click **Lock & Edit**.
   f. Check **cluster2** (ensuring that All servers in the cluster is also selected).
   g. Click **Save**.

6. Configure cluster2 servers to enable session local storage (the Coherence*Web cache).

   Note: When managed servers are created, session local storage is disabled by default. Because disabled is the default setting, servers in cluster1 are already configured to not store Coherence*Web sessions.

   a. In the **Domain Structure** Pane, select **Environment > Clusters > cluster2**.
   b. Click the **Coherence** tab.
   c. Deselect **Local Storage Enabled**. This disables servers of this cluster from storing data in regular Coherence caches that are not related to Coherence*Web.
   d. Select **Coherence Web Local Storage Enabled**.
   e. Note that cluster2 is part of CoherenceCluster1.
   f. Click **Save**.
   g. You can review the Coherence tab for cluster1 if you want to verify that Coherence Web Local Storage Enabled is not selected.
   h. Activate your changes.

7. Start the Coherence*Web cache servers.

   a. Execute the following commands in a terminal window on host01 to start server3 and server4:

   ```
   $ startServer3.sh
   $ startServer4.sh
   ```

   b. Wait for both servers to fully start before continuing.

8. Deploy the application.

   a. Now that you have changed the configuration of the application, you can deploy it to realize your configuration. Use the WebLogic administration console to deploy and activate the ShoppingCart application that is in the practice `resources` folder. Remember to deploy the application to all the servers in cluster**1**.

      Do not deploy the application to cluster2.

   b. Verify in the administration console that the application's state is Active. Remember you must start the application and refresh the page to see the proper state.

9. Test the ShoppingCart application by using the current web browser to ensure that the application is working as expected:
   a. Browse to http://host01:7777/ShoppingCart. You should see the application appear.
   b. Click *Go Shopping* and add something to your shopping cart.
   c. Take note of which server console window displays output related to the application (server1 or server2). That is the server that is currently handling the requests for the application.
   d. Shut down that server by pressing **Ctrl + C** in the terminal window and close the terminal window.

      Why? You shut down this server so that the subsequent request to the application fails over to the other server.
   e. Return to the ShoppingCart application and try to view the shopping cart. You should see that your purchase remains in your cart. This is because Coherence*Web session persistence is successfully persisting your session. The web server has successfully detected that the server is no longer available and failed over to the other server. You should also see that shopping cart items are now logged to the other server, which shows that the failover was successful.
   f. Shut down the other server by pressing **Ctrl + C** in the terminal window and close the terminal window.

      Why? How do you know that your session is actually persisted to Coherence? It could still be using in-memory replication. However, if you shut down both servers, restart them and view your cart again. Your data will still be there if you successfully configured Coherence*Web to persist your sessions. Even when your servers are shut down, the session data is still stored on the Coherence*Web cache in server3 and server4.
   g. Restart server1 and server2.

      ```
      $ startServer1.sh
      $ startServer2.sh
      ```
   h. Wait for both servers to fully start.
   i. Return to the ShoppingCart application and try to view the shopping cart. You should see that your purchase remains in your cart, even though your servers were completely shut down. This shows that your sessions were persisted by Coherence*Web in your cache servers.

      Congratulations! You have successfully configured and used Coherence*Web to persist your HTTP sessions.
10. Shut down the environment.
    a. Press **Ctrl + C** in all terminal windows that are running servers.
    b. Close all terminal windows used for each server.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Open a new terminal window **on each machine**:
2. Change the current directory to the current practice folder **on each machine**.
3. Execute the solution script **on host01**:

   ```
   $ ./solution.sh
   ```

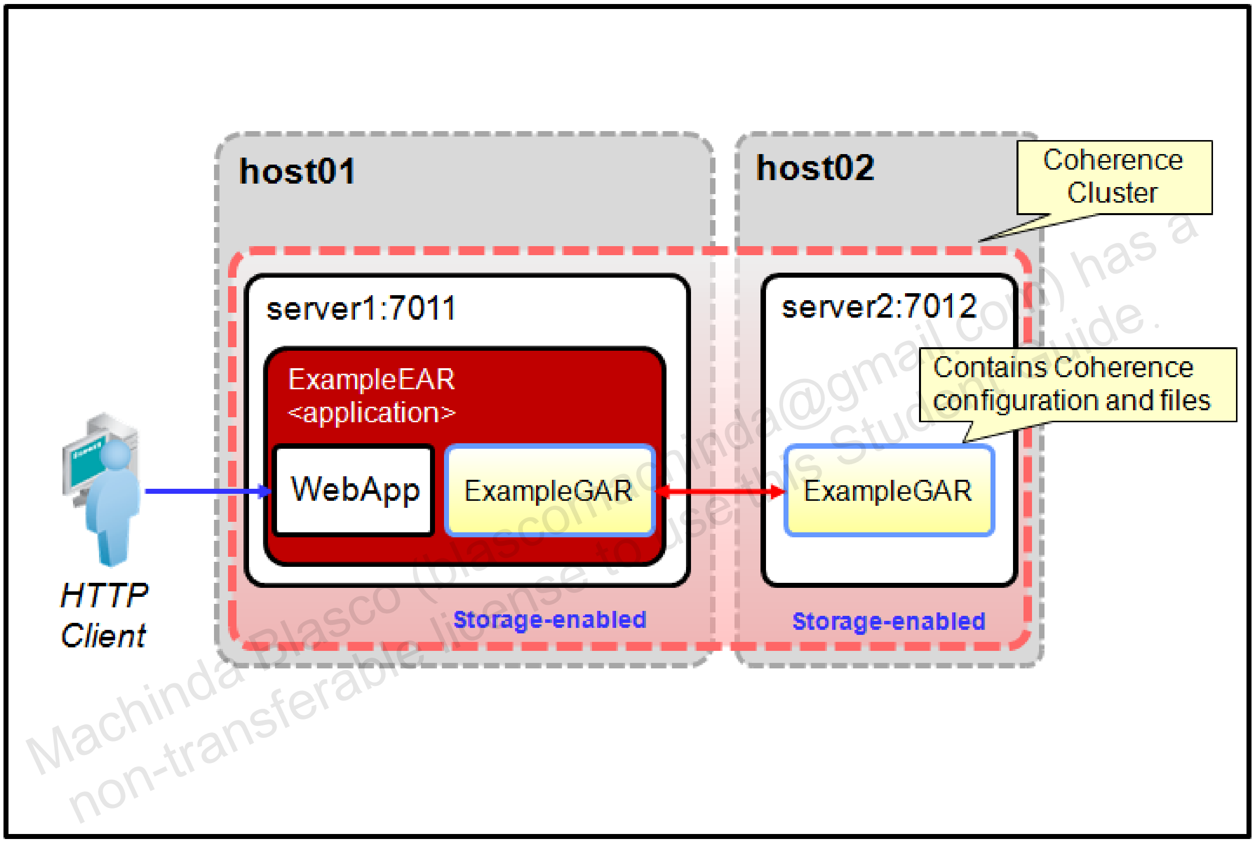
   The solution script performs the following:

   - Ensures that no previous servers are running on both machines
   - Restores the domain and practice to their original state
   - Creates the required Coherence cluster
   - Configures WebLogic cluster2 with managed servers server3 and server4 as members
   - Configures cluster2 as part of CoherenceCluster1
   - Disables Coherence Web Local Storage on cluster1
   - Enables Coherence Web Local Storage on cluster2
   - Starts the `wlsadmin` domain on both machines, including cluster2
   - Deploys the solution practice application
4. Wait for all servers to fully start.
5. Perform steps 2e and 2f to start the OHS server if it is not running already.
6. Continue starting at step 9.

Practice 16-2: Configuring Managed Coherence Servers

Overview

Coherence applications are typically deployed stand alone on a data-focused WebLogic Cluster, and as part of an Enterprise application on a web-tier cluster. This practice does not use the WebLogic cluster feature in any way. During this practice, you configure all the required WebLogic resources, including managed Coherence, and then deploy stand-alone and component Coherence applications to members of the cluster. When complete, the application architecture will resemble the following:



This image depicts the architecture of the environment for this practice:

1. The ExampleEAR application is deployed to `server1`, running on host01 on port 7011. This application includes ExampleGAR, a Coherence Grid Archive, as one of its modules.
2. The ExampleGAR application is deployed to `server2`, running on host02 on port 7012. The ExampleGAR application is a stand-alone Coherence Grid Archive, and it is identical to the module embedded within the ExampleEAR application.
3. The Coherence configuration on `server1` is storage enabled, which means that it is part of the Coherence cluster and it stores cache data.
4. The Coherence configuration on `server2` is storage enabled, which means that it is part of the Coherence cluster and it stores cache data.
5. Both servers store cache data because they belong to the same cluster. The cluster setting overrides any settings at the server level. If server2 was in a separate cluster, it could have a separately configured local storage setting.
6. You run the application to test the configuration.

Tasks

1. Connect to the **host01** and **host02** machines.
2. Set up the practice environment.
   a. You can reuse the MAIN terminal window from previous practices. Otherwise, open a terminal window **on each machine** by clicking the terminal icon in the launch panel located at the top of the screen.
   b. Set the title of the terminal window by selecting Terminal > Set Title and entering MAIN as the title. This makes it easier to distinguish the purpose of each window.
   c. In the terminal window **on host01**, navigate to the `practice16-02` folder:

```
$ cd /practices/part2/practice16-02
```

   d. Execute the `setup.sh` script to set up the initial practice environment:

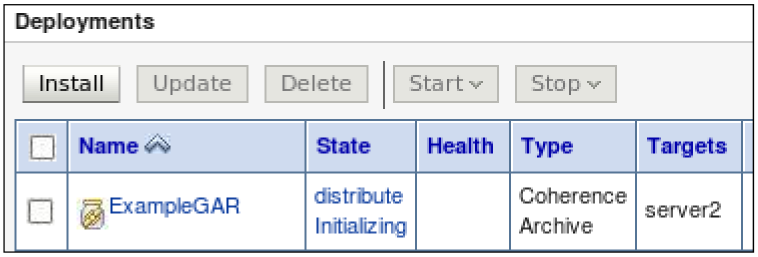
```
$ ./setup.sh
```

   This script performs the following:
   - Ensures that no previous servers are running on both the machines
   - Restores the domain and practice to their original state
   - Starts the `AdminServer` on host01
   - Ensures that no applications or libraries are deployed
3. Create Coherence Cluster and Configure Managed Coherence Servers.
   a. In the **Domain Structure** pane, expand **Environment** and select **Coherence Clusters**.
   b. Click **Lock & Edit**.
   c. Click **New** to create a new Coherence cluster.
   d. Name the cluster *ManagedCoherenceCluster* and click **Next**.
   e. In the *Coherence Cluster Addressing* step, leave all values unchanged and click **Next**.
   f. In the *Coherence Cluster Members* step, select `cluster1` and ensure that *All servers in the cluster* is selected.
   g. Click **Finish**.
4. Deploy ExampleGAR application.
   a. In the **Domain Structure** pane, click **Deployments**.
   b. Click **Install** to install the stand-alone GAR file.
   c. In the *Locate Deployment to install and prepare* step, enter the following:

```
/practices/part2/practice16-02/resources
```

   d. Select `ExampleGAR.gar` and click **Next**.
   e. Click **Next** to install as an application.
   f. In the *Select Deployment Targets* step, select `server2`, and then click **Next**.
   The stand-alone ExampleGAR application represents a Coherence caching application that is used to store data.
   g. In the *Optional Settings* step, leave all values unchanged and click **Finish**.

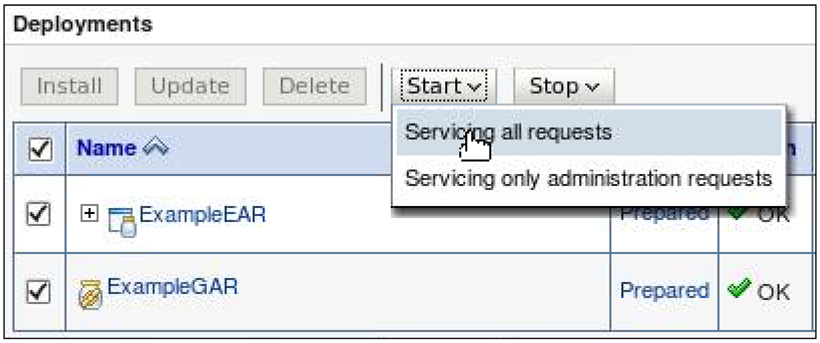
h. Use the administration console to verify that the application is deployed.



☐	Name	State	Health	Type	Targets
☐	ExampleGAR	distribute Initializing		Coherence Archive	server2

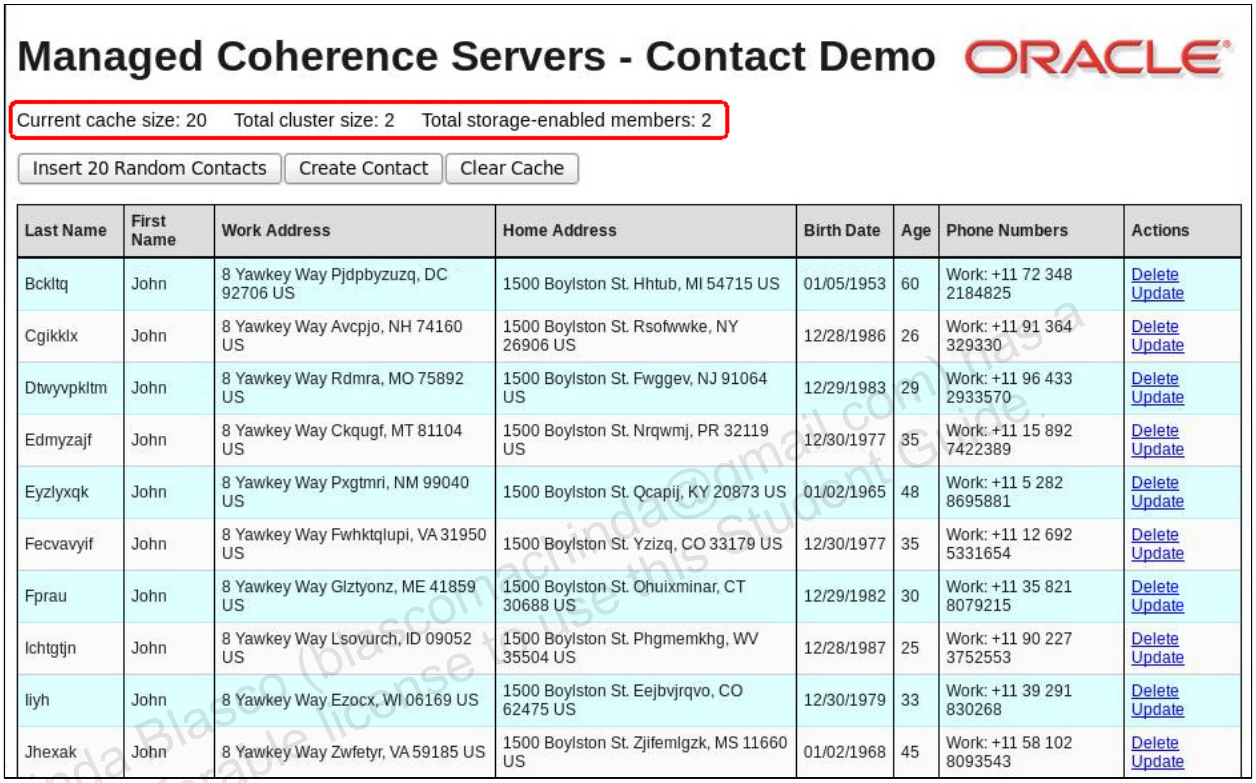
5. Deploy the ExampleEAR application.
   a. Click Install to install the EAR application file, which contains the embedded GAR file.
   b. In the *Locate Deployment to install and prepare* step, enter the following:
      `/practices/part2/practice16-02/resources`
   c. Select `ExampleEAR.ear` and click **Next**.
   d. Click **Next** to install as an application.
   e. In the *Select Deployment Targets* step, select `server1` and click **Next**. The EAR application represents a Coherence caching application that is a member of the Coherence cluster that is used to store data.
   f. In the *Optional Settings* step, leave all values unchanged and click **Finish**.
   g. Use the administration console to verify that the application is deployed.
   h. Activate your changes.
6. Start servers.
   a. Execute the following script to run Node Manager **on each host**:
      `$` **`wlst.sh startNM.py`**
   b. In the **Domain Structure** pane, navigate to **Environment > Servers**.
   c. Click the **Control** tab.
   d. Select both `server1` and `server2` servers and click Start to start the servers.
   e. Wait until both servers are in the RUNNING state.

      Hint: Use the ↻ icon to cause the page to automatically refresh.
   f. If the applications have not started (are not in the Active state), then go to the deployment list and start them both.



☑	Name	State	Health
☑	⊞ ExampleEAR	Prepared	✓ OK
☑	ExampleGAR	Prepared	✓ OK

7. Test the managed Coherence server application.
   a. Open a new Firefox tab.
   b. Browse to the following URL to use the application:
      `http://host01:7011/example-web-app/faces/ContactList.jsp`
   c. Click the *Insert 20 Random Contacts* button. The application will generate content and should resemble:



Managed Coherence Servers - Contact Demo

Current cache size: 20 Total cluster size: 2 Total storage-enabled members: 2

Last Name	First Name	Work Address	Home Address	Birth Date	Age	Phone Numbers	Actions
Bckltq	John	8 Yawkey Way Pjdpbyzuzq, DC 92706 US	1500 Boylston St. Hhtub, MI 54715 US	01/05/1953	60	Work: +11 72 348 2184825	Delete Update
Cgikklx	John	8 Yawkey Way Avcpjo, NH 74160 US	1500 Boylston St. Rsofwwke, NY 26906 US	12/28/1986	26	Work: +11 91 364 329330	Delete Update
Dtwyvpkltm	John	8 Yawkey Way Rdmra, MO 75892 US	1500 Boylston St. Fwggev, NJ 91064 US	12/29/1983	29	Work: +11 96 433 2933570	Delete Update
Edmyzajf	John	8 Yawkey Way Ckqugf, MT 81104 US	1500 Boylston St. Nrqwmj, PR 32119 US	12/30/1977	35	Work: +11 15 892 7422389	Delete Update
Eyzlyxqk	John	8 Yawkey Way Pxgtmri, NM 99040 US	1500 Boylston St. Qcapij, KY 20873 US	01/02/1965	48	Work: +11 5 282 8695881	Delete Update
Fecvavyif	John	8 Yawkey Way Fwhktqlupi, VA 31950 US	1500 Boylston St. Yzizq, CO 33179 US	12/30/1977	35	Work: +11 12 692 5331654	Delete Update
Fprau	John	8 Yawkey Way Glztyonz, ME 41859 US	1500 Boylston St. Ohuixminar, CT 30688 US	12/29/1982	30	Work: +11 35 821 8079215	Delete Update
Ichtgtjn	John	8 Yawkey Way Lsovurch, ID 09052 US	1500 Boylston St. Phgmemkhg, WV 35504 US	12/28/1987	25	Work: +11 90 227 3752553	Delete Update
Iiyh	John	8 Yawkey Way Ezocx, WI 06169 US	1500 Boylston St. Eejbvjrqvo, CO 62475 US	12/30/1979	33	Work: +11 39 291 830268	Delete Update
Jhexak	John	8 Yawkey Way Zwfetyr, VA 59185 US	1500 Boylston St. Zjifemlgzk, MS 11660 US	01/02/1968	45	Work: +11 58 102 8093543	Delete Update

   d. Feel free to explore the application.

   Congratulations! You have successfully configured managed Coherence servers.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Open a new terminal window **on each machine**:
2. Change the current directory to the current practice folder **on each machine**.
3. Execute the solution script **on host01**:

   ```
   $ ./solution.sh
   ```

   Note: You can ignore any errors regarding deployments getting deferred because the servers are not available during deployment.

   The solution script performs the following:

   - Terminates any running `java` programs on both machines
   - Starts the `AdminServer` server in its own terminal window
   - Configures a new WebLogic Coherence cluster for the practice
   - Configures the servers to use the cluster as needed for this practice
   - Undeploys all applications and libraries
   - Deploys the practice applications
4. Continue starting at step number 6.

Practices for Lesson 17: Application Staging and Deployment Plans

Chapter 17

Practices for Lesson 17: Overview

Practices Overview

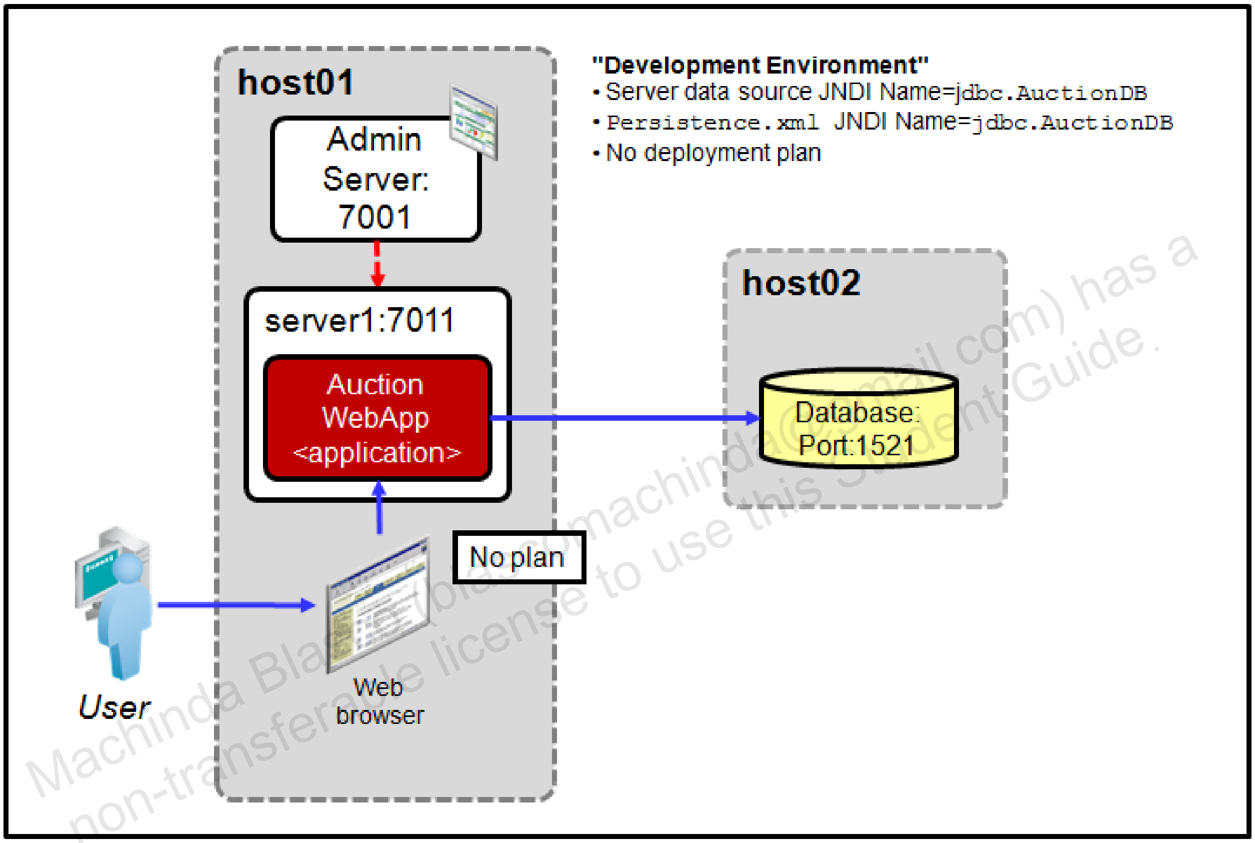
In the practices for this lesson, you create and use deployment plans to override the configuration properties contained in an application. This shows you how to successfully deploy WebLogic applications from one environment to another without the need to unpack a deployment archive to move an application.

Practice 17-1: (Optional) Creating and Using a Deployment Plan

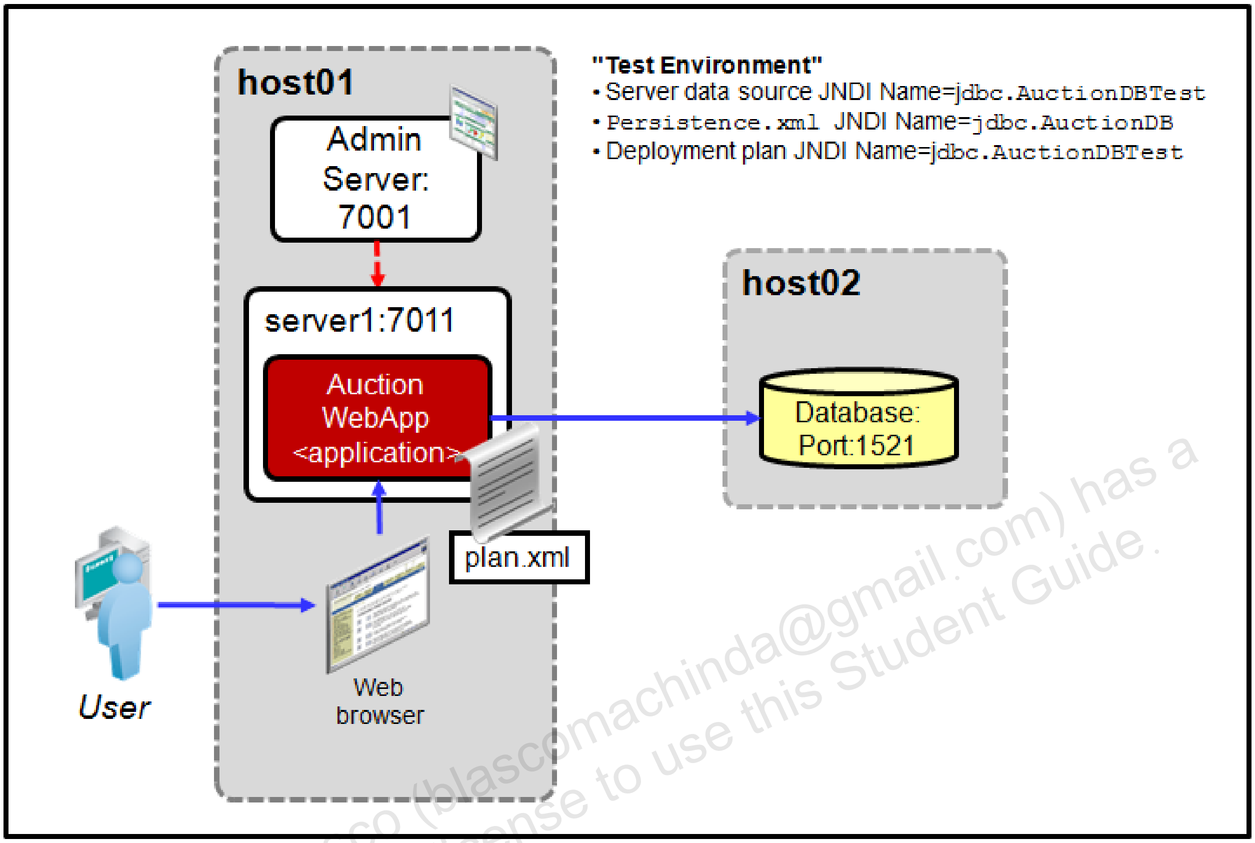
Overview

This practice shows you how to create a deployment plan by using the `weblogic.PlanGenerator` tool, modify the plan to change the JNDI name of the data source to use, and apply the plan to a deployed application.

The following image depicts the representation of the "Development" environment that does not incorporate a deployment plan:


host01
Admin Server: 7001
server1:7011
Auction WebApp <application>
No plan
User
Web browser
host02
Database: Port:1521
"Development Environment"
• Server data source JNDI Name=jdbc.AuctionDB
• Persistence.xml JNDI Name=jdbc.AuctionDB
• No deployment plan

The following image depicts the representation of the "Test" environment that uses a deployment plan to override the deployment's configuration so it works successfully in the new environment where the JNDI name of a data source is different:


host01
Admin Server: 7001
server1:7011
Auction WebApp <application>
plan.xml
User
Web browser
host02
Database: Port:1521
"Test Environment"
• Server data source JNDI Name=jdbc.AuctionDBTest
• Persistence.xml JNDI Name=jdbc.AuctionDB
• Deployment plan JNDI Name=jdbc.AuctionDBTest

Tasks

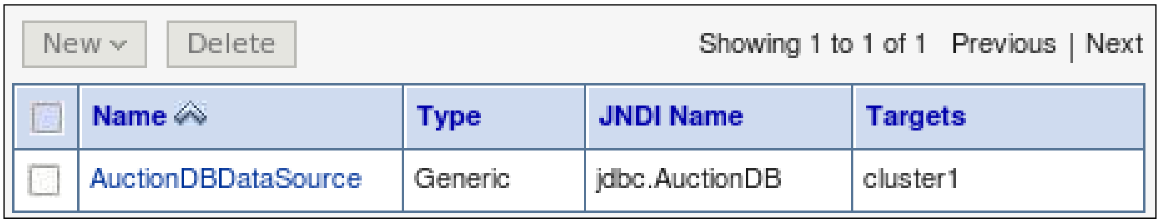
1. Connect to the **host01** machine.
2. Set up the practice environment.
   a. You may reuse the MAIN terminal window from previous practices. Otherwise, open a terminal window **on host01** by clicking the terminal icon in the launch panel located at the top of the screen.
   b. Set the title of the terminal window by selecting Terminal > Set Title and entering `MAIN` as the title. This makes it easier to distinguish the purpose of each window.
   c. In the terminal window **on host01**, navigate to the `practice17-01` folder and execute the `setup.sh` script to set up the initial practice environment:

```
$ cd /practices/part2/practice17-01
$ ./setup.sh
```

This script performs the following:

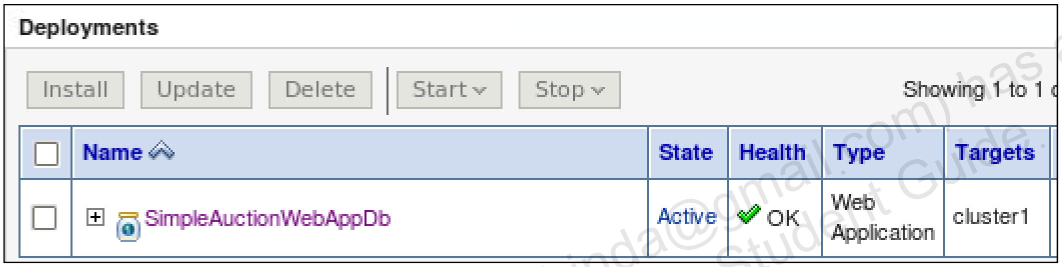
- Ensures that no previous servers are running on both machines
- Restores the domain and practice to their original state
- Starts the `wlsadmin` domain on host01 (host02 is not used in this practice.)
- Deploys the starting application used for this practice

3. Verify the domain's configuration.
   a. Launch a web browser and log in to the administration console:
      `http://host01:7001/console`
   b. In the **Domain Structure** pane, navigate to **Services > Data Sources**.
   c. Verify that the `jdbc.AuctionDB` data source exists and is targeted to `cluster1`. Verify that its JNDI setting is `jdbc.AuctionDB`.



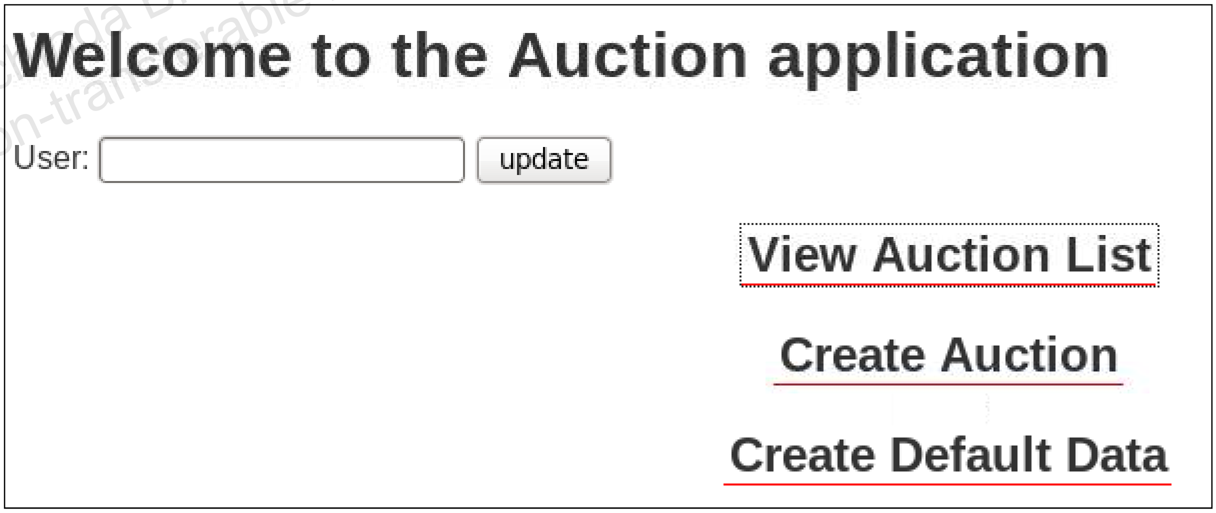
□	Name	Type	JNDI Name	Targets
□	AuctionDBDataSource	Generic	jdbc.AuctionDB	cluster1

   d. Verify that the SimpleAuctionWebAppDb application is deployed.



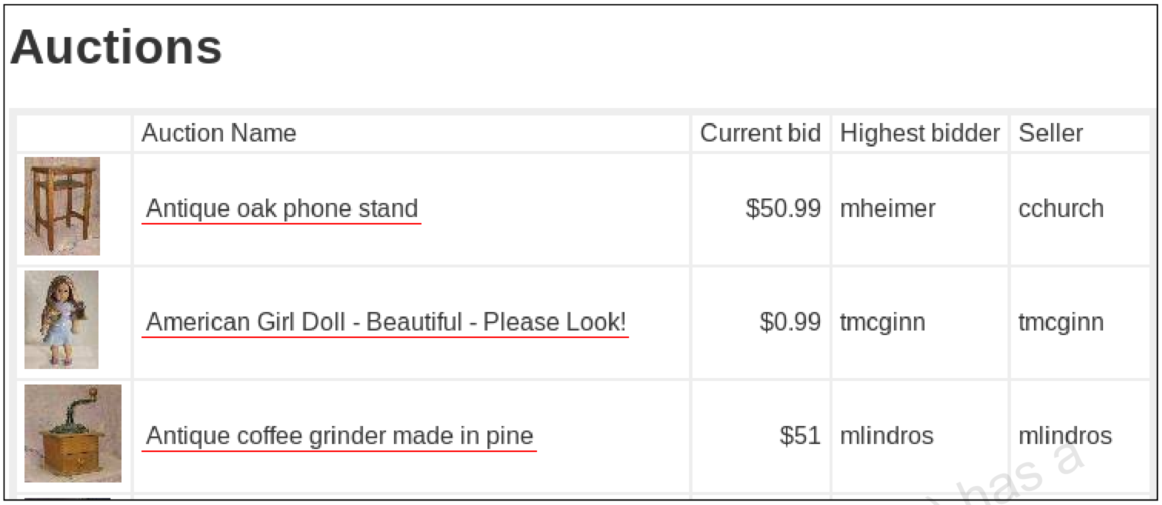
□	Name	State	Health	Type	Targets
□	⊞ SimpleAuctionWebAppDb	Active	✓ OK	Web Application	cluster1

4. Test the Auction application.
   a. Direct your web browser to the following URL:
      `http://host01:7011/SimpleAuctionWebAppDb`
   b. Click the *Create Default Data* link to populate the database with data for the Auction application. Follow this by clicking the confirmation to create the data.


Welcome to the Auction application

User:

c. Click the *Go Home* link and then click the *View Auction List* link to view the list of auctions stored in the database.



Auctions

	Auction Name	Current bid	Highest bidder	Seller
	Antique oak phone stand	$50.99	mheimer	cchurch
	American Girl Doll - Beautiful - Please Look!	$0.99	tmcginn	tmcginn
	Antique coffee grinder made in pine	$51	mlindros	mlindros

This shows that the Auction application is currently working with the AuctionDB data source with a JNDI name of `jdbc.AuctionDB`. Now you will convert the domain and emulate deploying the application to a new environment by using a deployment plan to override the application's JNDI name for the data source.

5. Generate deployment plan by using `weblogic.PlanGenerator`:

a. Perform the following steps to create a deployment plan from the beginning that is based on the currently deployed SimpleAuctionWebAppDb application:

```
$ java weblogic.PlanGenerator -root
/practices/part2/practice17-01 /practices/part2/practice17-
01/resources/SimpleAuctionWebAppDb.war
```

b. Review the arguments of the command in the last step:

- The tool's help or documentation says that the `-root` option specifies the location of the application. More specifically, it specifies the location to create the deployment plan. If an application happens to reside at this location, it will create the deployment plan based on the settings of that application.
- In this practice, the application is an archived WAR file, so you specify the location of the application archive as another argument.

c. After running the script, you should see that the tool generated the deployment plan in the `/practices/part2/practice17-01/plan` folder in a file called `plan.xml`.

6. Review and experiment with the deployment plan:

a. Perform the following commands to open the deployment plan:

```
$ cd plan
$ gedit plan.xml
```

b. Review the contents of the generated plan and close the file when finished:

   – The generator places all configurable options of the application into the plan without setting any properties.

   – The properties for the servlets in the application are defined in the `variable-definition` and `variable-assignment` elements. The configuration overrides only the name of each servlet. This provides a template for administrators to start with.

   – The properties that define the deployment descriptors found in the application are defined in the `module-descriptor` elements.

c. The generator can create more properties depending on the application itself and the options passed to the tool. Perform the following commands to rename the `plan.xml` file to save it for modification and to create a new plan with the generator again:

```
$ mv plan.xml plan.xml.1
$ java weblogic.PlanGenerator -all -root
/practices/part2/practice17-01 /practices/part2/practice17-
01/resources/SimpleAuctionWebAppDb.war
```

   The `-all` option tells the tool to create property entries for all discovered properties found in the application.

d. Open the new `plan.xml` file.

e. Look through the file and take note of how many more properties have been generated for the plan. Although this is useful for some types of applications where a lot of control is desired, it is not a good approach when you want to override only a handful of properties. Close the file when finished.

f. Delete the new plan file and rename the original plan file `plan.xml`:

```
$ mv plan.xml.1 plan.xml
```

7. Modify the AuctionDB data source in the plan:

   a. For this practice, you are interested in changing only the JNDI name that is associated with the application. The first thing you may notice is that there are no variable-definition elements that relate to `persistence.xml`, which is the descriptor that contains the configuration you want to modify. However, the `persistence.xml` module is defined in the `module-override` element with no `variable-assignments` defined.

b. So the first thing you want to do is remove all of the configuration that does not pertain to the `persistence.xml` file.

- Open the `plan.xml` file again for editing.
- Find the following lines for `variable` elements in the file and delete them:

```
<variable>
 <name>ServletDescriptor_UpdateUserServlet_...</name>
 <value>UpdateUserServlet</value>
</variable>
<variable>
 <name>ServletDescriptor_AuctionImageServlet_...</name>
 <value>AuctionImageServlet</value>
</variable>
<variable>
 <name>ServletDescriptor_BidServlet_...</name>
 <value>BidServlet</value>
</variable>
<variable>
 <name>ServletDescriptor_CreateAuctionServlet_...</name>
 <value>CreateAuctionServlet</value>
</variable>
<variable>
 <name>ServletDescriptor_SetupServlet_...</name>
 <value>SetupServlet</value>
</variable>
<variable>
 <name>ServletDescriptor_LogoutServlet_...</name>
 <value>LogoutServlet</value>
</variable>
<variable>
 <name>ServletDescriptor_ListServlet_...</name>
 <value>ListServlet</value>
</variable>
<variable>
 <name>ServletDescriptor_DetailServlet_...</name>
 <value>DetailServlet</value>
</variable>
```

– Find the following lines for `module-descriptor` elements in the file and delete them:

```
<module-descriptor external="false">
 <root-element>weblogic-web-app</root-element>
 <uri>WEB-INF/weblogic.xml</uri>
 <variable-assignment>
  <name>ServletDescriptor_UpdateUserServlet_...</name>
  <xpath>/weblogic-web-app/servlet-descriptor
         /[servlet-name="UpdateUserServlet"]
         /servlet-name</xpath>
  <origin>planbased</origin>
 </variable-assignment>
 <variable-assignment>
  <name>ServletDescriptor_AuctionImageServlet_...</name>
  <xpath>/weblogic-web-app/servlet-descriptor
         /[servlet-name="AuctionImageServlet"]
         /servlet-name</xpath>
  <origin>planbased</origin>
 </variable-assignment>
 <variable-assignment>
  <name>ServletDescriptor_BidServlet_...</name>
  <xpath>/weblogic-web-app/servlet-descriptor
         /[servlet-name="BidServlet"]/servlet-name</xpath>
  <origin>planbased</origin>
 </variable-assignment>
 <variable-assignment>
  <name>ServletDescriptor_CreateAuctionServlet_...</name>
  <xpath>/weblogic-web-app/servlet-descriptor
         /[servlet-name="CreateAuctionServlet"]
         /servlet-name</xpath>
  <origin>planbased</origin>
 </variable-assignment>
 <variable-assignment>
  <name>ServletDescriptor_SetupServlet_...</name>
  <xpath>/weblogic-web-app/servlet-descriptor
         /[servlet-name="SetupServlet"]/servlet-name</xpath>
  <origin>planbased</origin>
 </variable-assignment>
 <variable-assignment>
  <name>ServletDescriptor_LogoutServlet_...</name>
  <xpath>/weblogic-web-app/servlet-descriptor
         /[servlet-name="LogoutServlet"]
         /servlet-name</xpath>
  <origin>planbased</origin>
 </variable-assignment>
 <variable-assignment>
  <name>ServletDescriptor_ListServlet_...</name>
  <xpath>/weblogic-web-app/servlet-descriptor
         /[servlet-name="ListServlet"]/servlet-name</xpath>
  <origin>planbased</origin>
 </variable-assignment>
 <variable-assignment>
  <name>ServletDescriptor_DetailServlet_...</name>
  <xpath>/weblogic-web-app/servlet-descriptor
         /[servlet-name="DetailServlet"]
         /servlet-name</xpath>
  <origin>planbased</origin>
 </variable-assignment>
</module-descriptor>
<module-descriptor external="false">
 <root-element>web-app</root-element>
 <uri>WEB-INF/web.xml</uri>
</module-descriptor>
<module-descriptor external="true">
 <root-element>wldf-resource</root-element>
 <uri>META-INF/weblogic-diagnostics.xml</uri>
</module-descriptor>
<module-descriptor external="true">
 <root-element>persistence-configuration</root-element>
 <uri>WEB-INF/lib/persistence.jar!/META-INF/persistence-
 configuration.xml</uri>
</module-descriptor>
```

c. Now your plan file should resemble the following:

```
<?xml version='1.0' encoding='UTF-8'?>
<deployment-plan
xmlns="http://xmlns.oracle.com/weblogic/deployment-plan"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.oracle.com/weblogic/deployment-
plan http://xmlns.oracle.com/weblogic/deployment-
plan/1.0/deployment-plan.xsd" global-variables="false">
<application-name>SimpleAuctionWebAppDb.war</application-name>
 <variable-definition>

 </variable-definition>
 <module-override>
  <module-name>SimpleAuctionWebAppDb.war</module-name>
  <module-type>war</module-type>

  <module-descriptor external="false">
   <root-element>persistence</root-element>
   <uri>WEB-INF/classes/META-INF/persistence.xml</uri>
  </module-descriptor>
 </module-override>
<config-root>/practices/part2/practice17-01/plan</config-root>
</deployment-plan>
```

All that remains is the basic skeleton that you will use to create your plan:

- The standard XML declaration elements
- The `application-name` element
- The `variable-definition` element that contains your variable definition
- The `module-descriptor` element for the `persistence.xml` file
- The `config-root` element that specifies the location of the plan file

d. Add the following `variable` XML code to the `variable-definition` element so it resembles the following:

```
<variable-definition>
  <variable>
    <name>PersistenceUnit_AuctionPU_jtaDataSource</name>
    <value>jdbc.AuctionDBTest</value>
  </variable>
</variable-definition>
```

Note that the new value for the variable is `jdbc.AuctionDBTest`. This will be the new JNDI name for the application when it is deployed.

e. Add the following `variable-assignment` XML code to the `module-descriptor` element, so it resembles the following:

```
<module-descriptor external="false">
  <root-element>persistence</root-element>
  <uri>WEB-INF/classes/META-INF/persistence.xml</uri>
  <variable-assignment>
    <name>PersistenceUnit_AuctionPU_jtaDataSource</name>
    <xpath>/persistence/persistence-unit[name="AuctionPU"]/jta-
data-source</xpath>
  </variable-assignment>
</module-descriptor>
```

This code indicates that the variable value `jdbc.AuctionDBTest` is overriding the value specified in the `persistence.xml` file's `jta-data-source` element for the `AuctionPU` `persistence-unit` element.

f. Save your changes.

8. Redeploy the application along with the deployment plan:

a. Using the administration console, undeploy the currently running SimpleAuctionWebAppDb application as follows:

– In the **Domain Structure** pane, click **Deployments**.

– Click **Lock & Edit**.

– Select the **Control** tab.
Select the check box next to `SimpleAuctionWebAppDb`.

– Click **Stop > Force** stop now and verify Yes.

– Select the box next to `SimpleAuctionWebAppDb`.

– Click **Delete** and verify **Yes**.

– Click **Activate Changes** to realize the undeployment.

b. Modify the AuctionDBDataSource data source in the domain so its JNDI name is now `jdbc.AuctionDBTest`. This effectively changes the environment so that it is different from the environment the application was undeployed from.

– In the **Domain Structure** pane, navigate to **Services > Data Sources**.

– Verify that the `AuctionDBDataSource` data source exists and is targeted to `cluster1`. Verify that its JNDI setting is `jdbc.AuctionDB`.

– Click the data source to open its configuration page.

– Click **Lock & Edit**.

– Make sure the current tab selection is **Configuration > General**.

– Change the JNDI Name field to **`jdbc.AuctionDBTest`**.

– Click **Save**.

– Click **Activate Changes**.

– Verify that your data source is now set for your new JNDI name.

Note: This change requires restarting server1 before it is realized.

c. Execute the following commands to restart server1:

– Press Ctrl + C in the server1 terminal window and close the window.

– Execute the start server script.

```
$ startServer1.sh
```

d. Using the administration console, deploy the application using your new deployment plan:

Note: Remember that the domain now represents the Test environment you are deploying the application to. You deleted the application and changed the JNDI name of the data source to effectively change the domain from Dev to Test. Now you perform a deployment (not a redeployment), which is what you would do in the Test environment if it were a different domain.

– In the **Domain Structure** pane, click **Deployments**.

– Click **Lock & Edit**.

– Click **Install**.

– If the deployment wizard is not already set to the `host01.example.com/practices/part2/practice17-01/resources` folder, then navigate to this location.

– Select the `SimpleAuctionWebAppDb.war` deployment archive and click **Next**.

– Select *Install this deployment as an application* and click Next.

– Select *All servers in the cluster* and click **Next**.

– Leave all the default settings and click **Next**.

– Leave all the default settings and click **Finish**.

– Click **Activate Changes**.

– This has deployed the application in the Prepared state, but has not leveraged the deployment plan yet.

– Click **Deployments**.

– Click **Lock & Edit**.

– Select the check box next to `SimpleAuctionWebAppDb`.

– Click **Update**.

– Click *Change Path* next to the *Deployment plan path* field.

– Navigate to `/practices/part2/practice17-01/plan`, select `plan.xml`, and click **Next**.

– Ensure *Redeploy this application using the following deployment files* is selected.

– See that the *Source path* is still the `/practices/part2/practice17-01/resources/SimpleAuctionWebAppDb.war` file and the *Deployment plan path* is set properly and click **Finish**.

– Click Activate Changes.

e. Verify that the deployment shows it is using the deployment plan and that the new value is set properly:

- Click **Deployments**.
- Click `SimpleAuctionWebAppDb`.
- On the Overview tab, verify that the *Deployment Plan* value is set to your plan file.
- Click the **Configuration** tab.
- Click **Persistence > AuctionPU > Data Sources**.
- See that the JNDI name is now `jdbc.AuctionDBTest`.

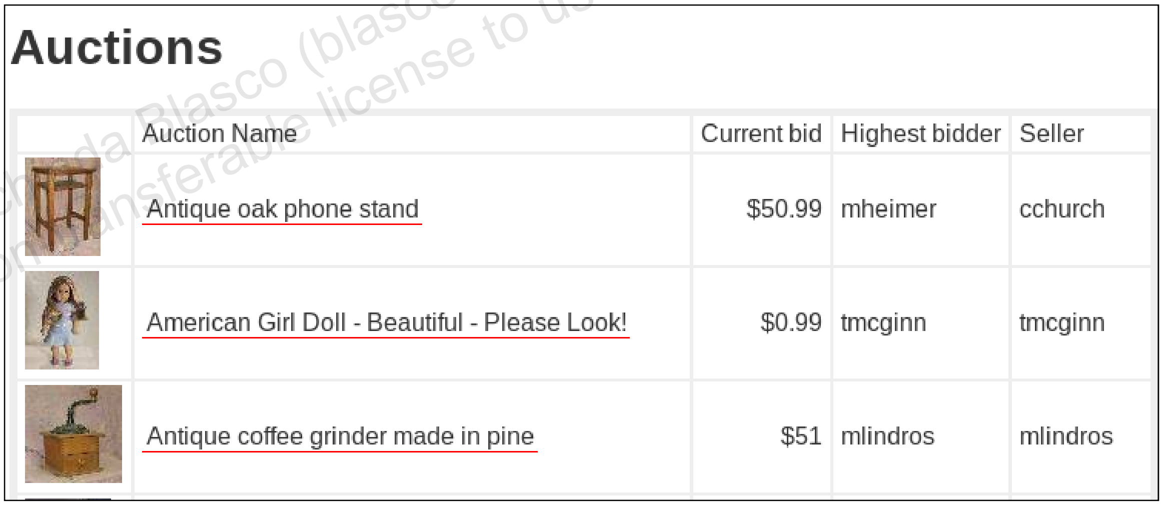
f. Start the application (if not already started):

- Click **Deployments**.
- Select the **Control** tab.
- Check the check box next to the application and **click Start > Servicing** all requests.
- Verify **Yes**.
- Ensure that the application's state is Active.

9. Test the Auction application again.

a. Direct your web browser to the following URL:
`http://host01:7011/SimpleAuctionWebAppDb`

b. Click the *View Auction List* link to view the list of auctions stored in the database. The database should already be created so you do not have to do that again.



Auctions

	Auction Name	Current bid	Highest bidder	Seller
	Antique oak phone stand	$50.99	mheimer	cchurch
	American Girl Doll - Beautiful - Please Look!	$0.99	tmcginn	tmcginn
	Antique coffee grinder made in pine	$51	mlindros	mlindros

This shows that the Auction application is currently working with the AuctionDB data source with a JNDI name of `jdbc.AuctionDBTest`. Congratulations! You have successfully used a deployment plan to deploy a single application to two different environments.

10. Experiment with deployment plan tools (optional): You used `weblogic.PlanGenerator` for this practice, but you could also have used the administration console completely to create a plan, change the configuration, and update the application with the new plan. You can undeploy the application and use the console to change the JNDI name of the data source for the application. This causes the administration console to create a new plan with your updated configuration and it sets the configuration changes into the plan automatically so you do not have to manually edit the deployment plan file. Give it a try if you have some time.

Practice Solution

There is no solution for this practice, although you can use the solution example files in the `solution` folder to assist with performing the practice. No other practices depend on this practice.

Practices for Lesson 18: Production Redeployment

Chapter 18

Practices for Lesson 18: Overview

Practices Overview

In the practices for this lesson, you configure multiple versions of an application that in turn reference different versions of a shared library. You also learn how to roll back to a previous application version.

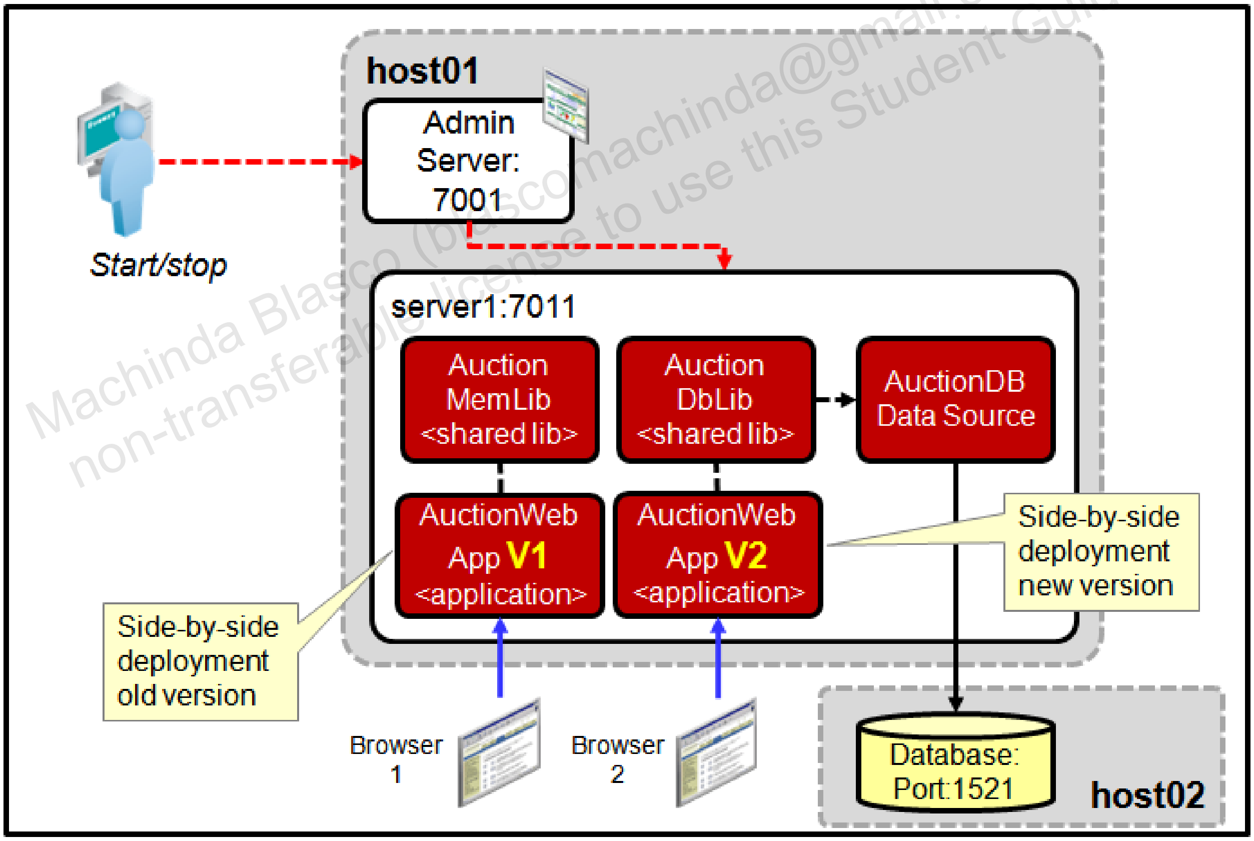
Practice 18-1: (Optional) Using Production Redeployment

Overview

This practice builds on `practice09-01` by showing you how to deploy multiple versions of the same application that also reference different versions of shared libraries. The servers are running and the shared libraries are already deployed as part of `practice09-01`. This practice guides you through the process of deploying a second version of the `AuctionWebApp` application that references the second "updated" version of the `AuctionLib` library that uses a database.

This practice is a good real-world example of how production redeployment, sometimes called side-by-side deployment, is used in earnest because applications that reference libraries will also need to reference different library versions. Although it is possible to deploy multiple versions of an application without regard for libraries, application and library versioning are connected. It is best to see how they work together to gain the maximum benefit of using these features.

This practice depends on completion of `practice09-01`. If you have not completed this practice, you may still continue because there is a step provided to quickly set up this practice as if `practice09-01` was completed.


host01
Admin Server: 7001
Start/stop
server1:7011
Auction MemLib <shared lib>
Auction DbLib <shared lib>
AuctionDB Data Source
AuctionWeb App V1 <application>
AuctionWeb App V2 <application>
Side-by-side deployment new version
Side-by-side deployment old version
Browser 1
Browser 2
Database: Port:1521
host02

This image depicts the architecture of the environment for this practice:

1. There are two shared libraries deployed:
   a. `AuctionMemLib`: A memory-based version of the library
   b. `AuctionDbLib`: A database version of the library
2. The first release of the `AuctionWebApp` application (V1) is deployed, which references the memory-based version of the `AuctionLib` library, `AuctionMemLib`.
3. The `AuctionDbLib` library is deployed.
4. You configure a new version of the `AuctionWebApp` application (V2) to reference the updated database version of the library.
5. You use production redeployment to deploy the new V2 version of `AuctionWebApp` at the same time that the existing V1 `AuctionWebApp` application is deployed so that both are running simultaneously. Existing clients continue to use the original V1 version while new clients are routed to use the new V2 version.
6. You test both versions of the application and verify that it is configured and working correctly.
7. You retire the V1 version of the application and see your clients seamlessly begin to use V2.

Tasks

1. Connect to the **host01** machine.
2. Set up the practice environment.
   a. You can reuse the MAIN terminal window from previous practices. Otherwise, open a terminal window **on host01** by clicking the terminal icon in the launch panel located at the top of the screen.
   b. Set the title of the terminal window by selecting Terminal > Set Title and entering `MAIN` as the title. This makes it easier to distinguish the purpose of each window.
   c. In the terminal window **on host01**, navigate to the `practice18-01` folder.

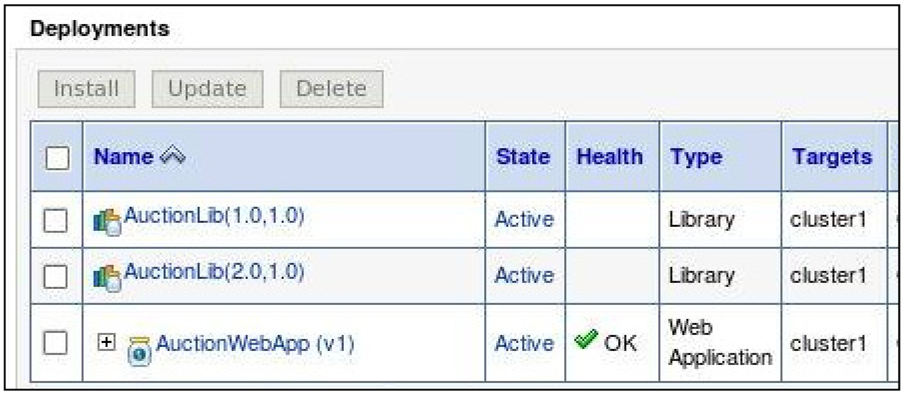
   ```
   $ cd /practices/part2/practice18-01
   ```

   d. Execute the `setup.sh` script to set up the initial practice environment:

   ```
   $ ./setup.sh
   ```

   This script performs the following:
   – Restores the domain and practice to their original state
   – Executes the solution script for `practice09-01`
3. Verify the domain's configuration.
   a. Launch a web browser and log in to the administration console:
   http://host01:7001/console

b. Verify that two versions of the `AuctionLib` library and the `AuctionWebApp (V1)` application are deployed.



Deployments

□	Name	State	Health	Type	Targets
□	AuctionLib(1.0,1.0)	Active		Library	cluster1
□	AuctionLib(2.0,1.0)	Active		Library	cluster1
□	⊞ AuctionWebApp (v1)	Active	✓ OK	Web Application	cluster1

4. Open a web browser to the `AuctionWebApp` application.

   a. Direct your web browser to the following URL:
   `http://host01:7011/AuctionWebApp/index.jsp`

   Note: You may need to exercise the Create Default Data functionality of the application to create the required in memory data.

   Note: This causes a client session to get created for the currently deployed `AuctionWebApp (V1)` application. We need the session to remain active during the production redeployment process because we want to show that both versions of the `AuctionWebApp` application are deployed at the same time and properly service existing and new cli ents.

5. Configure the new version of the `AuctionWebApp` application as the V2 version and to reference the database version of the `AuctionLib` library.

   a. Perform the following commands to copy your current `AuctionWebApp` V1 application to the current practice folder:

```
$ cp -r ../practice09-01/resources/AuctionWebApp .
```

   Note: If you did not successfully complete `practice09-01`, copy the solution files instead.

   b. Navigate to the `AuctionWebApp` folder and open its manifest file for editing:

```
$ cd AuctionWebApp/META-INF
$ gedit MANIFEST.MF
```

   c. Modify the `Weblogic-Application-Version` value to `v2`. This changes the application's version from v1 to v2 so that the original application version is different from this one. This causes WebLogic to deploy both versions of the application simultaneously.

```
Manifest-Version: 1.0
Class-Path:
Weblogic-Application-Version: v2
```

   d. Save the file.

   e. Navigate to the `AuctionWebApp` folder and open its deployment descriptor for editing:

```
$ cd ../WEB-INF
$ gedit weblogic.xml
```

f. Modify the `specification-version` value to `2.0`. This causes the `AuctionWebApp` application to reference the `AuctionDbLib` version of the `AuctionLib` library.

```
<wls:library-ref>
  <wls:library-name>AuctionLib</wls:library-name>
  <wls:specification-version>2.0</wls:specification-version>
  <wls:implementation-version>1.0</wls:implementation-version>
  <wls:exact-match>true</wls:exact-match>
</wls:library-ref>
```

The `exact-match` and `version` tags ensure that the `AuctionWebApp` application references the database version of the library. The existing V1 `AuctionWebApp` application deployment still references the memory-based version of the library.

g. Save the file.

6. Modify the application's content.

a. Open the application's `index.jsp` page for editing:

```
$ cd ..
$ gedit index.jsp
```

b. Modify it to display V2 instead of V1:

```
<h1>Welcome to the Auction application: V2</h1>
```

Note: This is useful because it makes it easy for you to see which version of the application you are using. Although the applications may reference different libraries, it is not as visible as simply seeing something different in a browser.

c. Save the file.

7. Continue using the currently released `AuctionWebApp` application from the current web browser. Again, this ensures that there is a session (client) for the existing V1 version of the `AuctionWebApp` application.

8. Deploy the application.

a. Perform the following commands to deploy the `AuctionWebApp` application:

```
$ java weblogic.Deployer -adminurl host01:7001 -username
weblogicUser -password webLogicPassoword -deploy -targets
cluster1 –retiretimeout 300
/practices/part2/practice18-01/AuctionWebApp
```

Note: This deploys the V2 version of the `AuctionWebApp` application because you configured it within its manifest file as `v2`. This causes WebLogic to deploy the new version while leaving the existing version undisturbed.

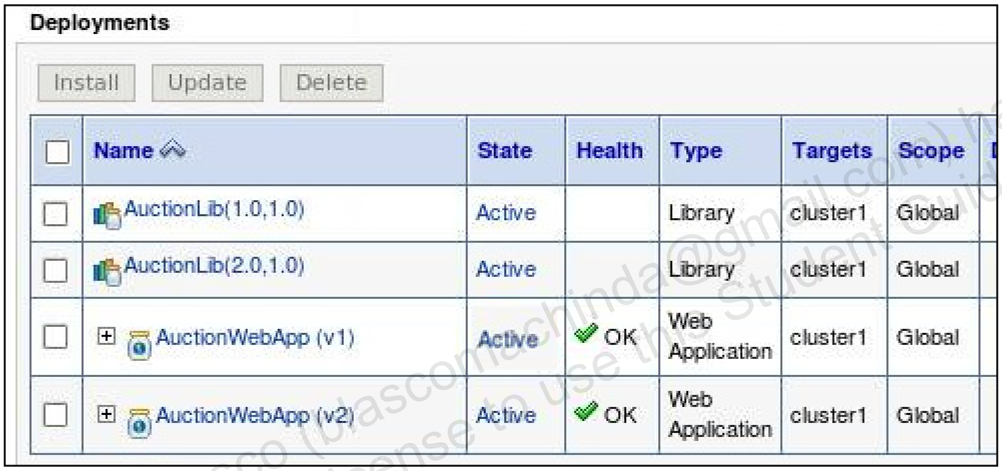
Note: You will need appropriate credentials for the weblogic user and password.

Your output should resemble the following. Again, you can ignore any exceptions relating to the unavailability of `server2`.

```
weblogic.Deployer invoked with options:  -adminurl host01:7001
M-username weblogic -deploy -targets cluster1
/practices/part2/practice18-01/AuctionWebApp

<Apr 26, 2016 4:14:24 PM UTC> <Info> <J2EE Deployment SPI> <BEA-
260121> <Initiating deploy operation for application,
AuctionWebApp [archive: /practices/part2/practice18-
01/AuctionWebApp], to cluster1 .>
```

```
Task 25 initiated: [Deployer:149026]deploy application
AuctionWebApp [Version=v2] on cluster1.
Task 25 deferred: [Deployer:149026]deploy application
AuctionWebApp [Version=v2] on cluster1.
Target state: deploy deferred on Cluster cluster1
java.rmi.RemoteException: [Deployer:149145]Unable to contact
"server2". Deployment is deferred until "server2" becomes
available.
```

b. Verify that your application is deployed by returning to the WebLogic Console and refreshing deployments. Notice that the state for the V1 version of `AuctionWebApp` is *stop Running*. This is because part of production redeployment is keeping the original version of the application running is only for the purposes of allowing existing clients to continue their work undisturbed.



Deployments

□	Name	State	Health	Type	Targets	Scope
□	AuctionLib(1.0,1.0)	Active		Library	cluster1	Global
□	AuctionLib(2.0,1.0)	Active		Library	cluster1	Global
□	⊞ AuctionWebApp (v1)	Active	✓ OK	Web Application	cluster1	Global
□	⊞ AuctionWebApp (v2)	Active	✓ OK	Web Application	cluster1	Global

9. Test the existing client again.

a. Using the existing web browser, browse to the application's `index.jsp` page again. Verify that it still says it is the V1 version of the application. The session timeout for the `AuctionWebApp` application should be 3600 seconds, or one hour. This gives you plenty of time to complete these steps without worrying about the session timing out.

10. Delete the database.

a. Perform the following steps **on host02** to delete the Auction application database. You do this to verify that the new V2 version of `AuctionWebApp` is referencing the database version of `AuctionLib`.

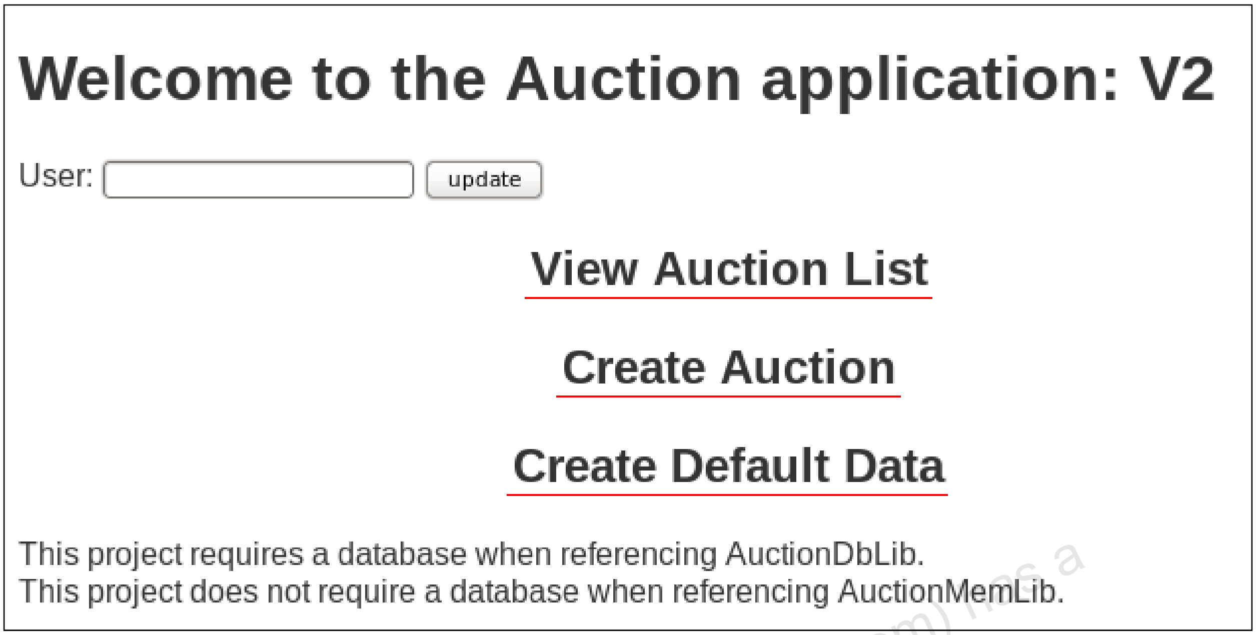
```
$ deleteDatabase.sh
```

11. Test the application.

a. Open a new web browser **on host02** and direct your web browser to the following URL:

```
http://host01:7011/AuctionWebApp/index.jsp
```

Note: Because this is a new web browser on another machine, it is considered a new client to WebLogic. As a result, this client's requests are routed to the newly deployed version of the `AuctionWebApp` application that references the database version of the library. You should see V2 displayed on this page.



Welcome to the Auction application: V2

This project requires a database when referencing AuctionDbLib.
This project does not require a database when referencing AuctionMemLib.

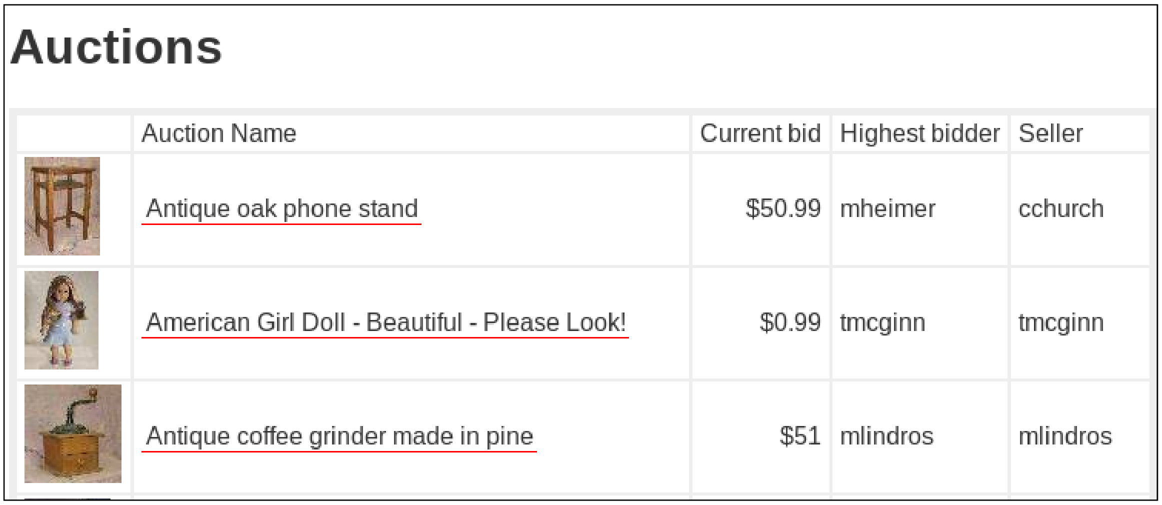
b. First, click *View Auction List* to see the list of auctions. You should get an error and no results. This is because the V2 version of the application is successfully referencing the database version of the library. You just intentionally deleted the Auction database so the application is having a problem now. If you use the original `AuctionWebApp` application on host01, it is still referencing the memory-based version of the library and should still show a list of auctions. This shows that the original V1 `AuctionWebApp` application still references the memory-based library, and the new V2 `AuctionWebApp` version references the database version of the library. You can verify that you have a database error by viewing the output in the server1 console window.

c. Create the database to get the new `AuctionWebApp` version working (**on host02**):

```
$ createDatabase.sh
```

d. Return to the previous page and click the *Create Default Data* link in the browser **on host02** to populate the database data model with data for the Auction application. Follow this by clicking the confirmation to create the data.

e. Click the *Go Home* link, and then click the *View Auction List* link to view the list of auctions stored in the database.



Auctions

	Auction Name	Current bid	Highest bidder	Seller
	Antique oak phone stand	$50.99	mheimer	cchurch
	American Girl Doll - Beautiful - Please Look!	$0.99	tmcginn	tmcginn
	Antique coffee grinder made in pine	$51	mlindros	mlindros

Congratulations! This shows that the V2 version of the Auction application is working properly and is referencing the database version of the library. You can try the V1 client again to see that version V1 of the application is still working as well.

12. Retire `AuctionWebApp V1`.

    IMPORTANT! Do NOT close the web browser that is using the original `AuctionWebApp` V1 application. This is because you will see how the server handles the removal of the V1 version of the application and how the client web browser reacts.

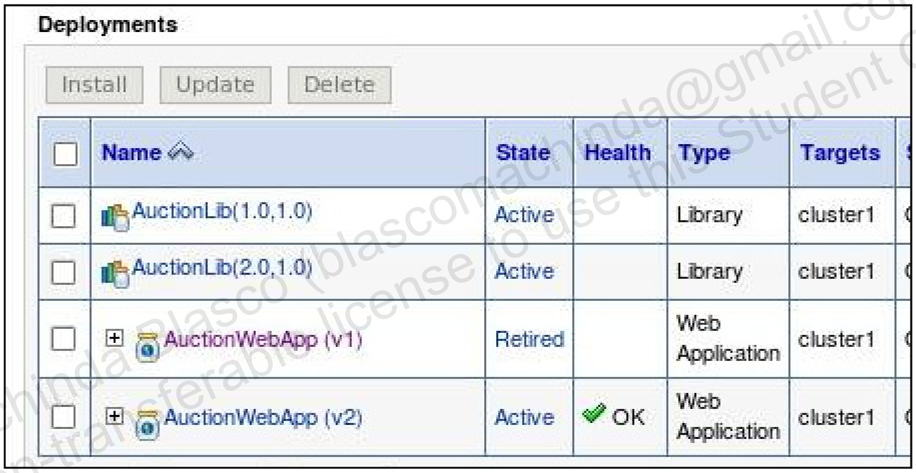
    a. Because the session timeout is an hour, you will just shut down version V1 of the application to cause it to retire.

    b. Select the **Control** tab.

    c. Select `AuctionWebApp (v1)` and click **Stop > Force Stop Now**.

    d. Click **Yes**.

    e. Now you should see that the application's state is changed to retired. If it says prepared, wait a few moments and refresh the page until it says retired.
    Note: The retirement interval was set to 5 minutes (60*5) and could take up to that long to move to retired.



Deployments

□	Name	State	Health	Type	Targets
□	AuctionLib(1.0,1.0)	Active		Library	cluster1
□	AuctionLib(2.0,1.0)	Active		Library	cluster1
□	⊞ AuctionWebApp (v1)	Retired		Web Application	cluster1
□	⊞ AuctionWebApp (v2)	Active	✓ OK	Web Application	cluster1

    f. Use the V1 client to use the Auction application again. Now you should see that the version is V2. This is because version V1 has stopped and is retired so the client now uses V2.

Congratulations! You have successfully deployed versioned applications that reference versioned shared libraries.

13. Shut down the environment.

    a. Press **Ctrl + C** in all terminal windows that are running servers.

    b. Close all terminal windows used for each server.

Practice Solution

Perform the following tasks if you did not complete this practice and want to use the finished solution.

Solution Tasks

1. Open a new terminal window **on host01**:
2. Change the current directory to the current practice folder.
3. Execute the solution script:

   ```
   $ ./solution.sh
   ```

   The solution script performs the following:

   - Executes the setup script for this exercise, which in turn executes the solution script for `practice09-01`
4. Wait for servers **on host01** to start fully.
5. Continue starting at step number 4. Be sure to skip all configuration steps, and instead of copying files from `practice09-01`, use the solution files in the current practice's `solution` folder.

Appendix A: Connecting to the Environment

Chapter 19

Overview

The set of Oracle WebLogic Server 12c courses are offered in multiple formats. Each format requires a different form of connecting to the environment to work with the hands-on practices of the course. To accommodate this, the connection instructions within each course generically tell you to connect to your hosts and leave the connection details to this document. Refer to this document and your instructor whenever you have questions about connecting to your lab environment.

Instructor-Led Training (ILT) or Live Virtual Class (LVC)

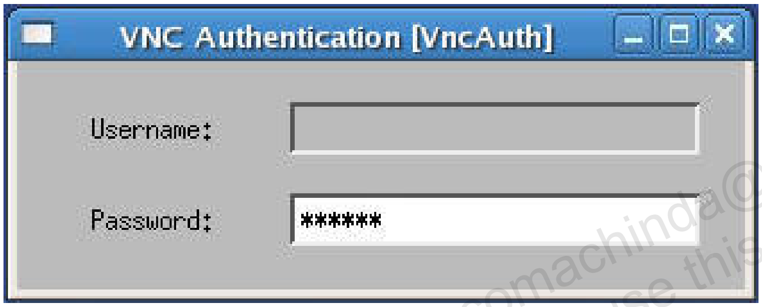
Environment: OVM

Primary Machine Connection: Ask your instructor to provide you with Oracle University's instructions for connecting to your environment.

Virtual Machine Instances: Connectivity is made by clicking the provided VNC shortcuts on the primary machine's desktop. Note that the password entered to connect via VNC is not the operating system password, but rather it is the password used to connect to the VNC server.

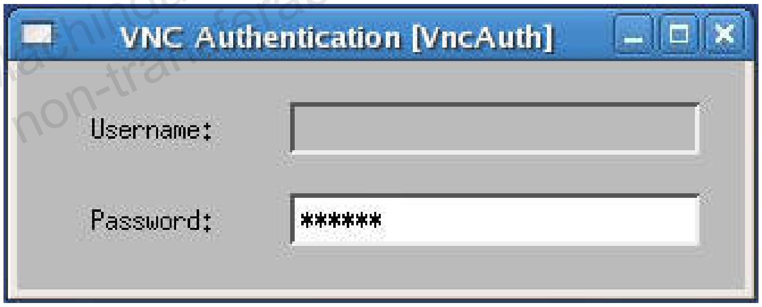
1. Accessing host01.

   From your gateway machine, double-click the **VNC Viewer - host01** icon on the desktop. You connect to host01 as the `oracle` user. (The Username field is not displayed.) Enter **`oracle`** in the Password field (it displays as `******`) and press the Enter key.



2. Accessing host02.

   From your gateway machine, double-click the **VNC Viewer - host02** icon on the desktop. You connect to host02 as the `oracle` user. (The Username field is not displayed.) Enter **`oracle`** in the Password field (it displays as `******`) and press the Enter key.



Training On Demand (TOD)

Environment: OVM

Primary Machine Connection: Ask your instructor to provide you with Oracle University's instructions for connecting to your environment.

Virtual Machine Instances: Ask your instructor to provide you with Oracle University's instructions for connecting to your environment.

Appendix B: Convenience Cheat Sheet

Chapter 20

Using Convenience Scripts and Aliases in this Course

This is a cheat sheet for using the convenience scripts and aliases that are built in to the environment. Using these shortcuts makes using the lab environment easier. These aliases are all defined in the `/home/oracle/.bashrc` file, and are initialized automatically when you open a new terminal window after you set the initial environment.

Shortcut	Type	Description
`install`	Navigation	Navigates to the `$INSTALLDIR` folder: `cd /install`
`app`	Navigation	Navigates to the $APP folder: `cd /u01/app`
`apps`	Navigation	Navigates to the `$LABHOME/part2/apps` folder: `cd /practices/part2/apps`
`labs`	Navigation	Navigates to the `$LABHOME/part2` folder: `cd /practices/part2`
`domains`	Navigation	Navigates to the `domains` folder: `cd /u01/domains/part2`
`ohome`	Navigation	Navigates to the `$ORACLE_HOME` folder: `cd $ORACLE_HOME`

`fmw`	Navigation	Navigates to the Fusion Middleware installation folder (`MW_HOME`):
		`cd /u01/app/fmw`

`wls`	Navigation	Navigates to the WebLogic Server product folder: `cd $MW_HOME/wlserver`
`ide`	Navigation	Navigates to the OEPE/Eclipse folder **(only works on host01):** `cd $APP/oepe`
`bin`	Navigation	Navigates to the lab `bin` folder: `cd $LABHOME/part2/bin`
`wlsd`	Navigation	Navigates to the `wlsadmin` domain folder: `cd /u01/domains/part2/wlsadmin`
`31`	Navigation	Navigates to the `practice03-01` folder: `cd $LABHOME/part2/practice03-01`
`41`	Navigation	Navigates to the `practice04-01` folder: `cd $LABHOME/part2/practice04-01`
`51`	Navigation	Navigates to the `practice05-01` folder: `cd $LABHOME/part2/practice05-01`
`61`	Navigation	Navigates to the `practice06-01` folder: `cd $LABHOME/part2/practice06-01`
`62`	Navigation	Navigates to the `practice06-02` folder: `cd $LABHOME/part2/practice06-02`
`71`	Navigation	Navigates to the `practice07-01` folder: `cd $LABHOME/part2/practice07-01`
`81`	Navigation	Navigates to the `practice08-01` folder: `cd $LABHOME/part2/practice08-01`
`91`	Navigation	Navigates to the `practice09-01` folder: `cd $LABHOME/part2/practice09-01`
`101`	Navigation	Navigates to the `practice10-01` folder: `cd $LABHOME/part2/practice10-01`
`111`	Navigation	Navigates to the `practice11-01` folder: `cd $LABHOME/part2/practice11-01`

121	Navigation	Navigates to the `practice12-01` folder: `cd $LABHOME/part2/practice12-01`
122	Navigation	Navigates to the `practice12-02` folder: `cd $LABHOME/part2/practice12-02`
131	Navigation	Navigates to the `practice13-01` folder: `cd $LABHOME/part2/practice13-01`
132	Navigation	Navigates to the `practice13-02` folder: `cd $LABHOME/part2/practice13-02`
141	Navigation	Navigates to the `practice14-01` folder: `cd $LABHOME/part2/practice14-01`
151	Navigation	Navigates to the `practice15-01` folder: `cd $LABHOME/part2/practice15-01`
161	Navigation	Navigates to the `practice16-01` folder: `cd $LABHOME/part2/practice16-01`
162	Navigation	Navigates to the `practice16-02` folder: `cd $LABHOME/part2/practice16-02`