ORACLE UNIVERSITY

Custom CW: Oracle WebLogic Server 12*c* Administration

Activity Guide

X95181GC10

Edition 1.0 | May 2016

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

ORACLE

Table of Contents

Practices for Lesson 1: Course Overview

Chapter 1

Practices for Lesson 1

There are no practices for this lesson.

Practices for Lesson 2: WebLogic Server: Overview

Chapter 2

Practices for Lesson 2

There are no practices for this lesson.

Practices for Lesson 3: Installing and Patching WebLogic Server

Chapter 3

Practices for Lesson 3: Overview

Practices Overview

In these practices, you install Oracle WebLogic Server 12*c* and apply a patch to it.

Practice 3-1: Installing WebLogic Server

Overview

In this practice, you install the Java Development Kit (JDK) and then WebLogic Server.

Both the JDK and WebLogic Server have already been installed on both hosts. You will rename the installation directory on host01 to gain installation practice. If something goes wrong during the installation on host01, you can revert to the preinstalled version of the products by changing the directories back to their original names.

Assumptions

None

Tasks

1. Connect to host01.
   a. Refer to the instructions in Appendix A: Connecting to the Environment to connect to your machine.
   b. Unless stated otherwise, you will be working within host01 for the remainder of this practice.
2. Rename the installation directories.
   a. Open a Terminal window, and navigate to `/u01/app`.

   Tip: There is a launcher for a Terminal window in the panel at the top of the desktop.

   ```
   $ cd /u01/app
   ```

   b. Rename the FMW installation directory.

   ```
   $ mv fmw fmw-orig
   ```

   c. Create a new, empty `fmw` directory.

   ```
   $ mkdir fmw
   ```

   d. Rename the JDK installation directory.

   ```
   $ mv jdk jdk-orig
   ```

3. Install the JDK.
   a. In the Terminal window, navigate to `/install/java`.

   ```
   $ cd /install/java
   ```

   b. Unzip the tar file by running the `tar` command.

   ```
   $ tar xvf jdk-linux-x64.tar.gz -C /u01/app
   ```

   Command options:

   - `x`: Extract
   - `v`: Verbose output
   - `f`: File to extract
   - `C`: The `tar` command changes the current directory to the one specified before performing any operations. (Note that this is a capital "C.")

   c. Rename the JDK installation directory to something more generic. In a Terminal window, navigate back to `/u01/app`.

   ```
   $ cd /u01/app
   $ mv jdk1.7.0_67 jdk
   ```

Note: The exact name of the directory created by unzipping the tar file may be different, depending upon the version of the JDK. Rename whatever the directory name is to `jdk`.

4. Install WebLogic Server.
   a. In the Terminal window, navigate to `/install/weblogic`.
      `$ cd /install/weblogic`
   b. Set the `JAVA_HOME` variable to where the JDK is installed.
      `$ export JAVA_HOME=/u01/app/jdk`
   c. Set the `PATH` to include the `bin` directory of the JDK.
      `$ export PATH=$JAVA_HOME/bin:$PATH`
   d. Run the Java virtual machine with the `jar` option to run the generic installer.

Note: The exact name of the JAR file may be different. Use the name of the generic installer JAR file in your system. The `-D` option ensures that the WebLogic installer performs a 64-bit product installation.

```
$ java -d64 -jar fmw_12.1.3.0.0_wls.jar
Extracting files.....
...
```

Note: If any of the installation requirements fail, continue with the installation anyway by answering **y** when you are asked if you want to continue. There really is not a problem; the installer might get confused because host01 is a virtual machine.

   e. After the files are extracted, the graphical installer appears. Use the guidelines in the following table to install the software:

Step	Window/Page Description	Choices or Values
a.	Specify Oracle Inventory Directory	This screen appears when you install the first Oracle product. If it appears, enter `/u01/app/oraInventory` and click **Next**. If it does not appear, move to the next step.
b.	Welcome	Click **Next**.
c.	Installation Location	Enter `/u01/app/fmw` and click **Next**.
d.	Installation Type	Select **Complete with Examples**. Click **Next**.
e.	Prerequisite Checks	When all prerequisite checks are 100% successful, click **Next**.
f.	Security Updates	Deselect **I wish to receive security updates via My Oracle Support**. (In your training environment, you do not want to associate the installations with a specific user.) Click **Next**. In the dialog box that opens, click **Yes**.
g.	Installation Summary	Click **Install**.
h.	Installation Progress	Note the progress and successful completion of the installation. (This step may take several minutes.) Click **Next**.

Step	Window/Page Description	Choices or Values
i.	Installation Complete	Deselect **Automatically Launch the Quickstart Configuration Wizard**. Click **Finish**.

f. Close the Terminal window.

Practice Solution: Installing WebLogic Server

There is no scripted solution for this practice. If you did not do the practice, there is nothing for you to do, the products are already installed.

If you started the practice, but did not complete it, you revert to the preinstalled version of the products by deleting any new directories you created and renaming the original installation directories back to their previous names.

Assumptions

You started the practice but did not complete it.

Solution Tasks

Important: Do these tasks only if you started the practice but did not complete it.

1. Connect to host01.
   a. Refer to the instructions in Appendix A: Connecting to the Environment to connect to your machine.
   b. Unless stated otherwise, you will be working within host01 for the remainder of this practice.
2. Check that the preinstalled FMW directory is there. Delete the new FMW installation directory that you created. Rename the preinstalled directory back to its original name.
   a. Open a Terminal window, and navigate to `/u01/app`. Then, list the directories to ensure that the `fmw-orig` directory is present. Check that this preinstalled directory has valid contents.

```
$ cd /u01/app
$ ls
fmw ... fmw-orig ...
$ ls fmw-orig
cfgtoollogs  coherence ...
```

   b. Ensure that you are in the proper directory. Remove the new installation directory that you created.

```
$ pwd
/u01/app
$ rm -rf fmw
```

   Warning: This command permanently deletes the directory, all its subdirectories, and files. Use with caution.

   c. Rename the preinstalled directory back to its original name.

```
$ mv fmw-orig fmw
```

3. Check that the preinstalled JDK directory is there. Delete the new JDK directory that you created. Rename the preinstalled directory back to its original name.
   a. In a Terminal window, navigate to `/u01/app`. Then list the directories to ensure that the `jdk-orig` directory is present.

```
$ cd /u01/app
$ ls
jdk ... jdk-orig ...
```

   b. Ensure that you are in the proper directory. Remove the new installation directory that you created.

```
$ pwd
/u01/app
$ rm -rf jdk
```

Warning: This command permanently deletes the directory, all its subdirectories, and files. Use with caution.

c. Rename the preinstalled directory back to its original name.

```
$ mv jdk-orig jdk
```

4. Close the Terminal window.

Practice 3-2: Patching WebLogic Server

Overview

In this practice, you apply a WebLogic patch to the WebLogic Server environment installed in the previous practice. Optionally, you can also roll back the patch at the end of this practice.

Assumptions

You completed “Practice 3-1: Installing WebLogic Server” and WebLogic Server has been successfully installed.

No instances of WebLogic Server are running.

Tasks

1. Connect to host01.

   Note: If a connection for host01 is already available, you can continue to use it.

2. Access the patch zip file (`p19234430_121300_Generic.zip`).
   a. Open a Terminal window, and navigate to `/install/weblogicpatch`.

      ```
      $ cd /install/weblogicpatch
      ```

3. Define the `PATCH_TOP` environment variable.
   a. Define the PATCH_TOP variable and assign it the value `/install/weblogicpatch`.

      ```
      $ export PATCH_TOP=/install/weblogicpatch
      ```

   b. Print the variable to the screen to make sure you set up `PATCH_TOP` correctly.

      ```
      $ echo $PATCH_TOP
      /install/weblogicpatch
      ```

4. Define (or redefine) the `ORACLE_HOME` environment variable.
   a. In a Fusion Middleware environment, the `ORACLE_HOME` variable should point to the top directory where WebLogic Server has been installed (`/u01/app/fmw`).

      ```
      $ export ORACLE_HOME=/u01/app/fmw
      ```

   b. Print the variable to the screen to make sure you set up `ORACLE_HOME` correctly.

      ```
      $ echo $ORACLE_HOME
      /u01/app/fmw
      ```

5. Define (or redefine) the `JAVA_HOME` environment variable.
   a. The `JDK` used to run OPatch must be the `JDK` installed in `$ORACLE_HOME`.

      ```
      $ export JAVA_HOME=/u01/app/jdk
      ```

   b. Print the variable to the screen to make sure you set up `JAVA_HOME` correctly.

      ```
      $ echo $JAVA_HOME
      /u01/app/jdk
      ```

6. Add the directory where the OPatch utility resides to the Linux `PATH`.
   a. The executable shell file `opatch` resides in the `OPatch` directory. The `OPatch` directory must be added to the `PATH` so that the shell will be able to invoke it.

      ```
      $ export PATH=$ORACLE_HOME/OPatch:$PATH
      ```

   Note: Notice that the “`O`” and “`P`” in the “`OPatch`” directory name are capitalized.

   b. Confirm the `opatch` executable appears in your system `PATH`.

      ```
      $ which opatch
      ```

```
/u01/app/fmw/OPatch/opatch
```

7. Unzip the patch zip file into the `PATCH_TOP` directory.
   a. Use the `unzip` utility to expand the zip file `p19234430_121300_Generic.zip`.

   Note: If asked to replace an existing file, enter **`A`** for "`All`."

```
$ unzip -d $PATCH_TOP p19234430_121300_Generic.zip
Archive:  p19234430_121300_Generic.zip
   creating: 19234430/
   creating: 19234430/files/
   creating: . . .
  inflating:
19234430/files/oracle.fmwconfig.common.wls.shared/12.1.3.0.0/fmw
config.common.symbol/modules/com.oracle.cie.config-
wls_8.1.0.0.jar/com/oracle/cie/domain/DomainNodeManagerHelper.cl
ass
  inflating: 19234430/README.txt
   creating: 19234430/etc/
   creating: 19234430/etc/config/
  inflating: 19234430/etc/config/actions.xml
  inflating: 19234430/etc/config/inventory.xml
```

   b. Set your current directory to the directory where the patch is located.

```
$ cd $PATCH_TOP/19234430
```

   c. Make sure that your default directory is correct.

```
$ pwd
/install/weblogicpatch/19234430
```

8. Before applying the patch, view the patch README file.
   a. In the patch directory, find the README file.

```
$ ls README*
README.txt
```

   b. Open the file in the editor. Briefly look at the instructions. The directions in this practice are based on the directions in this file.

```
$ gedit README.txt
```

   c. Close the editor without making any changes to the file.

9. Run the OPatch utility with the `lsinventory` parameter, which finds the Oracle WebLogic inventory and prints out all the components found.
   a. Run `opatch` with the `lsinventory` parameter.

```
$ opatch lsinventory -jdk $JAVA_HOME
Oracle Interim Patch Installer version...
. . .
Oracle Home       : /u01/app/fmw
Central Inventory : /u01/app/oraInventory
   from           : /u01/app/fmw/oraInst.loc
. . .
--------------------------------------------
```

```
There are no Interim patches installed in this Oracle Home.

--------------------------------------------
OPatch succeeded.
$
```

Note: As expected, the `lsinventory` output states that no interim patches are currently installed.

10. Apply the patch.
    a. Run the OPatch utility to apply the patch. When you are asked if the local system is ready for patching, enter **`y`**.

Note: This can take a while to complete.

```
$ opatch apply -jdk $JAVA_HOME
Oracle Interim Patch Installer version ...
...

Oracle Home       : /u01/app/fmw
Central Inventory : /u01/app/oraInventory
   from           : /u01/app/fmw/oraInst.loc

...

Applying interim patch '19234430' to OH '/u01/app/fmw'
Verifying environment and performing prerequisite checks...
All checks passed.

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...
...
Verifying the update...
Patch 19234430 successfully applied
...
OPatch succeeded.
$
```

11. Rerun the OPatch utility with the `lsinventory` parameter. This time `opatch` should find the interim patch you just applied, and print its information to the screen.
    a. Run `opatch` with `lsinventory`.

```
$ opatch lsinventory -jdk $JAVA_HOME
```

```
Oracle Interim Patch Installer version ...
...
Oracle Home       : /u01/app/fmw
Central Inventory : /u01/app/oraInventory
   from           : /u01/app/fmw/oraInst.loc
...
--------------------------------------------
Interim patches (1) :

Patch  19234430     : applied on ...
Unique Patch ID:  17945590
Patch description:  "One-off"
   Created on 14 Aug 2014, 13:47:02 hrs Asia/Calcutta
   Bugs fixed:
     19234430
--------------------------------------------------------
OPatch succeeded.
$
```

Note: This time the `lsinventory` output states that there is one interim patch, the one you just applied, 19234430.

12. (Optional) Roll back the applied patch. If an issue occurred while applying a patch, your system could be left in an unstable state, with a few files already uploaded and other files missing. In this case, the best course of action is to roll back the patch.

    Note: This can take a while to complete.

    a. Use OPatch to roll back the patch. When you are asked whether the local system is ready for patching, enter **y**.

```
$ opatch rollback -id 19234430 -jdk $JAVA_HOME
Oracle Interim Patch Installer version ...
...
RollbackSession rolling back interim patch '19234430' from OH
'/u01/app/fmw'

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
...
RollbackSession removing interim patch '19234430' from inventory
...
OPatch succeeded.
```

```
$
```

13. (Optional) Make sure that the patch has been removed.
    a. Run the OPatch utility with the `lsinventory` parameter one more time to show that no interim patches are found.

```
$ opatch lsinventory -jdk $JAVA_HOME
...
--------------------------------------------


There are no Interim patches installed in this Oracle Home.


--------------------------------------------


OPatch succeeded.
$
```

    Note: As expected, the `lsinventory` output states that no interim patches are installed. The interim patch installed and subsequently rolled back is no longer there.
14. Close the Terminal window.

Practice Solution: Patching WebLogic Server

There is no solution for this practice. This practice is not required and can be skipped.

Practices for Lesson 4: Creating Domains

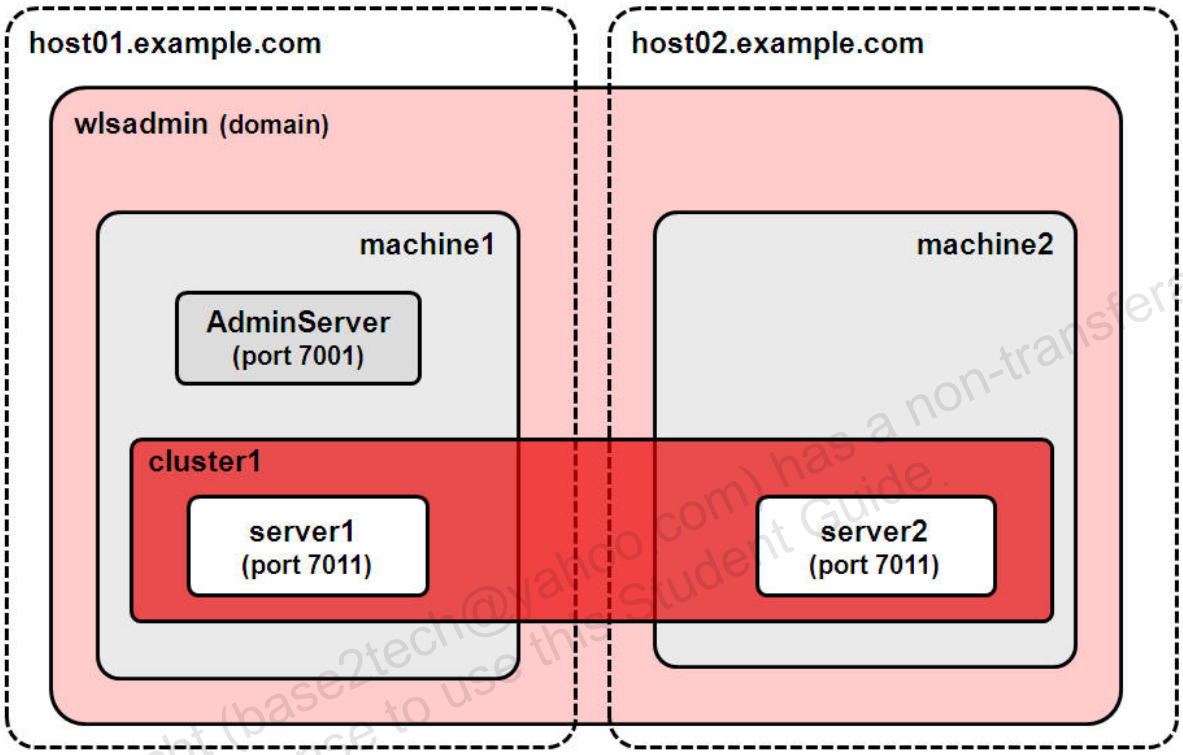
Chapter 4

Practices for Lesson 4: Overview

Practices Overview

In these practices, you create a new WebLogic Server domain. You then move the domain to a second machine by using the pack and unpack utilities.

The following is a diagram of the domain:



Practice 4-1: Creating a New Domain

Overview

In this practice, you create a new WebLogic Server domain by using the Configuration Wizard. As you create the domain, you create and configure some domain resources.

Assumptions

You completed “Practice 3-1: Installing WebLogic Server” and the installation was successful.

Completing “Practice 3-2: Patching WebLogic Server” is not required for this practice, but it is OK if you did it.

Tasks

1. Connect to host01.
   a. Unless stated otherwise, you will be working within host01 for the remainder of this practice.
2. Run the Configuration Wizard in graphical mode.
   a. Open a new Terminal window. Navigate to the location of the Configuration Wizard script, the `common/bin` directory under the Oracle Common directory: `/u01/app/fmw/oracle_common/common/bin`.

      ```
      $ cd /u01/app/fmw/oracle_common/common/bin
      ```

      Note: Remember that you can use the Tab key to auto-complete directory and file names as you type them.
   b. Run the Configuration Wizard.

      ```
      $ ./config.sh
      ```
   c. After the graphical wizard opens, use the guidelines in the following table to create the domain:

Step	Window/Page Description	Choices or Values
a.	Create Domain	Select **Create a new domain**. Enter the Domain Location: **`/u01/domains/part1/wlsadmin`** Click **Next**.
b.	Templates	Ensure that **Create Domain Using Product Templates** is selected. Click **Next**. **Note: Basic WebLogic Server Domain** is selected automatically and cannot be deselected.
c.	Administrator Account	Enter these values for the following fields: • Name: **`weblogic`** • Password: **`Welcome1`** • Confirm Password: **`Welcome1`** **Note:** The password displays as ●●●●●●●●. Click **Next**.
d.	Domain Mode and JDK	Under Domain Mode, select **Production**.

Step	Window/Page Description	Choices or Values
		Under JDK, ensure that **Oracle HotSpot** is selected (whatever version you have). Click **Next**.
e.	Advanced Configuration	Select **Administration Server** and **Managed Servers, Clusters, and Coherence**. Click **Next**. **Note:** Although Node Manager can be set up at this time, you will wait to configure it after it has been discussed.
f.	Administration Server	Enter or select these values for the following fields: • Server Name: **`AdminServer`** • Listen Address: **`host01.example.com`** • Listen Port: **`7001`** • Enable SSL: Deselected • SSL Listen Port: (Disabled) Click **Next**.
g.	Managed Servers	Click the **Add** button. Then, in the row, enter or select these values for the following fields: • Server Name: **`server1`** • Listen Address: **`host01.example.com`** • Listen Port: **`7011`** • Enable SSL: Deselected • SSL Listen Port: (Disabled) Click the **Add** button again. Then, in the row, enter or select these values for the following fields: • Server Name: **`server2`** • Listen Address: **`host02.example.com`** • Listen Port: **`7011`** • Enable SSL: Deselected • SSL Listen Port: (Disabled) Click **Next**.
h.	Clusters	Click the **Add** button. Then, in the row, enter or select these values for the following fields: • Cluster Name: **`cluster1`** • Cluster Address: (leave blank) Click **Next**.
i.	Assign Servers to Clusters	Under Servers, select **`server1`**. Then, under Clusters, ensure **`cluster1`** is selected and click the right arrow button. Do the same thing to move **`server2`** under

Step	Window/Page Description	Choices or Values
		cluster1. The Clusters column should look like this:  Click **Next**.
j.	Machines	Click the **Machine** tab. Click the **Add** button. Then, in the row, enter or select these values for the following fields: • Name: **machine1** • Node Manager Listen Address: **host01.example.com** • Node manager listen port: **5556** Click the **Add** button again. Then, in the row, enter or select these values for the following fields: • Name: **machine2** • Node Manager Listen Address: **host02.example.com** • Node manager listen port: **5556** Click **Next**.
k.	Assign Servers to Machines	Under Machines, select **machine1**. Then under Servers, select **AdminServer** and click the right arrow button. Under Machines, ensure **machine1** is still selected. Then under Servers, select **server1** and click the right arrow button. Under Machines, select **machine2**. Then under Servers, select **server2** and click the right arrow button. The Machines column should look like this:  Click **Next**.
l.	Configuration Summary	Review the domain. Click **Create**.

Step	Window/Page Description	Choices or Values
m.	Configuration Progress	When the progress bar reaches 100%, click **Next**.
n.	Configuration Success	Click **Finish**.

d. The Configuration Wizard closes. Close the terminal window.

Practice Solution: Creating a new Domain

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

Assumptions

You completed “Practice 3-1: Installing WebLogic Server” and the installation was successful or you did not do Practice 3-1at all (the products are pre-installed).

Solution Tasks

1. Connect to host01. If you partially completed the practice, delete the domain directory you created.
   a. Open a Terminal window and navigate to the parent directory of the domain directory. Then remove the domain directory.
      ```
      $ cd /u01/domains/part1
      $ rm -rf wlsadmin
      ```
      Warning: This command permanently deletes the directory, all its subdirectories, and files. Use with caution.
      Note: If you did not start the practice, there is no domain directory to delete.
2. Run the solution script.
   a. In a Terminal window, navigate to the practice directory.
      ```
      $ cd /practices/part1/practice04-01
      ```
   b. Run the solution script.
      ```
      $ ./solution.sh
      ```
      Note: This script starts the WebLogic Scripting Tool (WLST) passing it the WLST script `create_domain.py`, which creates the new domain for you. The WLST script prints out messages. When it writes the domain, it can take some time, so be patient and do not close the Terminal window until you see the message:
      ```
      >>>Domain created successfully at /u01/domains/part1/wlsadmin
      ```
3. Close the Terminal window.

Practice 4-2: Copying a Domain to a New Machine

Overview

In this practice, you use the pack and unpack utilities to copy the domain to another machine, so a managed server of the domain can run there.

Assumptions

You completed “Practice 4-1: Creating a New Domain” successfully.

Tasks

1. Connect to host01.
   a. Unless stated otherwise, you will be working within host01 for the remainder of this practice.
2. Create a managed server template by using the pack utility and the domain you just created.
   a. Open a Terminal window and navigate to the location of the `pack.sh` script.
      ```
      $ cd /u01/app/fmw/oracle_common/common/bin
      ```
   b. Run the `pack.sh` script with the options to use your domain and create a managed server template in your home directory:

   Note: Enter the command on one line. It is on multiple lines here due to space limitations.
      ```
      $ ./pack.sh -domain=/u01/domains/part1/wlsadmin
                  -template=/home/oracle/managedserver.jar
                  -template_name=wlsadmin_managed
                  -managed=true
      << read domain from "/u01/domains/part1/wlsadmin"
      >>  succeed: read domain from "/u01/domains/part1/wlsadmin"
      ..........................................
      >>  succeed: write template to "/home/oracle/managedserver.jar"
      << close template
      >>  succeed: close template
      $
      ```
3. Send the template JAR file over to host02.
   a. In the Terminal window, navigate to the location of the new JAR file.
      ```
      $ cd /home/oracle
      ```
   b. Use SFTP to put the JAR file on host02. You use the username `oracle`. If you are warned about the authenticity of host02 and asked if you want to continue, enter **`yes`**. If asked for the password of the user, it is also **`oracle`**.
      ```
      $ sftp oracle@host02.example.com
      Connecting to host02.example.com...
      The authenticity of host...
      ...
      Are you sure you want to continue connecting (yes/no) ? yes
      ...
      sftp> cd /home/oracle
      ```

```
sftp> pwd
Remote working directory: /home/oracle
sftp> put managedserver.jar
Uploading managedserver.jar to /home/oracle/managedserver.jar
managedserver.jar           100% ...
sftp>
```

c. Exit SFTP.

```
sftp> bye
$
```

d. Close the Terminal window.

4. Connect to host02.
   a. Unless stated otherwise, you will be working within host02 for the remainder of this practice.
5. Create a copy of the domain on host02 by using the unpack utility.
   a. Open a Terminal window. Verify that the domain does not already exist on this host.

```
$ ls /u01/domains/part1
$
```

Note: Nothing displays because there are currently no files or directories there.

   b. Navigate to the location of the `unpack.sh` script.

```
$ cd /u01/app/fmw/oracle_common/common/bin
```

   c. Run the `unpack.sh` script with the options to use the managed server template to create your domain on this host.

Note: Enter the command on one line. It is on multiple lines here due to space limitations.

```
$ ./unpack.sh -domain=/u01/domains/part1/wlsadmin
              -template=/home/oracle/managedserver.jar
<< read template from "/home/oracle/managedserver.jar"
>>  succeed: read template from "/home/oracle/managedserver.jar"
<< set config option DomainName to "wlsadmin"
>>  succeed: set config option DomainName to "wlsadmin"
<< write Domain to "/u01/domains/part1/wlsadmin"
.................................................
>>  succeed: write Domain to "/u01/domains/part1/wlsadmin"
<< close template
>>  succeed: close template
$
```

Note: Notice that the domain on host02 is placed in the same location as it is on host01. This is the recommended practice.

6. Check out the domain directories on host02.
   a. In the same Terminal window, navigate to the domain directory and look around by using the `cd` and `ls` commands.

```
$ cd /u01/domains/part1/wlsadmin
$ ls
```

```
bin  fileRealm.properties  lib  security  startWebLogic.sh
...
$ cd bin
$ ls
nodemanager  setDomainEnv.sh ...
...
```

7. Close the Terminal window.

Practice Solution: Copying a Domain to a New Machine

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

Assumptions

You completed “Practice 4-1: Creating a New Domain” successfully.

Solution Tasks

1. Connect to host02.

   Note: Notice you are going to host02 (not host01).

2. If you partially completed the practice, delete the domain directory on host02. (If you did not start the practice, skip this task.)

   Note: Do *not* delete the domain directory on host01.

   a. Open a Terminal window and navigate to the parent directory of the domain directory. Then remove the domain directory.

   ```
   $ cd /u01/domains/part1
   $ rm –rf wlsadmin
   ```

   Warning: This command permanently deletes the directory, all its subdirectories, and files. Use with caution.

   Note: If you did not start the practice, there is no domain directory to delete.

3. Run the solution script.

   a. In a Terminal window, navigate to the practice directory.

   ```
   $ cd /practices/part1/practice04-02
   ```

   b. Run the solution script.

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

   Note: This script runs the unpack utility, which uses a previously created managed server template, to create the domain on host02. The command can take a little while, so wait for the messages that unpack succeeded in writing the domain and that the template file was closed, before you close the Terminal window.

4. Close the Terminal window.

Practices for Lesson 5: Starting Servers

Chapter 5

Practices for Lesson 5: Overview

Practices Overview

In this practice, you start and stop instances of WebLogic Server by using the scripts supplied with the domain. You also update a start script to modify the `CLASSPATH` of the servers.

Practice 5-1: Starting and Stopping Servers

Overview

In this practice, you start the administration server by using the domain start script. You create a boot identity file for the administration server so you do not have to enter a username and password each time you start and stop the server. You start a managed server by using the domain start script. You also stop servers by using domain stop scripts. Finally, you update a script to add to the servers' `CLASSPATH`.

Assumptions

You completed "Practice 4-1: Creating a New Domain" and "Practice 4-2: Copying a Domain to a New Machine" so that the domain is created and installed on both hosts.

Tasks

1. Connect to host01.
   a. Unless stated otherwise, you will be working within host01 for the remainder of this practice.
2. Start the administration server by using the script supplied with the domain.
   a. Open a Terminal window and navigate to the domain directory. Then run the `startWebLogic.sh` script.

      ```
      $ cd /u01/domains/part1/wlsadmin
      $ ./startWebLogic.sh
      ```

      Note: Remember that you can use the Tab key to auto-complete directory and file names as you type them.
   b. When prompted for the username, enter **`weblogic`**. When prompted for the password, enter **`Welcome1`**:

      ```
      ...
      Enter username to boot WebLogic server: weblogic
      Enter password to boot WebLogic server: Welcome1
      ...
      ```

      Note: The password does not display.
   c. Messages will appear in the window. When the server successfully completes its startup process, it writes the following message to the window:

      ```
      <The server started in RUNNING mode>
      ```

      Note
      - To remember that this is the window in which the administration server is running, you might want to change the title of the Terminal window. In the window's menu, select **Terminal**, and then **Set Title**. Enter the title of **`AdminServer`** and click **OK**.
      - If you close the window in which a server is running, you kill the server. Minimize the window, if you want.
3. Create a boot identity file for the administration server.
   a. Open a new Terminal window and navigate to the AdminServer directory in the domain:

      ```
      $ cd /u01/domains/part1/wlsadmin
      $ cd servers/AdminServer
      ```

b. Create the `security` subdirectory under the `AdminServer` directory:

```
$ mkdir security
```

c. Create a text file in this security directory called `boot.properties` and edit it. In this example, the gedit text editor is used.

```
$ cd security
$ gedit boot.properties
```

d. Add the following two lines to the file:

```
username=weblogic
password=Welcome1
```

e. Save the file (select **File** > **Save**, or click **Ctrl** + **S**). Close the text editor.

f. Leave the terminal window open for the next task.

4. Stop the administration server and restart it. Observe the server's use of the boot identity file.

a. In the Terminal window, navigate to the `bin` directory under the domain directory, and run the script to stop the administration server:

```
$ cd /u01/domains/part1/wlsadmin/bin
$ ./stopWebLogic.sh
```

Note

- Notice that the script uses the WebLogic Scripting Tool (WLST).
- You can also stop a server by pressing **Ctrl + C** in the window in which it is running.

b. When the stop script finishes, find the Terminal window in which the administration server was running. You can see that it has been shut down. Close this window.

c. Return to the other Terminal window, navigate to the domain directory, and once again run the script to start the administration server:

```
$ cd /u01/domains/part1/wlsadmin
$ ./startWebLogic.sh
...
```

Note: This time you are *not* prompted for the username and password.

d. Messages will appear in the window. When the server successfully completes its startup process, it writes the following message to the window:

```
<The server started in RUNNING mode>
```

Note

- If you look through the messages in the window, you can see the message about storing the boot identity: `<Storing boot identity in the file: ...>`. It was at this point that the server found the `boot.properties` file, used the credentials in it, and because those credentials were valid, encrypted the credentials and copied over the plain text in the file.
- You may have noticed that the stop script you used earlier did not prompt you for the username and password. That script normally does, but it too can use the

`boot.properties` file. It, however, does not encrypt the username and password, as the start script does.

e. Leave the administration server running. Give the Terminal window the title of `AdminServer` and minimize it, if you want.

5. Start a managed server by using the script supplied with the domain.

a. In a new Terminal window, navigate to the `bin` directory under the domain directory, and run the script to start the managed server called `server1`:

```
$ cd /u01/domains/part1/wlsadmin/bin
$ ./startManagedWebLogic.sh server1 host01.example.com:7001
```

Note: Notice that this script takes two parameters: the name of the managed server and the URL of the domain's administration server.

b. When prompted for the username, enter **`weblogic`**. When prompted for the password, enter **`Welcome1`**:

```
...
Enter username to boot WebLogic server: weblogic
Enter password to boot WebLogic server: Welcome1
```

Note: The password does not display.

c. Messages will appear in the window. When the server successfully completes its startup process, it writes the following message to the window:

```
<The server started in RUNNING mode>
```

Note

- To remember that this is the window in which server1 is running, you might want to change the title of the Terminal window. In the window's menu, select **Terminal**, and then **Set Title**. Enter the title of **`server1`** and click **OK**.
- If you close the window in which a server is running, you kill the server. Minimize the window, if you want.
- If you want to skip being prompted for the username and password when starting this server too, create a `boot.properties` file for it as you did for the administration server. (Create the `security` directory under the `server1` directory and create a new `boot.properties` file there.)

d. Leave the managed server running.

e. Minimize the host01 desktop window to use it later.

6. Start a managed server on the other host, host02.

a. Connect to host02.

b. Open a Terminal window, navigate to the `bin` directory under the domain directory, and run the script to start the managed server called `server2`:

```
$ cd /u01/domains/part1/wlsadmin/bin
$ ./startManagedWebLogic.sh server2 host01.example.com:7001
```

Note: Notice that this script takes two parameters: the name of the managed server and the URL of the domain's administration server.

c. When prompted for the username, enter **`weblogic`**. When prompted for the password, enter **`Welcome1`**:

```
...
Enter username to boot WebLogic server: weblogic
```

```
Enter password to boot WebLogic server: Welcome1
```

Note: The password does not display.

d. Messages will appear in the window. When the server successfully completes its startup process, it writes the following message to the window:

```
<The server started in RUNNING mode>
```

Note

- You might want to set the title of the Terminal window to **server2.**
- If you close the window in which the server is running, you kill the server. Minimize the window, if you want.
- If you want to, you could create a `boot.properties` file for this server, too.

e. Leave the managed server running.

7. Add a new JAR file to the `CLASSPATH` of all servers.

a. Return to the host01 desktop.

Note: If you closed the window, then you must open it again.

b. Open a new Terminal window and copy the JAR file from the current practice directory to the `/home/oracle` directory.

```
$ cd
$ cp /practices/part1/practice05-01/sample.jar .
```

c. Navigate to the `bin` directory of the domain and create the `setUserOverrides.sh` file.

Note: The `setUserOverrides.sh` file is invoked by the `setDomainEnv.sh` script if the file exists. It is a best practice to set environmental changes in the override file instead of making changes directly to WebLogic Server startup scripts. This avoids your changes getting overwritten when new domain templates or Oracle product installations are introduced that affect your domains.

```
$ cd /u01/domains/part1/wlsadmin/bin
$ gedit setUserOverrides.sh
```

d. Add the following line to the file:

```
$ export PRE_CLASSPATH=/home/oracle/sample.jar
```

e. Save the file and exit the editor.

8. Use SFTP to copy the updated script and the JAR file to the other host.

a. In the same Terminal window, start SFTP and copy the updated script to other host (replacing the script that is there). Also, put the JAR file onto the other host (in the same relative location).

```
$ sftp oracle@host02.example.com
Connecting to host02.example.com...
sftp> cd /u01/domains/part1/wlsadmin/bin
sftp> pwd
Remote working directory: /u01/domains/part1/wlsadmin/bin
sftp> put setDomainEnv.sh
Uploading setDomainEnv.sh to ...
setDomainEnv.sh                100% ...
sftp> cd /home/oracle
```

```
sftp> pwd
Remote working directory: /home/oracle
sftp> put /home/oracle/sample.jar
Uploading /home/oracle/sample.jar to /home/oracle/sample.jar
/home/oracle/sample.jar     100% ...
sftp> bye
$
```

9. Stop and restart a server. These instructions will use `server1`.
   a. In a Terminal window, navigate to the `bin` directory under the domain directory, and run the script to stop the managed server, `server1`. If you created a boot identity file for this server, you will not be prompted for the username and password. If you did not, when prompted for them, enter **`weblogic`** and **`Welcome1`**, respectively.

```
$ cd /u01/domains/part1/wlsadmin/bin
$ ./stopManagedWebLogic.sh server1 host01.example.com:7001
...
```

   Note: Notice that this script takes two parameters: the name of the managed server and the URL for the domain's administration server.

   b. From a Terminal window, restart `server1`. You can use the window you were in, the window in which the server was previously running, or a new window. Again, if prompted for the username and password, enter **`weblogic`** and **`Welcome1`**, respectively.

```
$ cd /u01/domains/part1/wlsadmin/bin
$ ./startManagedWeblogic.sh server1 host01.example.com:7001
...
<The server started in RUNNING mode>
```

   c. Leave the window open in which the server is running.

10. Notice the `CLASSPATH` update.
   a. Scroll up in the window in which the server is running until you find the print out of the `CLASSPATH`.

```
...
CLASSPATH=/home/oracle/sample.jar:...
...
```

   Note: Notice the new JAR file at the front of the `CLASSPATH`.

   b. Minimize the server window, if you want.
   c. Leave all servers running for now.

   Note: Only one server was stopped and restarted, so only that server picked up the addition to the `CLASSPATH`. The others will pick it up the next time they are stopped and started again. Because the JAR file code is not used by anything, it does not matter that the other servers do not yet have access to it.

Practice Solution: Starting and Stopping Servers

There is no solution for this practice. It primarily involves starting and stopping servers. The modification of the domain script is not required, so not completing this practice has no effect on completing subsequent practices.

Practices for Lesson 7: Configuring JDBC

Chapter 7

Practices for Lesson 7: Overview

Practices Overview

In this practice, you create and configure a JDBC data source.

Practice 7-1: Configuring a JDBC Data Source

Overview

In this practice, you create and configure a generic JDBC data source by using the administration console.

Assumptions

You completed "Practice 5-1: Starting and Stopping Servers."

All servers in the domain are currently running.

Note: If you did not complete "Practice 5-1: Starting and Stopping Servers," but completed "Practice 4-1: Creating a New Domain," and "Practice 4-2: Copying a Domain to a New Machine," and then start the servers in the domain, you can still do this practice.

Tasks

1. Connect to host01.
2. Open the WebLogic Server administration console.
3. Create a generic JDBC data source.
   a. Lock the configuration. In the Change Center, click the **Lock & Edit** button.
   b. In the Domain Structure, expand **Services** (click the "**+**" sign) and then select **Data Sources**.
   c. On the right, above the Data Sources table, click the **New** button and select **Generic Data Source**.
   d. The "create a data source" wizard starts. Use the guidelines in the following table to create a new generic data source:

Step	Window/Page Description	Choices or Values
a.	Create a New JDBC Data Source: JDBC Data Source Properties (page 1)	Enter or select these values for the following fields: • Name: **`datasource1`** • JNDI Name: **`datasource1`** • Database Type: **`Oracle`** **Note:** The name and the JNDI name do not have to match, that is just the choice made in this example. Click **Next**.
b.	Create a New JDBC Data Source: JDBC Data Source Properties (page 2)	Select this value for the following field: • Database Driver: **`*Oracle's Driver (Thin XA) for Instance connections; Versions:Any`** Click **Next**.
c.	Create a New JDBC Data Source: Transaction Options	Click **Next**. **Note:** Because an XA driver was selected, there are no transaction options to choose.
d.	Create a New JDBC Data Source: Connection	Enter or select these values for the following fields: • Database Name: **`orcl`**

Step	Window/Page Description	Choices or Values
	Properties	**Note:** That is a lowercase "L" on the end. • Host Name: `host02.example.com` • Port: `1521` • Database User Name: `oracle` • Password: `Welcome1` • Confirm Password: `Welcome1` • oracle.jdbc.DRCPConnectionClass: (leave blank) Click **Next**.
e.	Create a New JDBC Data Source: Test Database Connection	Click the **Test Configuration** button. When the message "Connection test succeeded." displays, click **Next**. **Note:** If the test fails, click the **Back** button and check your entries. Correct any mistakes and come back to this page and try the test again.
f.	Create a New JDBC Data Source: Select Targets	Under Clusters, select **cluster1**. Ensure that **All servers in the cluster** is selected. Click **Finish**.

e. In the Change Center, click **Activate Changes**.

Note: Notice the message at the top: "All changes have been activated. No restarts are necessary." This data source was created and targeted without having to restart the affected servers.

f. Verify the JNDI name is set on the servers in the cluster. Expand **Environments**, click **Servers**, click **server1**, and click **View JNDI Tree**. You should see **datasource1** as an entry in the list. Repeat for **server2**.

4. Tune the new generic data source.

Note: The scenario for the database is this:

- The database has intermittent peak usage and the database should be allowed to shrink every five minutes.
- The DBA has determined that each connection pool should start with one connection and grow to a maximum of five connections.
- Prepared and callable statements are used. The DBA has calculated that each connection pool should cache 15 statements.
- The DBA wants connections to be tested before they are given to applications.
- The DBA wants to test unused connections every four minutes.
- Even though connections are tested, performance impact is to be minimized by trusting idle connections for one minute.

a. Lock the configuration again by clicking **Lock & Edit** in the Change Center.

b. In the Domain Structure, expand **Services** and then select **Data Sources**. In the Data Sources table, select the new data source, **datasource1**.

c. Click the **Configuration** tab and then the **Connection Pool** subtab.

d. Use the guidelines in the following table to set some of the connection pool attributes:

Step	Field and Value	Reason
a.	Initial Capacity: `1`	The initial value from the DBA **Note:** With intermittent peak usage, the initial capacity should be smaller than the maximum.
b.	Maximum Capacity: `5`	The maximum value from the DBA **Note:** Remember when setting this value that the actual maximum number of connections that can be retrieved from the database is this maximum times the number of target servers.
c.	Minimum Capacity: `1`	The same as the initial capacity **Note:** This number is used in pool shrinking calculations.
d.	Statement Cache Type: `LRU`	This algorithm keeps statements used frequently in the cache.
e.	Statement Cache Size: `15`	The cache size value from the DBA

e. Click **Save**.

f. Click **Advanced**. Use the guidelines in the following table to set the rest of the connection pool attributes:

Step	Field and Value	Reason
a.	Test Connections On Reserve: Selected	Test connections before they are distributed
b.	Test Frequency: `240`	Test unused connections every four minutes
c.	Seconds to Trust an Idle Pool Connection: `60`	Trust idle connections for one minute
d.	Shrink Frequency: `300`	Allow the database to shrink every five minutes

g. Click **Save**.

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

Note: Notice the message at the top: “All changes have been activated. No restarts are necessary.”

5. Log out of the administration console.
   a. Click the **Log Out** link.
   b. Close the web browser.

Practice Solution: Configuring a JDBC Data Source

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

> **Note:** If you started creating the data source, saved and activated the changes, this script will not work. To run the script in that case: access the administration console, lock the configuration, select the check box next to `datasource1`, click the **Delete** button, click **Yes**, and activate the changes.

Assumptions

You completed “Practice 5-1: Starting and Stopping Servers.”

All servers in the domain are currently running.

> **Note:** If you did not complete “Practice 5-1: Starting and Stopping Servers,” but completed “Practice 4-1: Creating a New Domain,” and “Practice 4-2: Copying a Domain to a New Machine,” and then start the servers in the domain, you can still run this solution.

Solution Tasks

1. Connect to host01.
2. Run the solution script.
   a. Open a Terminal window and navigate to the practice directory.
      ```
      $ cd /practices/part1/practice07-01
      ```
   b. Run the solution script.
      ```
      $ ./solution.sh
      ```
      Note: This script starts the WebLogic Scripting Tool (WLST) passing it the WLST script `create_data_source.py`, which creates the new generic data source for you. The WLST script prints out messages. Be patient and do not close the Terminal window until you see these messages:
      ```
      ...
      >>>Data source created successfully!


      Exiting WebLogic Scripting Tool.
      ```
3. Close the Terminal window.

Practices for Lesson 8: Monitoring a Domain

Chapter 8

Practices for Lesson 8: Overview

Practices Overview

In these practices, you use the administration console to access and configure WebLogic Server log files. You also use the monitoring capabilities of the admin console and the Monitoring Dashboard to monitor various aspects of WebLogic Server.

Practice 8-1: Working with WebLogic Server Logs

Overview

In this practice, you access a WebLogic Server log file by using the administration console. You search the server log file by using an editor. You create a log filter and apply it to one of the logging outputs.

Assumptions

You completed “Practice 7-1: Configuring a JDBC Data Source.”

All servers in the domain are currently running.

Tasks

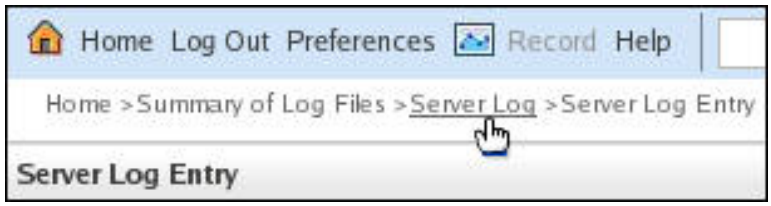
1. Connect to host01.
2. Open the WebLogic Server administration console.
3. In the admin console, locate the server log file for `server1`.
   a. In the Domain Structure, expand **Diagnostics** (click the “**+**” sign), and then select **Log Files**.
   b. In the Log Files table, select the **ServerLog** option for server1. Then click the **View** button.

   Note: If you do not see this log on the first page of the table, click the **Next** link.

   c. The Server Log Entries table displays.

   Note: There may be no rows in the table. The table settings determine what to display.

   d. Click the **Customize this table** link.
   e. Use the drop-down list to update **Time Interval**. Choose **`Last 2 day(s)`** or some other interval that will display quite a few messages. Also, use the drop-down list to change the **Number of rows displayed per page** to `25` (or a number of your choice).
   f. Click the **Apply** button.
   g. Select the option button next to a message and click the **View** button to see the details of one of the messages.
   h. Use the breadcrumbs at the top of the page to go back to the log by clicking **Server Log**.



   i. Scroll up and down and look at the log messages. Click the **Next** and **Previous** links.

   Note: You may notice that the administration console does not provide a log search capability. Because it does not, next you will open the log file in an editor.

4. In a Terminal window, navigate to the location of the log file and edit it. Use the editor to search the log file.
   a. Open a Terminal window and navigate to the `logs` directory under the `server1` directory. List the directory contents.

      `$ cd /u01/domains/part1/wlsadmin/servers/server1/logs`

```
$ ls
access.log diagnostic_images server1.log
```

Note: There may be more files and directories in your `logs` directory than listed here.

b. Open the `server1.log` file in an editor.

```
$ gedit server1.log
```

c. Use the search capabilities of the gedit editor to find messages placed in the log by the JDBC subsystem. Press **Ctrl + F** to use the editor's search capabilities. Enter **<jdbc>** in the search field and click the **Find** button.

d. Read the first found message. Continue to click the **Find** button and look at other from the JDBC subsystem. Can you find a message that gives which JDBC driver is being used for the data source?

Hint: You can do find again, this time find: **jdbc driver**.

e. Close the Find window. Close the editor. Close the Terminal window.

5. Use the admin console to create a log filter.

Note: Here is the scenario: Let us say that you are having issues with server2 and clustering. You want to temporarily have server2 only publish messages from the Cluster subsystem into its log, so it is easier to look through the log and find the problem. You create and apply a log filter that accomplishes this.

a. In the Change Center, click **Lock & Edit**.

b. In the Domain Structure, select the domain, **wlsadmin**.

c. Click the **Configuration** tab and then the **Log Filters** subtab.

d. Above the Log Filters table, click the **New** button.

e. For the name of the filter, enter **clusterfilter**. Click **OK**.

f. In the Log Filters table, click the name of the new filter, **clusterfilter**.

g. On the **Configuration** tab of the filter, click the **Add Expressions** button.

h. Enter or select these values for the following fields, and then click **OK**:

- Message Attribute: **SUBSYSTEM**
- Operator: **LIKE**
- Value: **Cluster**

i. Click **OK**.

j. In the Change Center, click **Activate Changes**.

6. Apply the new log filter to the server log of server2.

a. In the Change Center, click **Lock & Edit**.

b. In the Domain Structure, expand **Environment** and then select **Servers**.

c. In the servers table, select **server2**.

d. Click the **Logging** tab. Ensure that the **General** subtab is selected.

e. Scroll down to find the **Advanced** link. Right above this link is the **Rotate log file on startup** check box. Select this check box to enable this feature.

Note

- In production systems, this is disabled by default. You enable it here so it is easier to see the log filter at work.
- Notice that this attribute change does not take effect until the server is restarted.

f. Click **Save**.

g. Click **Advanced**.

h. Scroll down to the “Message destination(s)” area. Under “Log file” use the **Filter** drop-down list to select `clusterfilter`.

i. Also under “Log file,” change the value of **Log File Buffer** to `2`.

Note

- This will ensure that the log file is written to more often. This is probably too small for the buffer in a real production system.
- Notice that this attribute change also does not take effect until the server is restarted.

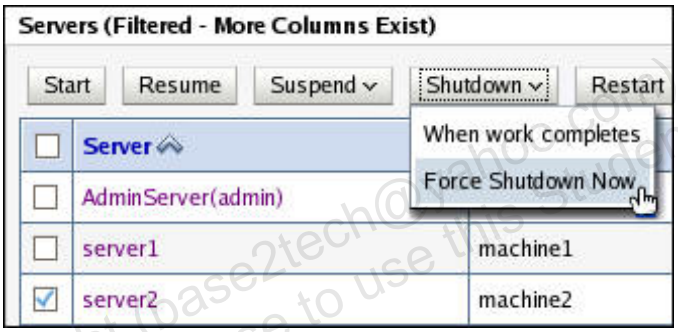
j. Click the **Save** button.

k. In the Change Center, click **Activate Changes**.

7. Check that the filter is working. First, stop and restart server2, then go look at the server2 log file.

a. In the admin console, in the Domain Structure, expand **Environment**, then select **Servers**. Click the **Control** tab.

b. Select the check box next to **server2**. Then click the **Shutdown** button and select **Force Shutdown Now**.



c. When asked to confirm the shutdown, click **Yes**.

d. Connect to host02.

e. Find the Terminal window on host02 that was running server2. You should see that server2 has shut down. Start it again by pressing the up arrow key to retrieve the command you used to previously start it, and pressing **Enter**.

Note

- Remember to enter the username and password if you do not have a boot identity file for server2.
- If you closed the previous Terminal Window, open a new window and start server2 there:

```
$ cd /u01/domains/part1/wlsadmin/bin
$ ./startManagedWebLogic.sh server2 host01.example.com:7001
```

f. When you see the message `<The server started in RUNNING mode.>`, minimize the Terminal window.

g. Open a new Terminal window and navigate to the location of server2’s log file. List the server2 log files.

```
$ cd /u01/domains/part1/wlsadmin/servers/server2/logs
$ ls server2.*
```

```
server2.log  server2.log00001
```

Note: You may have more than one older server2 log file. The newest server log file has no digits in its extension.

h. Edit the newest log file.

```
$ gedit server2.log
```

i. Look at the log file in the editor. The subsystem is after the timestamp and severity level. When server2 first comes up, the log filter has not yet been engaged. But soon it is, and from that point on, you notice that all messages are from the Cluster subsystem.

```
...
####<Feb 22, 2013 11:57:06 AM UTC> <Info> <WebLogicServer>...
####<Feb 22, 2013 11:57:27 AM UTC> <Notice> <Cluster>...
####<Feb 22, 2013 11:57:36 AM UTC> <Info> <Cluster>...
...
####<Feb 22, 2013 11:57:37 AM UTC> <Info> <Cluster>...
```

Note: The timestamps in your log file will be different. The severity levels may be different.

j. Exit the editor.

k. Close the Terminal window.

8. Remove the filter from server2.

Note: A filter like this would only be used temporarily. After you remedied whatever the cluster issue was, you would remove the filter. Without the usual messages going to the server log, it would be difficult to debug new problems as they arise.

a. Return to the web browser with the admin console.

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

c. In the Domain Structure, expand **Environment** and then select **Servers**.

d. In the servers table, select **server2**.

e. Click the **Logging** tab. Ensure that the **General** subtab is selected.

f. Click **Advanced**.

g. Scroll down to the “Message destination(s)” area. Under “Log file” use the **Filter** drop-down list to select **None**.

h. Click the **Save** button.

i. In the Change Center, click **Activate Changes**.

j. If you are not proceeding to the next practice, log out of the admin console, and close the web browser.

Practice Solution: Working with WebLogic Server Logs

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

Assumptions

You completed “Practice 7-1: Configuring a JDBC Data Source.”

All servers in the domain are currently running.

Solution Tasks

1. Connect to host01.
2. Run the solution script.
   a. In a Terminal window, navigate to the practice directory.
      ```
      $ cd /practices/part1/practice08-01
      ```
   b. Run the solution script.
      ```
      $ ./solution.sh
      ```
      Note: This script starts the WebLogic Scripting Tool (WLST) passing it the WLST script `create_logfilter.py`, which creates the log filter. It does not apply the filter, because this filter is removed at the end of the practice. Also note that the WLST script does not change any server2 log settings that would require a server restart, nor are those settings necessary.
3. Close the Terminal window.

Practice 8-2: Monitoring WebLogic Server

Overview

In this practice, you use the monitoring capabilities of the admin console and the Monitoring Dashboard to monitor various aspects of WebLogic Server.

Assumptions

You completed “Practice 7-1: Configuring a JDBC Data Source.” (“Practice 8-1: Working with WebLogic Server Logs” is not required for this or any other practice.)

All servers in the domain are currently running.

Tasks

1. Connect to host01.
2. Access the WebLogic Server administration console.
3. Use the admin console to monitor server health.
   a. In the Domain Structure, expand **Environment** (click the “**+**” sign), and then select **Servers**.
   b. Before doing anything else, notice the Servers table. By default it displays each server’s state and health. Other columns of information can be added to the table by using the **Customize this table** link.
   c. In the Servers table, select **server2**.
   d. Click the **Monitoring** tab. Ensure that the **General** subtab is selected. Notice the information available, such as when the server started, the version of WebLogic Server, which JVM and version WebLogic Server is running under, the operating system and version, and so on.
   e. Click the **Health** subtab under the **Monitoring** tab. Some of the WebLogic Server subsystems are listed along with their health.
   f. Stay on the **Monitoring** tab of the server for the next task.
4. Modify the information displayed by a monitoring table. You will use this table later.
   a. Click the **JDBC** subtab under the server’s **Monitoring** tab.
   b. Click the **Customize this table** link. In the Chosen column, select **Active Connections Average Count** and **Active Connections High Count**. Click the left arrow ( ) to move these attributes to the Available column. In the Available column, select **Current Capacity** and **Num Available**, and click the right arrow ( ) to move them to the Chosen column. Click the **Apply** button.
5. Before monitoring a data source, check on some of its settings.
   a. In the admin console, select the data source (in the Domain Structure, expand **Services**, select **Data Sources**, and select **datasource1**).
   b. Select **Configuration** > **Connection Pool**. Make note of the Maximum Capacity.

   Note: It should be set to `5`. But if it is something different, that is OK. Just remember it.

   c. Still on this data source, click the **Monitoring** > **Statistics** tabs. Ensure that the data source is enabled and running on both servers.

   Note: If not, then perhaps one of the servers has not been started. Start it now.

   d. Minimize the web browser. You will be using it later.
6. Connect to host02. From there, run the script that sets up the database with a table and loads it with some data. Then, run the script that calls a Java client. This client uses the

data source to retrieve a connection to the database, accesses the database, and loops to do it again.

Note: Notice that this is host02, not host01.

a. Open a Terminal window and navigate to the practice directory. Set a couple of environment variables and run the script that sets up the database.

Note: The `ORACLE_HOME` variable needs to point to where the database is installed. `ORACLE_SID` (Oracle System ID) is the unique name of this particular database.

```
$ cd /practices/part1/practice08-02
$ export ORACLE_HOME=/u01/app/db11g/product/11.2.0/dbhome_1
$ export ORACLE_SID=orcl
$ ./setup.sh
...
1 row created

Commit complete.
...
$
```

Note: The SQL script drops the table (and sequence) it creates, so you can run it multiple times. Therefore, the first time the script is run, it produces messages that no such table or sequence exist. Ignore those messages if you see them.

b. Now, run the script that calls the Java client. When you run the script, you pass it the number of times you want it to access the database. The first time it accesses the database, it prints out some data it retrieves. It then loops, accessing the database, in total, the number of times you entered. The client then pauses. You let the client wait, still running, and go see what is happening with the data source.

Important: The number you enter when you run the script should be TWO TIMES the Maximum Capacity you made note of earlier. Why double it? Because the data source is targeted to the cluster, and there are two servers in the cluster, there are two instances of the data source. Therefore, the number of available connections is twice the data source connection pool **Maximum Capacity**.

Note: This Java client is written poorly on purpose. Each time the client code wants to access the database, it asks for a connection from the data source. When the code is finished with the connection, the code does not close that connection. Therefore, that connection is not returned to the connection pool.

```
$ ./runclient.sh 10
...
Your environment has been set.
>>>Obtained initial context
>>>Data source named datasource1 retrieved
>>>Querying data source...
Contacts from database:
1 Homer Simpson 742...
...
This client is still running...
When you are ready to stop it, press Enter.
```

Note: Do NOT press Enter yet!

c. Go back to the admin console on host01.

d. Navigate to the **Monitoring** > **JDBC** tab of **server2** (in the Domain Structure expand **Environment** and select **Servers**. Select **server2**. Click the **Monitoring** > **JDBC** tabs). Refresh the web browser (you can press **F5**).

e. Notice that **Active Connections Current Count** is equal to the **Maximum Capacity** you noted earlier. The **Current Capacity** value is also the **Maximum Capacity**. And **Num Available** is `0`. The bad code in the client has used up each one of the database connections.

f. In the Terminal window on host02 where the Java client is running, press the **Enter** key. The client code completes.

7. View the kinds of messages that the JDBC subsystem places in a server log.

a. Open a Terminal window and navigate to the `logs` directory under the `server2` directory. Edit the server log.

```
$ cd /u01/domains/part1/wlsadmin/servers/server2/logs
$ gedit server2.log
```

b. In the editor, scroll all the way to the bottom of the log file. Then use the search capabilities of the gedit editor to find messages placed in the log by the JDBC subsystem. Press **Ctrl + F** to use the editor's search capabilities. Because you are at the end of the file, select **Search backwards**. Enter **<jdbc>** in the search field and click the **Find** button.

c. You should see some messages that connections have been closed. This happened when the client stopped running earlier.

```
####...<JDBC>...<Connection for pool "datasource1" has been
closed.>
```

d. Close the Find window. Close the editor.

8. Use the admin console to enable JDBC debugging. This increases the number of messages about JDBC sent to the server log.

a. Go back to the admin console on host01.

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

c. Select **server2**. (In the Domain Structure, click **Environment** > **Servers** and then select **server2** in the table).

d. Click the **Debug** tab.

e. In the table, expand **weblogic**. Then, select the box next to **jdbc**. Click the **Enable** button. Notice that jdbc is now "Enabled," as well as all of its children.

f. In the Change Center, click **Activate Changes**. Notice that no restarts are necessary.

9. Run the Java database client again on host02.

a. In a Terminal window on host02, once again navigate to the practice directory and run the script to call the Java database client. This time you can press Enter right away to stop the client.

```
$ cd /practices/part1/practice08-02
$ ./runclient.sh 10
...
>>>Obtained initial context
>>>Data source named datasource1 retrieved
```

```
>>>Querying data source...
Contacts from database:
1 Homer Simpson 742...
...
This client is still running...
When you are ready to stop it, press Enter.

$
```

10. View the server2 log to see the debug messages.
    a. In a Terminal window, navigate to the `logs` directory under the `server2` directory and edit the server log.
       ```
       $ cd /u01/domains/part1/wlsadmin/servers/server2/logs
       $ gedit server2.log
       ```
    b. In the editor, scroll all the way to the bottom of the log file. Then use the search capabilities of the gedit editor to find debug messages placed in the log by the JDBC subsystem. Press **Ctrl + F** to use the editor's search capabilities. Because you are at the end of the file, select **Search backwards**. Type **<debug>** in the search field and click the **Find** button.

    Note
    - Notice that there are many messages of the "debug" severity level. They are from various JDBC-related subsystems.
    - The "info" level messages you found before from the JDBC subsystem, about connections closing, are still there. In addition, there are quite a few debug messages around each one of those.
    - As you are editing this file, the editor may interrupt you with a message that the file has "changed on disk" and ask if you want to reload the file. You can reload it if you want to. Because many debug messages are being sent to the log, it is possible that the file changes as you are viewing it.

    c. See whether you can find debug messages that show connection testing. Scroll to the bottom of the file, use the find, and search backwards for the word **dual**.

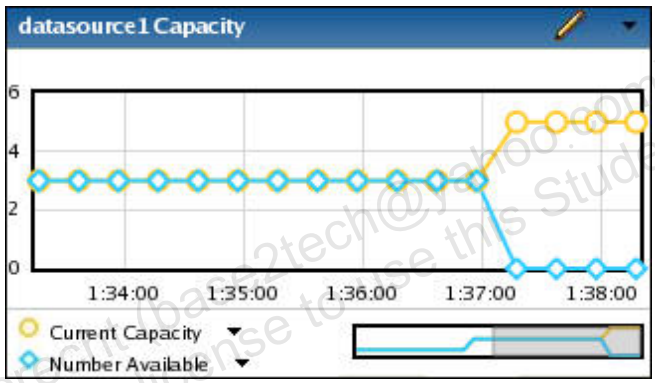
    Note: The default Test Table Name setting for a data source connection pool using an Oracle database is `SQL SELECT 1 FROM DUAL`.

    d. Close the Find window. Close the editor.
    e. Close the Terminal window.
11. Turn off debugging.
    a. Return to the admin console.
    b. In the Change Center, click **Lock & Edit**.
    c. Select **server2**. (In the Domain Structure, click **Environment** > **Servers** and then select **server2** in the table).
    d. Click the **Debug** tab.
    e. In the table, expand **weblogic**. Then, select the box next to **jdbc**. Click the **Disable** button.
    f. In the Change Center, click **Activate Changes**. Notice that no restarts are necessary.
12. Monitor a data source by using the Monitoring Dashboard. You will run the database client one more time.

a. Go to the home page of the admin console. (You can use the Home link.)
b. Under “Charts and Graphs,” click the **Monitoring Dashboard** link. The dashboard opens in a new window (or tab).
c. In the Monitoring Dashboard View List, expand **server2**. Select the view **JDBC Data Sources on server2**.
d. Above the view list, press the Start button ( ).
e. In a Terminal window on host02, navigate to the practice directory and run the script to call the Java database client. When the client is waiting, do *not* press Enter, let it continue to run.

```
$ cd /practices/part1/practice08-02
$ ./runclient.sh 10
...
This client is still running...
When you are ready to stop it, press Enter.
```

f. Return to the Monitoring Dashboard. You should notice changes in the “datasource1 Capacity” chart (top left). Your chart may not look exactly like this one, but the “Number Available” should go to 0.



g. In the “datasource1 Connections” chart, you should also notice that the “Active Connections Current Count” goes up.
h. Above the View List, click the Stop button ( ).
i. Close the Monitoring Dashboard window (or tab). When asked whether you are sure, click the **Leave Page** button.
j. Log out of the admin console.
k. Close the web browser.
l. In the Terminal window on host02 that is running the Java client, press Enter. Once the client stops, close the Terminal window.

Practice Solution: Monitoring WebLogic Server

There is no solution for this practice.

Practices for Lesson 9: Node Manager

Chapter 9

Practices for Lesson 9: Overview

Practices Overview

In this practice, you configure Node Manager and start it on both machines. You then start managed servers through it by using the administration console.

Practice 9-1: Configuring and Using Node Manager

Overview

In this practice, you configure the Java-based Node Manager on one of the machines. You start that Node Manager. Once it is running, you use the administration console to start the managed server that runs on that machine. Then you perform the same tasks on the other machine.

Assumptions

You completed “Practice 7-1: Configuring a JDBC Data Source.”

The administration server is currently running.

Tasks

1. Connect to host01.
2. Access the WebLogic Server administration console on host01.
3. Stop the managed servers that are running.
   a. In the Domain Structure, expand **Environment** and then select **Servers**.
   b. On the right, click the **Control** tab.
   c. Select the check box next to any managed servers that are running. Click the **Shutdown** button and select **Force Shutdown Now**. When asked to confirm, click **Yes**.
   d. Wait a moment and refresh the screen. Ensure that the State column in the table confirms that all managed servers are shut down.
4. Update the configuration of the machines.
   a. In the Change Center, click **Lock & Edit**.
   b. In the Domain Structure, under **Environment**, select **Machines**.
   c. Select **machine1**.
   d. Click the **Configuration** tab and the **Node Manager** subtab.
   e. Use the drop-down list to change the **Type** to `Plain`.
   f. Ensure that **Listen Address** is `host01.example.com` and the **Listen Port** is `5556`.
   g. Click **Save**.
   h. Click the **Configuration** > **Servers** tab. Ensure that server1 is assigned to machine1.
   i. Return to the Machines table and select **machine2**.
   j. Once again, click the **Configuration** tab and the **Node Manager** subtab. Use the drop-down list to change the **Type** to `Plain`. Ensure the **Listen Address** is `host02.example.com` and the **Listen Port** is `5556`.
   k. Click **Save** again.
   l. Click the **Configuration** > **Servers** tab. Ensure that server2 is assigned to machine2.
   m. In Change Center, click **Activate Changes**.

   Note: You will note that one restart is required. Because the admin server is assigned to machine1, it must be restarted for the machine’s configuration changes to take effect.
5. Stop and restart the administration server running on host01.
   a. Find the Terminal window in which server1 was running and close it.

Note: Even if you do not have the Terminal window titles set, you can tell in which window server1 was running, because you stopped server1. You can see “shut down” messages, and the window shows the prompt again. Close the server1 window.

b. Find the Terminal window in which the admin server is running.

c. Press **Ctrl + C** in the admin server window.

Note: This is not as bad as it seems. It has the same effect as Force Shutdown Now.

d. Use the up arrow to bring back the command issued to start the admin server, and start it again.

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

6. View the files needed by Node Manager on host01. Update one of them.

a. Open a new Terminal window on host01. Navigate to the domain’s Node Manager directory. List the files there.

```
$ cd /u01/domains/part1/wlsadmin/nodemanager
$ ls
nodemanager.domains  nodemanager.properties
```

b. List the contents of the `nodemanager.domains` file. Notice that the only entry is the name of your domain and its location. This Node Manager is for the `wlsadmin` domain.

```
$ cat nodemanager.domains
...
wlsadmin=/u01/domains/part1/wlsadmin
```

Note: The comments at the top of the file are not shown here.

c. Edit the `nodemanager.properties` file.

```
$ gedit nodemanager.properties
```

d. Look at the file following properties and their values:

- `ListenAddress=host01.example.com`
- `ListenPort=5556`
- `StartScriptEnabled=true`
- `StartScriptName=startWebLogic.sh`

Note

- This Node Manager runs at the listen address and listen port listed.
- Using a start script to start servers is recommended, and it is the default.
- Notice that the start script listed is `startWebLogic.sh`. Even though the script `startManagedWeblogic.sh` is the one you used to start managed servers, it is there as a convenience. It calls `startWebLogic.sh`, which does the real work of starting a server.

e. Find the property `SecureListener` and change its value to **`false`**.

```
SecureListener=false
```

f. Save the file and close the editor.

g. Leave the Terminal window open.

7. Start the Node Manager on host01.

a. Run the Node Manager start script in the `bin` directory of the domain:

```
$ cd /u01/domains/part1/wlsadmin/bin
```

```
$ ./startNodeManager.sh
...
... <INFO> Plain socket listener started on port 5556, host
host01.example.com>
```

b. Do not close this window. This is where the Node Manager is running. Minimize the window, if you want. You may want to change the title of the Terminal window to **Node Manager**.

8. Start server1 through Node Manager by using the administration console.
   a. In the Domain Structure, expand **Environment**, then select **Servers**. Click the **Control** tab. Select the check box next to server1 and click the **Start** button. When asked whether you are sure, click **Yes**.
   b. Wait a moment, and refresh the browser. Wait until server1's State is RUNNING.
9. View the files needed by Node Manager on host02. Update one of them. Then start Node Manager on host02.
   a. Connect to host02.
   b. Find the Terminal window in which server2 was running and close it.
   c. Open a new Terminal window. Navigate to the domain's Node Manager directory. Edit the `nodemanager.properties` file.

```
$ cd /u01/domains/part1/wlsadmin/nodemanager
$ gedit nodemanager.properties
```

   d. Find the property `SecureListener` and change its value to **false**.

```
SecureListener=false
```

   e. Save the file and close the editor.
   f. Run the Node Manager start script in the `bin` directory of the domain:

```
$ cd /u01/domains/part1/wlsadmin/bin
$ ./startNodeManager.sh
...
...<INFO> <Plain socket listener started on port 5556, host
host02.example.com>
```

   g. Do not close this window. This is where the Node Manager is running. Minimize the window, if you want. You may want to change the title of the Terminal window to **Node Manager**.
10. Start server2 through Node Manager by using the administration console.
    a. Open the web browser running the administration console.
    b. In the Domain Structure, expand **Environment**, then select **Servers**. Click the **Control** tab. Select the check box next to server2 and click the **Start** button. When asked whether you are sure, click **Yes**.
    c. Wait a moment, and refresh the browser. Wait until server2's State is RUNNING.
    d. Log out of the admin console and close the web browser.

Practice Solution: Configuring and Using Node Manager

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

Assumptions

You completed “Practice 7-1: Configuring a JDBC Data Source.”

The administration server is currently running.

Solution Tasks

1. Connect to host01.
2. Run the solution script for host01.
   a. Open a Terminal window and navigate to the practice directory.
      ```
      $ cd /practices/part1/practice09-01
      ```
   b. Run the host01 solution script.
      ```
      $ ./solution-host01.sh
      ```
      Note: This script copies the correct `nodemanager.properties` file to the domain’s `nodemanager` directory. It then starts the WebLogic Scripting Tool (WLST) passing it the WLST script `update_machines.py`, which updates the two machines for you.
   c. Close the Terminal window.
3. Stop the admin server and the managed server on host01.
   Note: You want to stop the admin server so that Node Manager is running when it comes up again. You want to stop the managed server because from this point forward you will start it through Node Manager by using the admin console.
   a. Find the Terminal window in which the admin server is running. Press **Ctrl + C**.
   b. Minimize this Terminal window. You will use it later.
   c. Find the Terminal window in which server1 is running. Press **Ctrl + C**.
   d. Close this Terminal window.
4. Start the Node Manager on host01.
   a. Open a new Terminal window.
   b. Navigate to the `bin` directory of the domain and run the Node Manager start script:
      ```
      $ cd /u01/domains/part1/wlsadmin/bin
      $ ./startNodeManager.sh
      ...
      ... <INFO> Plain socket listener started on port 5556, host
      host01.example.com>
      ```
   c. Do not close this window. This is where the Node Manager is running.
   d. If you want to, set the title of the window to “Node Manager” by using the window’s **Terminal** > **Set Title** option.
   e. Minimize this Terminal window.
5. Restart the administration server.
   a. Find the Terminal window in which the admin server was running.
   b. Press the up arrow to bring up the last command (`startWebLogic.sh`). Press Enter to run it again.
      ```
      $ ./startWebLogic.sh
      ```

c. If you do not have a boot identity file, when you are prompted, enter the username and password (**weblogic** and **Welcome1**). Wait for the server to indicate it is in RUNNING mode.

d. Minimize this Terminal window.

6. Connect to host02.
7. Stop the managed server on host02.

   Note: You want to stop the managed server because from this point forward you will start it through Node Manager by using the admin console.

   a. Find the Terminal window in which server2 is running. Press **Ctrl + C**.

   b. Close this Terminal window.
8. Run the solution script for host02.

   a. Open a new Terminal window and navigate to the practice directory.

   ```
   $ cd /practices/part1/practice09-01
   ```

   b. Run the host02 solution script.

   ```
   $ ./solution-host02.sh
   ```

   Note: This script copies the correct `nodemanager.properties` file to the domain's `nodemanager` directory.

   c. Close the Terminal window.
9. Start the Node Manager on host02.

   a. Open a new Terminal window and navigate to the `bin` directory of the domain. Run the Node Manager start script:

   ```
   $ cd /u01/domains/part1/wlsadmin/bin
   $ ./startNodeManager.sh
   ...
   ... <INFO> Plain socket listener started on port 5556, host
   host02.example.com>
   ```

   b. Do not close this window. This is where the Node Manager is running. If you want to, set the title of the window to "Node Manager" by using the window's **Terminal** > **Set Title** option.

   c. Minimize this Terminal window.
10. Restart the managed servers through Node Manager by using the admin console.

    a. In the Domain Structure, expand **Environment** and then select **Servers**. Click the **Control** tab. Select the check boxes next to **server1** and **server2**. Click the **Start** button. When asked whether you are sure, click **Yes**.

    b. Wait a little while, and refresh the browser. Ensure that the state of both managed servers is RUNNING.

    c. Close the web browser.

Practices for Lesson 10: Deploying Applications

Chapter 10

Practices for Lesson 10: Overview

Practices Overview

In these practices, you deploy, redeploy, and undeploy an application. You also load test an application by using The Grinder.

Practice 10-1: Deploying an Application

Overview

In this practice, you deploy, redeploy, and undeploy an application by using the administration console.

Assumptions

You completed “Practice 9-1: Configuring and Using Node Manager.”

All servers are currently running.

Tasks

1. Create a new directory from which to deploy applications. Copy the application from the practice directory to this new directory.
   a. Connect to host01.
   b. Open a new Terminal window and navigate to the domain directory. Create a new subdirectory called `apps` from which you will deploy applications.
      ```
      $ cd /u01/domains/part1/wlsadmin
      $ mkdir apps
      ```
   c. Copy the application from the current practice directory to this directory.
      ```
      $ cd apps
      $ cp /practices/part1/practice10-01/benefits.war .
      ```
   d. Leave the Terminal window open.
2. Open the WebLogic Server administration console.
3. Deploy the application.
   a. In the Change Center, click **Lock & Edit**.
   b. In the Domain Structure, select **Deployments**.
   c. Above the Deployments table, click the **Install** button.
   d. Use the links next to and below Current Location to navigate to:
      `host01.example.com/u01/domains/part1/wlsadmin/apps`

      Note: This fills in the **Path** field as you click links.
   e. When the path is correct, select the option button next to **benefits.war**.

   f. Click the **Next** button.
   g. Select **Install this deployment as an application** and click **Next**.
   h. Select the check box next to **cluster1**. Ensure that the **All servers in the cluster** option button is selected. Click **Next**.
   i. Leave all the Optional Settings at their default values. Click **Next**.
   j. Select **No, I will review the configuration later**. Click **Finish**.

k. In the Change Center, click **Activate Changes**.

Note: Notice the message that the changes have been activated and no restarts are necessary.

4. Start the application.
   a. In the Deployments table, select the check box next to the benefits application.
   b. Click the **Start** button and select **Servicing all requests**.
   c. When asked to confirm, click **Yes**.

   Note: Notice that the State of the benefits application is now "Active."
5. Test the application.
   a. In the Deployments table, click the link that is the name of the newly deployed application, **benefits**.
   b. Click the **Testing** tab.
   c. Click any of the test links.
   d. The benefits application opens in a new window or tab.
   e. Try the application by selecting things and clicking buttons or links.

   f. When you are finished trying the application, close the Benefits window or tab.
6. Obtain an updated version of the application.
   a. Open a Terminal window on host01 and navigate to the current practice directory. List the contents of the directory. Then list the contents of the `update` subdirectory.

```
$ cd /practices/part1/practice10-01
$ ls
...  benefits.war  ...  update
$ ls update
benefits.war
```

   Note: The `update` directory has an updated version of the benefits application.
   b. Navigate to the `apps` directory. Rename the current benefits archive. Copy the updated version to its location.

```
$ cd /u01/domains/part1/wlsadmin/apps
$ mv benefits.war benefits.old
$ cp /practices/part1/practice10-01/update/benefits.war .
```

   c. Close the Terminal window.
7. Redeploy the application.
   a. Return to the administration console.
   b. In the Change Center, click **Lock & Edit**.

c. In the Domain Structure, select **Deployments**.
d. Select the check box next to the **benefits** application.
e. Click the **Update** button.
f. Leave the “Source path” as it is. There is no deployment plan, so the empty “Deployment plan path” is correct. Click **Next**.
g. On the next page, click **Finish**.
h. In the Change Center, click **Activate Changes**.

Note: Notice the message that the changes have been activated and no restarts are necessary. Also notice that the State of the benefits application is “Active,” even though you did not explicitly start it.

i. Follow the earlier steps to test the application again. When you are finished, close the Benefits window or tab.

8. Stop the application, so you can undeploy it.
   a. Return to the administration console.
   b. In the Domain Structure, select **Deployments**.
   c. Select the check box next to the **benefits** application.
   d. Click the **Stop** button and select **Force Stop Now**.
   e. When asked to confirm, click **Yes**.

   Note: Notice the State of the benefits application is no longer “Active.”
9. Undeploy the application.
   a. In the Change Center, click **Lock & Edit**.

   Note: If it is already locked, skip this step.

   b. In the Domain Structure, select **Deployments**.
   c. Select the check box next to the **benefits** application.
   d. Click the **Delete** button.
   e. When asked to confirm, click **Yes**.
   f. In the Change Center, click **Activate Changes**.

   Note: Notice the message that the changes have been activated and no restarts are necessary. Also notice that the benefits application is no longer in the Deployments table.

Practice Solution: Deploying an Application

Perform the following tasks if you did not complete this practice and want to use the finished solution. Note that this solution leaves the application deployed, so you can use the application if you want to. To have your domain in the same state as if you did this practice, do nothing. (The final task of the practice is to undeploy the new application.)

Assumptions

You completed "Practice 9-1: Configuring and Using Node Manager."

All servers are currently running.

Solution Tasks

1. Connect to host01.
2. Run the solution script.
   a. Open a Terminal window and navigate to the practice directory.
      `$ cd /practices/part1/practice10-01`
   b. Run the solution script.
      `$ ./solution.sh`

      Note
      - This script creates the `apps` directory under the domain directory. It copies the original `benefits.war` file there, calling it `benefits.old`. It then copies the updated `benefits.war` into the `apps` directory from the practice's `update` subdirectory. The script then starts the WebLogic Scripting Tool (WLST) passing it the WLST script `deploy_app.py`, which deploys the benefits application, targets it to `cluster1`, and starts it.
      - If you get a warning about closing in a different thread, ignore it (`WLContext.close() was called in a different thread...`). It is a known issue with WLST deployment and does no harm.

   c. Do not close the Terminal window.
   d. If you want to access the newly deployed application, enter the following URL in the web browser:
      `http://host01.example.com:7011/benefits`
   e. Close the web browser when you are finished.

3. If you want to undeploy the application after you have looked at it, as the practice does, run the `undeploy.sh` script.
   a. In the same terminal window, run the undeploy script.
      `$ ./undeploy.sh`

      Note
      - The script starts the WebLogic Scripting Tool (WLST) passing it the WLST script `undeploy_app.py`, which undeploys the benefits application.

   b. Close the Terminal window.

Practice 10-2: Load Testing an Application

Overview

In this practice, you use The Grinder to load test an application running on WebLogic Server. You then view the results in The Grinder console.

Assumptions

You completed “Practice 9-1: Configuring and Using Node Manager.”

“Practice 10-1: Deploying an Application” is not necessary for this practice.

All servers are currently running.

Tasks

1. Connect to host01. Run a script to deploy the application from which The Grinder script was created.
   a. Open a Terminal window and navigate to the practice directory. Then, run the `deploy.sh` script.

```
$ cd /practices/part1/practice10-02
$ ./deploy.sh
...
>>>Deploying application contacts. Please wait.
...
>>>Application contacts deployed.
...
$
```

2. Connect to host02. Run the database setup script. It initializes (or reinitializes) the database to create a table and put some data in it that the application requires.

   Note: Notice that this is host02, not host01.

   a. Open a Terminal window and navigate to the practice directory. Set some environment variables. Then run the `setup.sh` script.

   Note: The `ORACLE_HOME` variable points to where the database is installed. `ORACLE_SID` (Oracle System ID) is the unique name of this particular database.

```
$ cd /practices/part1/practice10-02
$ export ORACLE_HOME=/u01/app/db11g/product/11.2.0/dbhome_1
$ export ORACLE_SID=orcl
$ ./setup.sh

SQL*Plus: Release 11.2...
...
1 row created.

Commit complete.
...
```

Note: The SQL script drops the table (and sequence) it creates, so you can run it multiple times. Therefore, the first time the script is run, it produces messages that no such table or sequence exist. Ignore those messages if you see them.

3. Access the newly deployed application from the web browser. You will use the application in the same way that it was used to create the `grinder.py` script. This script was recorded for you by using The Grinder TCPProxy utility and using the contacts application in a web browser.

   Note: To record the script, the web browser first had to be configured. It was set to communicate with the TCPProxy utility by setting a manual proxy configuration. The HTTP Proxy field was set to `localhost`, and the Port field set to `8001`. (That is the port on which the TCPProxy utility runs.)

   a. In the web browser, enter the URL for the application:
      `http://host01.example.com:7011/contacts`

   Note: This is the first request in The Grinder script (which will display in The Grinder console as "Page 1").

   b. On the home page, click the **browse all contacts** link.

   Note: This is the second request in the script (which will display in The Grinder console as "Page 2").

   c. Click the **[edit]** link in the first row.

   Note: This is the third request in the script (which will display in The Grinder console as "Page 3").

   d. View the data on the edit screen.

   Note: This is the fourth request in the script (which will display in The Grinder console as "Page 4").

4. Install The Grinder on host02.

   a. Open a Terminal window on host02 and navigate to `/install/grinder`. Then, list the contents of the directory.

```
$ cd /install/grinder
$ ls
grinder-3.11-binary.zip
```

   b. Unzip the zip file to `/home/oracle`.

```
$ unzip grinder-3.11-binary.zip -d /home/oracle
Archive:  grinder-3.11-binary.zip
  creating: /home/oracle/grinder-3.11/
 inflating: /home/oracle/grinder-3.11/CHANGES
...
```

   c. Leave the Terminal window open for later.

5. Install The Grinder on host01.

   a. Open a Terminal window on host01 and navigate to `/install/grinder`.

```
$ cd /install/grinder
```

   b. Unzip the zip file to `/home/oracle`.

```
$ unzip grinder-3.11-binary.zip -d /home/oracle
Archive:  grinder-3.11-binary.zip
  creating: /home/oracle/grinder-3.11/
```

```
 inflating: /home/oracle/grinder-3.11/CHANGES
...
```

6. Examine the `grinder.properties` file and start The Grinder console.
   a. In the Terminal window on host02, navigate to the practice directory.
      `$ cd /practices/part1/practice10-02`
   b. View the `grinder.properties` file in the editor.
      `$ gedit grinder.properties`
   c. Scroll down past the comments and notice the following attributes and their values:
      - `grinder.script = grinder.py`
      - `grinder.processes = 2`
      - `grinder.threads = 200`
      - `grinder.runs = 0`

      Note: What this means is that the agent starts two worker processes with 200 threads each. It runs the script called `grinder.py`. It continues running until the console tells it to stop.
   d. Exit the editor without making any changes.
   e. View the `setgrinderenv.sh` script.

```
$ cat setgrinderenv.sh
GRINDERPATH=/home/oracle/grinder-3.11
GRINDERPROPS=/practices/part1/practice10-02/grinder.properties
CLASSPATH=$GRINDERPATH/lib/grinder.jar:$CLASSPATH
JAVA_HOME=/u01/app/jdk
PATH=$JAVA_HOME/bin:$PATH
export CLASSPATH PATH GRINDERPROPS
```

      Note: Notice this script sets various environment variables that The Grinder needs: where The Grinder is, the location of its properties file, the `CLASSPATH`, and the `PATH`.
   f. View the `startconsole.sh` script.

```
$ cat startconsole.sh
source setgrinderenv.sh
java -classpath $CLASSPATH net.grinder.Console
```

      Note: Notice this script calls the one that sets environment variables, and then it starts the JVM with the `net.grinder.Console` class.
   g. Start The Grinder console.
      `$ ./startconsole.sh`
   h. Leave the console open.

   Note: Now that the console is running, when an agent is started, it waits for signals from the console.

7. Start a Grinder agent on host01.
   a. In the Terminal window on host01, navigate to the practice directory.
      `$ cd /practices/part1/practice10-02`
   b. Run the `startagent.sh` script to start a Grinder agent.
      `$ ./startagent.sh`

```
... INFO  agent: The Grinder 3.11
... INFO  agent: connected to console at host02.example.com/...
... INFO  agent: waiting for console signal
```

Note: The agent is running in this window and waiting for communication from the console.

c. Leave the Terminal window open.

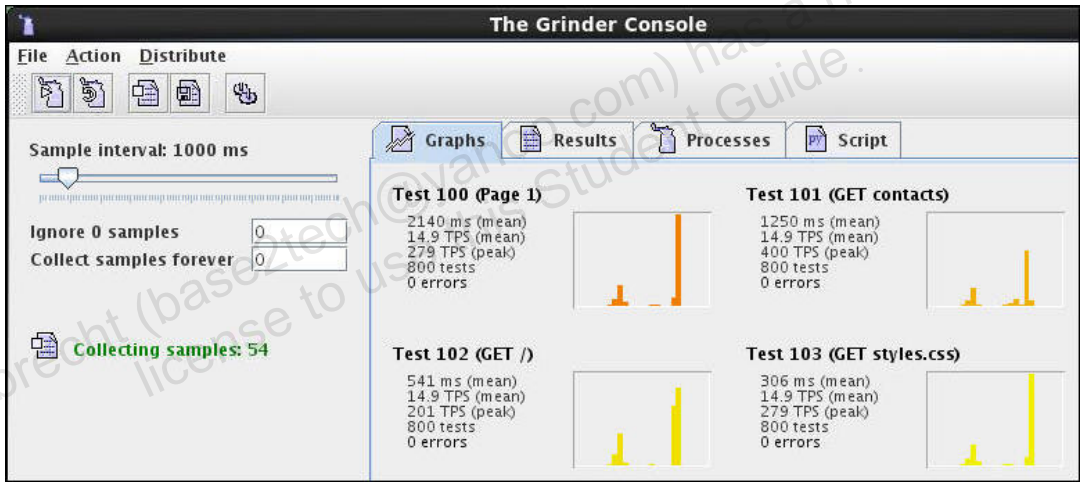
8. From The Grinder console on host02, signal the agent (worker processes) to start working. Then view the data that comes back. After a while, stop collecting data and reset the agent (worker processes).

a. Access The Grinder console on host02.

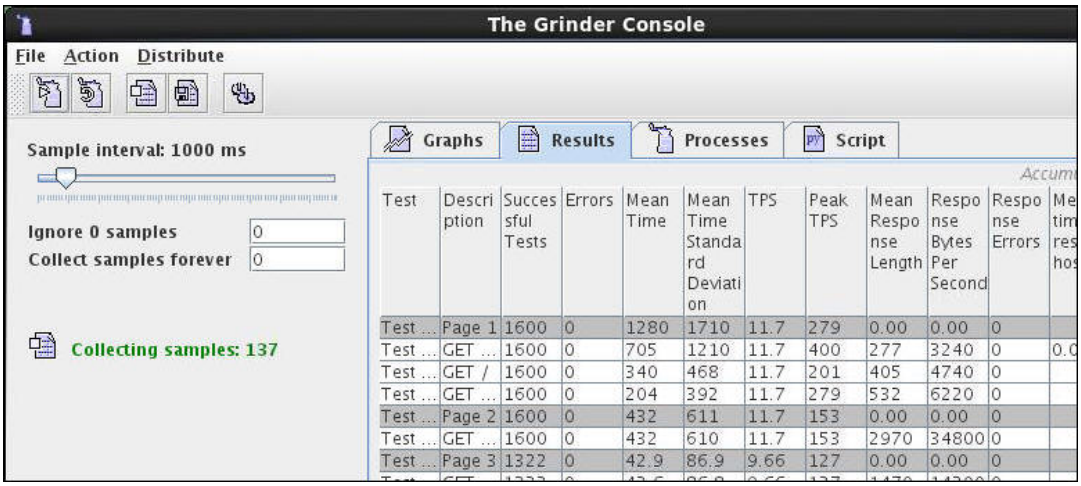
b. Click the button to “Start the worker processes” (the first button on the left).

c. When the dialog box warns that you have not selected a properties file, click **OK**.

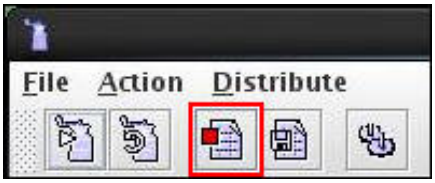
d. Click the **Graphs** tab. Watch the graphs go by for a while.



e. Click the **Results** tab. Watch the numbers update for a while.



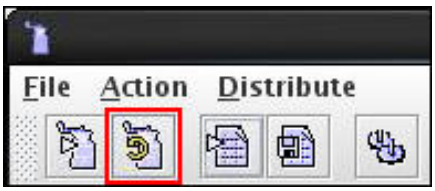
f. Click the “Stop collecting statistics” button.



g. Look at the numbers in the console on the **Results** tab. The “Mean Time” is the number of milliseconds a request takes. Although the script has “think time” in it for each web page accessed (like a real user would have), that is not included in the Mean Time, which is just the time the HTTP request takes.

Note: Remember, this time is in milliseconds. You probably will see much lower than sub-second response time for all the “pages.”

h. Next, click the “Reset the worker processes” button. When asked whether you also want to reset the console, click **No**. (You did not reset the console in case you want to continue to look at the numbers.)



i. When you have finished investigating the numbers, close The Grinder console.

Note: If you are interested, the raw data has been saved in files in a newly created `log` directory in the practices directory.

j. Close the host02 Terminal window.

9. Return to host01 and view the agent window.
   a. The Terminal window in which the agent was running has various messages in it. You should notice that the agent is no longer running. It stopped when it could no longer communicate with the console, which you closed.

```
...
... INFO  agent: finished
$
```

   b. Close the Terminal window.

Practice Solution: Load Testing an Application

Perform the following tasks if you did not complete this practice and want to use the finished solution. Note that this solution does not install The Grinder or run its load test. The solution just deploys the application that was used in the load test.

Assumptions

You completed "Practice 9-1: Configuring and Using Node Manager."

"Practice 10-1: Deploying an Application" is not necessary for this solution.

All servers are currently running.

Solution Tasks

1. Connect to host02 and run the practice setup script which initializes (or reinitializes) the database to create a table and put some data in it that the application requires.

   Note: Notice that this is host02, not host01.

   a. Open a Terminal window and navigate to the practice directory. Set up some environment variables, and then run the script.

   ```
   $ export ORACLE_HOME=/u01/app/db11g/product/11.2.0/dbhome_1
   $ export ORACLE_SID=orcl
   $ cd /practices/part1/practice10-02
   $ ./setup.sh
   ```

   Note: The SQL script drops the table (and sequence) it creates, so you can run it multiple times. Therefore, the first time the script is run, it produces messages that no such table or sequence exist. Ignore those messages if you see them.

   b. Close the Terminal window.

2. Connect to host01 and run the script to deploy the application from which The Grinder script was created.

   a. Open a new Terminal window, navigate to the practice directory, and run the deploy script.

   ```
   $ cd /practices/part1/practice10-02
   $ ./deploy.sh
   ```

   Note: If you get a warning from WLST about closing in a different thread, ignore it (`WLContext.close() was called in a different thread…`). It is a known issue with WLST deployment and does no harm.

   b. Close the terminal window.

Practices for Lesson 12: Clusters

Chapter 12

Practices for Lesson 12: Overview

Practices Overview

In these practices, you create a “regular” cluster and a dynamic cluster. You also deploy an application to the dynamic cluster.

Practice 12-1: Configuring a Cluster

Overview

In this practice, you create and configure a cluster. You start the servers in the cluster.

Assumptions

You completed “Practice 9-1: Configuring and Using Node Manager.”
The administration server is currently running.

Tasks

1. Open the WebLogic Server administration console.
2. Create and configure a cluster.
   a. In the Change Center, click **Lock & Edit**.
   b. In the Domain Structure, expand **Environment** and select **Clusters**.
   c. Above the Clusters table, click the **New** button and select **Cluster**.
   d. Enter or select the following values for these fields and then click **OK**.
      – **Name:** `cluster3` (You are saving the name `cluster2` for the dynamic cluster.)
      – **Messaging Mode:** `Unicast`.
   e. Click **OK**.
   f. Select the new cluster in the Clusters table.
   g. Select the **Configuration** > **Servers** tabs.
   h. Scroll down to the Servers table and click the **Add** button.
   i. Select **Create a new server and add it to this cluster**. Click **Next**.
   j. Enter or select the following values for these fields, then click **Finish**.
      – **Server Name:** `server3`
      – **Server Listen Address:** `host01.example.com`
      – **Server Listen Port:** `7013`
   k. Click the **Add** button again.
   l. Select **Create a new server and add it to this cluster**. Click **Next**.
   m. Enter or select the following values for these fields and then click **Finish**.
      – **Server Name:** `server4`
      – **Server Listen Address:** `host02.example.com`
      – **Server Listen Port:** `7013`
   n. Click the **Add** button again.
   o. Select **Create a new server and add it to this cluster**. Click **Next**.
   p. Enter or select the following values for these fields and then click **Finish**.
      – **Server Name:** `server5`
      – **Server Listen Address:** `host01.example.com`
      – **Server Listen Port:** `7015`
   q. Click the **Add** button again.
   r. Select **Create a new server and add it to this cluster**. Click **Next**.

s. Enter or select the following values for these fields and then click **Finish**.
   - **Server Name:** `server6`
   - **Server Listen Address:** `host02.example.com`
   - **Server Listen Port:** `7015`

t. In the Change Center, click **Activate Changes**.

3. Assign the new servers to the proper machines.
   a. In the Change Center, click **Lock & Edit**.
   b. In the Domain Structure, expand **Environment** and select **Servers**.
   c. Ensure the **Configuration** tab is selected.
   d. Click **server3**.
   e. Ensure that the **Configuration** > **General** tabs are selected.
   f. Use the Machine drop-down list to select **`machine1`**.
   g. Click the **Save** button.
   h. Use the breadcrumbs and select **Summary of Servers**. Click **server4**. Ensure that the **Configuration** > **General** tabs are selected. Use the Machine drop-down list to select **`machine2`**. Click the **Save** button.
   i. Use the breadcrumbs and select **Summary of Servers**. Click **server5**. Ensure that the **Configuration** > **General** tabs are selected. Use the Machine drop-down list to select **`machine1`**. Click the **Save** button.
   j. Use the breadcrumbs and select **Summary of Servers**. Click **server6**. Ensure that the **Configuration** > **General** tabs are selected. Use the Machine drop-down list to select **`machine2`**. Click the **Save** button.
   k. In the Change Center, click **Activate Changes**.

4. Start the new servers in the new cluster.
   a. In the Domain Structure, expand **Environment** and select **Servers**.
   b. Select the **Control** tab.
   c. Select the check boxes next to `server3`, `server4`, `server5`, and `server6`. Then click **Start**. When asked to confirm, click **Yes**.
   d. Wait for a little while and refresh the web browser. Ensure that the state of the new managed servers is “RUNNING.”

Practice Solution: Configuring a Cluster

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

Assumptions

You completed “Practice 9-1: Configuring and Using Node Manager.”

The administration server is currently running.

Tasks

1. Connect to host01.
2. Run the solution script.
   a. Open a Terminal window and navigate to the practice directory. Run the solution script.

      ```
      $ cd /practices/part1/practice12-01
      $ ./solution.sh
      ```

      Note

      - This script calls a WLST script that creates the new servers, the new cluster, and adds the servers to the cluster. It then calls another WLST script to start all the servers in the new cluster.
      - The WLST script that starts the servers can take a while to run, so be patient.

   b. Close the Terminal window.

Practice 12-2: Configuring a Dynamic Cluster

Overview

In this practice, you create and configure a dynamic cluster. You start the servers in the cluster. You also deploy an application to all the servers in the dynamic cluster.

Assumptions

You completed “Practice 9-1: Configuring and Using Node Manager.” (“Practice 12-1: Configuring a Cluster” is not a prerequisite for this practice.)

The administration server is currently running.

Tasks

1. Connect to host02. From there, run a script that sets up the database with a table and loads it with some data. This is required by the application you will deploy later in this practice.

   Note: Notice that this is host02, not host01.

   a. Open a Terminal window and navigate to the practice directory. Set a couple of environment variables and run the script that sets up the database.

   Note: The `ORACLE_HOME` variable points to where the database is installed. `ORACLE_SID` (Oracle System ID) is the unique name of this particular database.

```
$ cd /practices/part1/practice12-02
$ export ORACLE_HOME=/u01/app/db11g/product/11.2.0/dbhome_1
$ export ORACLE_SID=orcl
$ ./setup.sh
...
1 row created


1 row created

Commit complete.
...
$
```

   Note: The SQL script drops the table (and sequence) it creates, so you can run it multiple times. Therefore, the first time the script is run, it produces messages that no such table or sequence exist. Ignore those messages if you see them.

   b. Close the Terminal window.

2. Connect to host01 and copy the application to the domain's application directory.

   a. Open a Terminal window and navigate to the practice directory. Copy the application to the domain's application directory.

```
$ cd /practices/part1/practice12-02
$ cp supplies.war /u01/domains/part1/wlsadmin/apps
```

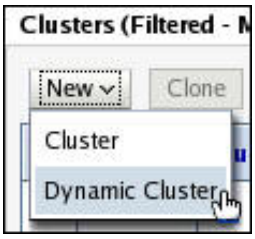
   b. Close the Terminal window.

3. Open the WebLogic Server administration console.

4. Create and configure a dynamic cluster.

   a. In the Change Center, click **Lock & Edit**.

b. In the Domain Structure, expand **Environment** and select **Clusters**.
c. Above the Clusters table, click the **New** button and select **Dynamic Cluster**.



d. Enter or select the following values for these fields and then click **Next**.
   - **Name:** `cluster2`
   - **Messaging Mode:** `Unicast`.

e. Enter or select the following values for these fields and then click **Next**.
   - **Number of Dynamic Servers:** `4`
   - **Server Name Prefix:** `cluster2server-`

Note: That is a "dash" on the end of the prefix name.

f. Select **Use any machine configured in this domain**. Click **Next**.
g. Enter or select the following values for these fields, then click **Next**:
   - **Assign each dynamic server unique listen ports:** Selected
   - **Listen Port for First Server:** `7099`
   - **SSL Listen Port for First Server:** `8099`

h. Notice the details of this new dynamic cluster, including the name of the new server template that will be created. Click **Finish**.
i. In the Change Center, click **Activate Changes**.

5. View the new servers generated for the dynamic cluster. Start those servers.
   a. In the Domain Structure, expand **Environment** and select **Servers**. Ensure that the **Configuration** tab is selected.
   b. Notice the four new servers. Their names all start with "`cluster2server-`" and end in an index.
   c. Notice their type is "Dynamic," as opposed to the other servers, which are "Configured."
   d. Also notice that the servers alternate between `machine1` and `machine2`.
   e. Click the **Control** tab.
   f. First, so not many servers are running at one time, stop all other managed servers. Select the check boxes next to all managed servers not in the new dynamic cluster. Click **Shutdown** and select **Force Shutdown Now**. When asked to confirm, click **Yes**.

   Note: You may have to click the **Next** link and do this more than once if not all servers are displayed on the first page.

   g. Next, start the new dynamic servers. Select the check boxes next to all the dynamic servers: `cluster2server-1`, `cluster2server-2`, `cluster2server-3`, and `cluster2server-4`. Then click **Start**. When asked to confirm, click **Yes**.
   h. Wait for a little while and refresh the web browser. Ensure that the state of all the new managed servers is "RUNNING," and the state of the other managed servers is "SHUTDOWN."

6. Change the data source target to include the new cluster.
   a. In the Change Center, click **Lock & Edit**.
   b. In the Domain Structure, expand **Services** and select **Data Sources**.
   c. In the Data Sources table, click the name of **`datasource1`**.
   d. Click the **Targets** tab.
   e. Select the check box for **`cluster2`**. Ensure that **All servers in the cluster** is selected.
   f. Click **Save**.
   g. In the Change Center, click **Activate Changes**.
7. Deploy an application to the new cluster.
   a. In the Change Center, click **Lock & Edit**.
   b. In the Domain Structure, select **Deployments**.
   c. Above the Deployments table, click the **Install** button.
   d. Use the links next to and below **Current Location** to navigate to:
      `host01.example.com/u01/domains/part1/wlsadmin/apps`

   Note: This fills in the **Path** field as you click links.
   e. When the path is correct, select the option button next to **supplies.war**. Click **Next**.
   f. Select **Install this deployment as an application** and click **Next**.
   g. Select the check box next to **`cluster2`**. Ensure that the **All servers in the cluster** option button is selected. Click **Next**.
   h. Leave all the Optional Setting at their default values. Click **Next**.
   i. Select **No, I will review the configuration later**. Click **Finish**.
   j. In the Change Center, click **Activate Changes**.

   Note: Notice the message that the changes have been activated and no restarts are necessary.
8. Start the application.
   a. In the Deployments table, select the check box next to the **supplies** application.
   b. Click the **Start** button and select **Servicing all requests**.
   c. When asked to confirm, click **Yes**.

   Note: Notice that the State of the supplies application is now “Active.”

Practice Solution: Configuring a Dynamic Cluster

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

Note: If you started the practice and created the dynamic cluster or the server template, you must delete them before running this solution. Use the administration console. If any of the dynamic servers were created and started, first stop them (**Environment** > **Servers** > **Control**; select the servers and click **Shutdown** > **Force Shutdown Now**). Then delete the dynamic cluster (**Lock & Edit**; **Environment** > **Clusters**; select **cluster2** and click **Delete**, then **Yes**; **Activate Changes**). Then delete the server template (**Lock & Edit**; **Environment** > **Clusters** > **Server Templates**; select **cluster2server-Template** and click **Delete**, then **Yes**; **Activate Changes**).

Assumptions

You completed "Practice 9-1: Configuring and Using Node Manager." ("Practice 12-1: Configuring a Cluster" is not a prerequisite for running this solution.)

The administration server is currently running.

Solution Tasks

1. Connect to host02 and run the database setup script.

   Note: Notice that this is host02, not host01.

   a. Open a Terminal window and navigate to the practice directory. Set a couple of environment variables and run the script that sets up the database.

   ```
   $ cd /practices/part1/practice12-02
   $ export ORACLE_HOME=/u01/app/db11g/product/11.2.0/dbhome_1
   $ export ORACLE_SID=orcl
   $ ./setup.sh
   ```

   b. Close the Terminal window.

2. Connect to host01 and run the solution script.

   a. Open a Terminal window, navigate to the practice directory, and run the solution script.

   ```
   $ cd /practices/part1/practice12-02
   $ ./solution.sh
   ```

   Note

   - This script calls a WLST script that creates the server template and the dynamic cluster based on it. It then calls another WLST script to deploy the application to the new cluster. It then calls a WLST script to stop all servers in the other two clusters. Next, it calls a WLST script to start all the servers in the new cluster. Finally, it calls a WLST script to target the data source to the new cluster.
   - The WLST script that starts servers can take a while to run, so be patient.

   b. Close the Terminal window.

Practices for Lesson 13: Clusters

Chapter 13

Practices for Lesson 13: Overview

Practices Overview

In these practices, you install and create an instance of Oracle HTTP Server. You then configure Oracle HTTP Server as the proxy to the dynamic cluster. You test the application deployed to that cluster. You also configure replication groups in another cluster.

Practice 13-1: Installing OHS (Optional)

Overview

In this practice, you install the Oracle HTTP Server (OHS) portion of Oracle Web Tier on host01. You also create an instance of OHS on that host.

OHS is already installed and a Web Tier instance with an OHS process has been created on host01. You will rename the installation directory (that also contains the instance) to gain OHS installation practice. If something goes wrong during the installation, you can revert to the preinstalled version of the product by changing the directory back to its original name.

Assumptions

You completed "Practice 12-2: Configuring a Dynamic Cluster."

Tasks

1. Connect to host01.
2. Rename the OHS installation directory and create a new, empty directory.
   a. Open a Terminal window, and navigate to `/u01/app`. Then rename the OHS installation directory (that also contains the OHS instance).
      ```
      $ cd /u01/app
      $ mv ohs ohs-orig
      $ mkdir ohs
      ```
3. Install the OHS part of Web Tier.
   a. Navigate to the directory that holds the Web Tier installer and execute it.
      ```
      $ cd /install/webtier
      $ ./fmw_12.1.3.0.0_ohs_linux64.bin
      0%.............................
      ```
      Note
      – If the installer file name is different (a different version of OHS), use that file name instead.
      – Wait for all the files to extract before the installer starts. It can take a while.
   b. Use the guidelines in the following table to install the software by using the graphical installer:

Step	Window/Page Description	Directions
a.	Welcome	Click **Next**.
b.	Installation Location	Oracle Middleware Home: **`/u01/app/ohs`** Click **Next**.
c.	Installation Type	Select **Standalone HTTP Server** and click **Next**. **Note:** The environment intentionally uses separate OHS and WebLogic installation locations so they can be maintained separately.
d.	Prerequisite Checks	Wait for the progress bar to reach 100% and all checks to pass.

Step	Window/Page Description	Directions
		Click **Next**.
e.	Security Updates	Deselect **I wish to receive security updates via My Oracle Support**. Click **Next**.
f.	Pop-up window	Click **Yes**.
g.	Installation Summary	Click **Install**.
h.	Installation Progress	Wait for the progress bar to reach 100%. Click **Next**. **Note:** This can take a few minutes.
i.	Installation Complete	Click **Finish**.

4. Configure an OHS instance.
   a. Rename the current OHS domain directory.
      ```
      $ cd /u01/domains
      $ mv ohs ohs-orig
      ```
   b. Navigate to the common Configuration Wizard for OHS.
      ```
      $ cd /u01/app/ohs/oracle_common/common/bin
      ```
   c. Run the Domain Configuration Wizard to create an OHS domain.
      ```
      $ ./config.sh
      ```
   Note: The OHS 12*c* release provides a WebLogic domain template for FMW system components.
   d. Use the guidelines in the following table to configure the OHS domain by using the Configuration Wizard:

Step	Window/Page Description	Directions
a.	Create Domain	Select **Create a new domain**. Enter `/u01/domains/ohs` as the domain location. Click **Next**.
b.	Templates	Select **Oracle HTTP Server (Standalone) – 12.1.3.0 [ohs]**. Click **Next**. **Note:** The Basic Standalone System... template is selected by default.
c.	JDK Selection	Select **Oracle HotSpot 1.7.0_xx /u01/app/ohs/oracle_common/jdk**. Click **Next**.
d.	System Components	Leave all default values and click **Next**. Note the system component name, ohs1.
e.	OHS Server	System Component: `ohs1` Admin Host: `host01.example.com`

Step	Window/Page Description	Directions
		Admin Port: 9999 Listen Address: `host01.example.com` Listen Port: 7777 SSL Listen Port: 7778 Server Name: http://host01.example.com:7777 Click **Next**.
f.	Node Manager	Node Manager Type: Per Domain Default Location Node Manager Credentials: Username: **weblogic**, Password: **Welcome1** Click **Next**.
g.	Configuration Summary	Review and click **Create**.
h.	Configuration Progress	Wait for the progress bar to reach 100%. Click **Next**. **Note:** This can take a few minutes.
i.	Success	Click **Finish**.

e. Verify that the domain files are in place.

```
$ ls -l /u01/domains/ohs
drwxr-x--- 2 oracle oinstall 4096 Sep 23 09:52 bin
drwxr-x--- 4 oracle oinstall 4096 Sep 23 09:52 config
drwxr-x--- 2 oracle oinstall 4096 Sep 23 09:52 init-info
drwxr-x--- 2 oracle oinstall 4096 Sep 23 09:52 nodemanager
drwxr-x--- 2 oracle oinstall 4096 Sep 23 09:52 security
...
```

5. Install patch file for OHS.

```
$ /install/webtier/patch/patch.sh
If no errors are shown, then the patch file successfully
installed.
```

Practice Solution: Installing OHS (Optional)

There is no scripted solution for this practice. If you did not do the practice, there is nothing for you to do, the product is already installed and the OHS instance is created.

If you started the practice, but did not complete it, you revert to the preinstalled version of the products by deleting any new directories you created and renaming the original installation directory back to its previous name.

Solution Tasks

Important: Only do these tasks if you started the practice but did not complete it.

1. Connect to host01.
2. Check that the preinstalled directory is there. Delete the new installation directory that you created. Rename the preinstalled directory back to its original name.
   a. Open a Terminal window, and navigate to `/u01/app`. Then list the directories to ensure the `ohs-orig` directory is present. Check that this preinstalled directory has valid contents.

```
$ cd /u01/app
$ ls
ohs ... ohs-orig ...
$ ls ohs-orig
bin  domain-registry.xml  inventory  lib  nls
oracle_common  oui  precomp  sqlplus  wlserver
...  oraInst.loc  plsql  slax  webgate
```

   b. Ensure you are in the proper directory. Remove the new installation directory that you created.

```
$ pwd
/u01/app
$ rm -rf ohs
```

   Warning: This command permanently deletes the directory, all its subdirectories, and files. Use with caution.

   c. Rename the preinstalled directory back to its original name.

```
$ mv ohs-orig ohs
```

3. Restore the original OHS domain folder.
   a. In a Terminal window on host01, navigate to the `/u01/domains` folder. Then list the directories to ensure the `ohs-orig` directory is present. Check that this preinstalled directory has valid contents.

```
$ cd /u01/domains
$ ls
ohs ... ohs-orig ...
$ ls ohs-orig
auditlogs  bin  config  init-info  nodemanager  security
servers  system_components
```

   b. Ensure you are in the proper directory. Remove the new installation directory that you created.

```
$ pwd
```

```
/u01/domains
$ rm -rf ohs
```

Warning: This command permanently deletes the directory, all its subdirectories, and files. Use with caution.

c. Rename the preinstalled directory back to its original name.

```
$ mv ohs-orig ohs
```

Practice 13-2: Configuring a Cluster Proxy

Overview

In this practice, you configure Oracle HTTP Server (OHS) to act as the proxy to the cluster created in the previous practice. You access the application deployed to the cluster via the proxy, and verify that in-memory session replication works.

Assumptions

You completed "Practice 12-2: Configuring a Dynamic Cluster."

"Practice 13-1: Installing OHS" was completed successfully, or not done at all (OHS is already installed and ready on host01).

The administration server and all managed servers in the new dynamic cluster are currently running.

Tasks

1. Connect to host01.
2. Edit the WebLogic plug-in configuration file for OHS.
   a. Open a Terminal window and navigate to the configuration directory of the OHS instance. Edit both `httpd.conf` and `mod_wl_ohs.conf`.

      ```
      $ cd /u01/domains/ohs/config/fmwconfig/components/OHS/instances/ohs1
      $ gedit httpd.conf mod_wl_ohs.conf
      ```

   b. Check that `httpd.conf` includes the plug-in configuration file. Make sure the **httpd.conf** tab in the editor is selected. Press **Ctrl + F** and in the pop-up window enter **mod_wl_ohs.conf** and click the **Find** button. Notice the `include` directive that includes this configuration file.
   c. Close the Find pop-up window. Close the `httpd.conf` file (click the "X" in the tab) without making any changes to it.
   d. Edit the plug-in configuration file, `mod_wl_ohs.conf`. Notice that most lines are commented out (they start with a "#"). Also notice the `LoadModule` line, which loads the WebLogic plug-in.
   e. In between `<IfModule weblogic_module>` and `</IfModule>` enter following lines:

      ```
      <Location />
        WLSRequest On
        WebLogicCluster
                    host01.example.com:7100,host02.example.com:7101
      </Location>
      ```

      Note
      - Enter the `WebLogicCluster` line on a single line in the file.
      - This initial list does not include all servers in the cluster, nor does it have to. The plug-in will be updated with the other servers in the cluster when the cluster servers it uses return the "dynamic cluster list" to it.

– The / parameter set in Location tells the proxy that requests for all locations ("slash") should go to the cluster, no matter what the context root of the web applications

Tip: Make sure there is no space between the "/" and the ">" or the line will not work.

f. When finished editing this file, it should look like this:

```
LoadModule weblogic_module
                      "${PRODUCT_HOME}/modules/mod_wl_ohs.so"
<IfModule weblogic_module>
#  WebLogicHost <WEBLOGIC_HOST>
#  WebLogicPort <WEBLOGIC_PORT>
<Location />
   WLSRequest On
   WebLogicCluster
              host01.example.com:7100,host02.example.com:7101
</Location>
</IfModule>

#<Location /weblogic>
#  SetHandler weblogic-handler
#  PathTrim /weblogic
#  ErrorPage  http:/WEBLOGIC_HOME:WEBLOGIC_PORT/
#</Location>
```

Note: The file is shown with some of the comment lines removed. You can keep them, if you prefer. Also, due to space limitations, the LoadModule line is shown here on two lines, but in the file it is on one line.

g. When you are satisfied that the file is correct, save it and exit the editor.

3. Configure OHS Node Manager to use a non-SSL connection.

You have to modify two configuration settings to make Node Manager use a non-SSL connection. Add one setting in the domain's config.xml file and the other setting in the nodemanager.properties file.

a. Open the domain's config.xml file for editing.

```
$ gedit /u01/domains/ohs/config/config.xml
```

b. Add the <nm-type> line to the machine's node-manager element, and then save and close the file.

```
<machine>
    <name>localmachine</name>
    <node-manager>
      <name>localmachine</name>
      <nm-type>Plain</nm-type>
      <listen-address>localhost</listen-address>
      ...
```

c. Open the nodemanager.properties file for editing.

```
$ gedit /u01/domains/ohs/nodemanager.properties
```

d. Set the `SecureListener` property to `false`, and then save and close the file.

```
SecureListener=false
```

4. Start OHS.

a. Open a new Terminal window, name the window **OHS NM**, and navigate to the OHS domain's `bin` folder.

```
$ cd /u01/domains/ohs/bin
```

b. Execute the following commands to explore the `bin` folder and start the OHS Node Manager.

```
$ ls
startComponent.sh    startRSDaemon.sh  stopNodeManager.sh
startNodeManager.sh  stopComponent.sh  stopRSDaemon.sh
$ ./startNodeManager.sh
...
ohs -> /u01/domains/ohs

<Sep 24, 2014 4:40:35 AM PDT> <INFO> <Node manager v12.1.3
#1604337>
<Sep 24, 2014 4:40:36 AM PDT> <INFO> <Secure socket listener
started on port 5556, host localhost/127.0.0.1>
```

Note: Wait for the command to finish.

c. Open a new Terminal window, name the window **OHS**, and navigate to the OHS domain's `bin` folder.

```
$ cd /u01/domains/ohs/bin
```

d. Execute the following command to start OHS.

```
$ ./startComponent.sh ohs1
Starting system Component ohs1 ...
...
Reading domain from /u01/domains/ohs
Please enter Node Manager password:
```

e. Enter **`Welcome1`** as the Node Manager password.

```
Connecting to Node Manager ...
Successfully Connected to Node Manager.
Starting server ohs1 ...
Successfully started server ohs1 ...
Successfully disconnected from Node Manager.
...
```

f. Execute the following command to check the status of OHS.

```
$ /u01/app/fmw/wlserver/common/bin/wlst.sh
/install/webtier/config/status.py
Connecting to Node Manager ...
Successfully Connected to Node Manager.

RUNNING
```

Note

- Make sure that the “status” is “RUNNING.”

5. Access the application deployed to the cluster through the OHS cluster proxy. Use the application to ensure it creates a session for you.
   a. Open your web browser and enter the OHS URL, followed by `/supplies`:
      `http://host01.example.com:7777/supplies`
   b. When the application home page displays, click the **browse the catalog** link.
   c. Click the **[add]** link next to a few items.
   d. Scroll down and view the items in your shopping cart.

   Note: Remember what items are in your shopping cart.

   e. Do not close the web browser window. For the admin console, use a different browser window. You will come back to this web browser window later.
6. Open the WebLogic Server administration console in a new tab or window.
7. Use the administration console to see which cluster server is servicing your requests. Stop that server.
   a. Under Domain Structure, expand **Environment** and select **Clusters**.
   b. In the Clusters table, click **cluster2**.
   c. Click the **Monitoring** tab.
   d. Use the **Primary** column to determine which of the clustered servers is hosting your current shopping cart session (the value should be greater than `0`). Remember which server it is.

   Note: You can also see which server holds the backup copy of the session. Look at the **Secondary Distribution Names** column. For example, if `cluster2server-1` is the primary, another server will have a value in this column of: `cluster2server-1:1`. What this means is that this server has `1` secondary session for `cluster2server-1`.

   Tip: If you cannot tell which server is the primary, write down the Primary session count number for all the servers. Close all browsers. Open a new browser and access and use the application again. In the admin console, come back to this table and see which Primary number has increased. That is the server that currently holds your session and is servicing your requests.

   e. Stop the server that has your session. In the Domain Structure, expand **Environment** and select **Servers**. Click the **Control** tab. Select the check box next to the server that is hosting your session. Click the **Shutdown** button and select **Force Shutdown Now**. When asked whether you are sure, click **Yes**.
   f. Wait a little while and refresh the web browser. Ensure that the state of the server is SHUTDOWN.
8. Ensure your shopping cart still has your items to show that in-memory session replication is working.
   a. Go back to the web browser in which you were using the application. Remember the items that were in your shopping cart. Click the **[add]** link next to some new and different item. Scroll down and view your shopping cart. The new item and all the old items should display in the shopping cart.
9. Shut down all of the servers in cluster2.

Practice Solution: Configuring a Cluster Proxy

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

Assumptions

You completed “Practice 12-2: Configuring a Dynamic Cluster.”

The administration server and all managed servers in the new dynamic cluster are currently running.

Solution Tasks

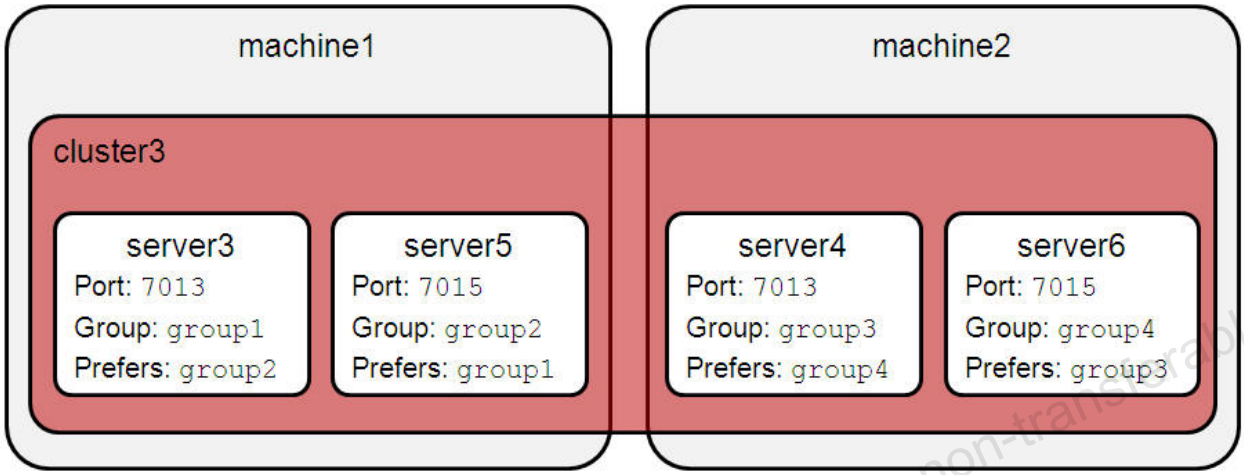
1. Connect to host01.
2. Run the script.
   a. Open a Terminal window and navigate to the practice directory. Then run the script.
      ```
      $ cd /practices/part1/practice13-02
      $ ./solution.sh
      ```
      Note:
      This script performs the following tasks:
      - Copies a completed WebLogic plug-in configuration file for OHS to the right location
      - Configures the OHS Node Manager to use a non-SSL connection
      - Starts the OHS Node Manager
      - Starts OHS
      - Checks the status of the OHS server
      - You should see the print out in the Terminal window that `ohs1` is `RUNNING`.

   b. Close the Terminal window.
3. If you want to access the application deployed to the cluster (in a previous practice) through the OHS proxy, use the web browser.
   a. Enter this URL:
      `http://host01.example.com:7777/supplies`

Practice 13-3: Configuring Replication Groups

Overview

In this practice, you configure replication groups in a cluster.



Assumptions

You completed "Practice 12-1: Configuring a Cluster."

The administration server is running.

Tasks

1. Connect to host01.
2. Open the WebLogic Server administration console.
3. Set replication groups and preferred secondary groups.

   Note: This is not a realistic example because you would not want to set up replication groups so that a server's secondary server was on its same machine. It does give you practice in configuring replication groups, however. And, if you do the optional task at the end of this practice, it will prove that replication group settings take precedence over the "choose a different machine" preference when a secondary server is selected.

   a. First, stop all the servers in cluster3, if they are running. In the Domain Structure, expand **Environment** and select **Clusters**. In the Clusters table, select **cluster3**. Click the **Control** tab. Select the check box in front of all the servers in the cluster that are RUNNING. Click **Shutdown** and select **Force Shutdown Now**. When asked to confirm, click **Yes**.

   Why? Changing a server's replication group or preferred secondary group does not take effect until the server restarts. You are shutting down these servers ahead of time and will start them later.

   b. In the Change Center, click **Lock & Edit**.
   c. In the Domain Structure, expand **Environment** and select **Servers**.
   d. In the Servers table, select **server3**.
   e. Click the **Configuration** > **Cluster** tabs.
   f. Enter the following values for these fields, then click **Save**.
      - Replication Group: **`group1`**

- Preferred Secondary Group: `group2`

g. Use the breadcrumbs to select **Summary of Servers**. In the Servers table, select **server5**. Click the **Configuration** > **Cluster** tabs. Enter the following values for these fields, then click **Save**.
   - Replication Group: `group2`
   - Preferred Secondary Group: `group1`

Note: Notice that you are putting two servers on the same machine in different groups, each preferring the other's group.

h. Use the breadcrumbs to select **Summary of Servers**. In the Servers table, select **server4**. Click the **Configuration** > **Cluster** tabs. Enter the following values for these fields, then click **Save**.
   - Replication Group: `group3`
   - Preferred Secondary Group: `group4`

i. Use the breadcrumbs to select **Summary of Servers**. In the Servers table, select **server6** (you may have to click the **Next** link to be able to select this server). Click the **Configuration** > **Cluster** tabs. Enter the following values for these fields and then click **Save**.
   - Replication Group: `group4`
   - Preferred Secondary Group: `group3`

j. In the Change Center, click **Activate Changes**.

4. Start the servers in this cluster.
   a. To cut down on the number of concurrently running servers, stop all other managed servers that are running. In the Domain Structure, expand **Environment** and select **Servers**. Click the **Control** tab. Select the check boxes next to all *managed* servers *not* in cluster3 that are RUNNING. Click **Shutdown** and select **Force Shutdown Now**. When asked to confirm, click **Yes**.
   b. In the Domain Structure, expand **Environment** and select **Clusters**.
   c. In the Clusters table, select **cluster3**.
   d. Click the **Control** tab.
   e. Select the check box in front of all four servers. Click **Start**. When asked to confirm, click **Yes**. Wait a while and refresh the page. Ensure that the State of all four servers is RUNNING.
5. (Optional if you have time and are interested) Check to see whether the replication group configuration overrides the "chose a different machine" preference. To do this, you need to target a data source to this cluster, deploy an application that uses in-memory session replication to this cluster, run the application, and then see which server is chosen as the secondary server to the primary server servicing your requests.
   a. In the Change Center, click **Lock & Edit**.
   b. In the Domain Structure, expand **Services** and select **Data Sources**.
   c. In the Data Sources table, click on the name of **datasource1**.
   d. Click the **Targets** tab.
   e. Select the check box for **cluster3**. Ensure that **All servers in the cluster** is selected.
   f. Click **Save**.
   g. In the Domain Structure, select **Deployments**.
   h. In the Deployments table, click the **contacts** application.

i. Click the **Targets** tab.

j. Select the check box next to **cluster3**. Ensure that the **All servers in the cluster** option button is selected.

k. Click **Save**.

l. In the Change Center, click **Activate Changes**.

m. In the Domain Structure, select **Deployments**. Ensure the **contacts** application is Active. If not, start servicing all requests.

n. Do not close the admin console.

o. In the web browser, open a new window or tab.

p. Enter the URL for the application running on server3:

```
http://host01.example.com:7013/contacts
```

Note: A proxy for this cluster is not set up, so you are accessing the application directly on server3.

q. Click the link **browse all contacts**.

r. Return to the admin console. In the domain structure, expand **Environment** and select **Clusters**. In the Clusters table, select **cluster3**.

s. Click the **Monitoring** tab.

t. In the **server3** row, you should see a `1` in the **Primary** column. Look in the **Secondary Distribution Names** column in all of the rows. If the replication group configuration works as suspected, you should see in the **server5** row this value:

```
server3:1
```

Note: Remember, server3 has “group1” as its replication group and “group2” as its preferred secondary group. And server5 has “group2” as its replication group. So, even though server5 is on the same machine as server3, it is still chosen as the secondary server because of the replication group configuration.

Practice Solution: Configuring Replication Groups

There is no solution for this practice. Setting the replication groups is not needed in the cluster.

Practices for Lesson 14: Clusters

Chapter 14

Practices for Lesson 14: Overview

Practices Overview

In this practice, you configure a replication channel for a cluster.

Practice 14-1: Configuring a Replication Channel

Overview

In this practice, you configure a replication channel for peer-to-peer communication in a cluster.

Assumptions

You completed “Practice 9-1: Configuring and Using Node Manager.”

The administration server is running.

To do the optional task of monitoring the replication channel, you must have also completed “Practice 10-2: Load Testing an Application,” or deploy the contacts application manually.

Tasks

1. Connect to host01.
2. Open the WebLogic Server administration console.
3. So unnecessary servers are not running, shut down all managed servers.

   Note: You are also shutting down servers you want to be running, because the changes being made only take effect after a server restart, so you are shutting down these servers ahead of time.

   a. In the Domain Structure, expand **Environment** and select **Servers**.
   b. Click the **Control** tab.
   c. Select the check box in front of all RUNNING managed servers.
   d. Click **Shutdown** > **Force Shutdown Now**. Click **Yes** when asked to confirm. Wait for the State of the servers to be SHUTDOWN.

   Note: You may need to click the **Next** link to access all the servers. Select any running managed servers on the next page and shut them down, too.
4. Configure a replication channel for each server in cluster1.
   a. In the Change Center, click **Lock & Edit**.
   b. In the Domain Structure, expand **Environment** and select **Servers**.
   c. Select **server1** in the Servers table.
   d. Click the **Protocols** > **Channels** tab.
   e. Click the **New** button.
   f. Enter or select the following values for these fields and then click **Next**.
      – **Name:** `ReplicationChannel`
      – **Protocol:** `t3`

   Note: Case matters in this name, as you want it to match everywhere you enter it.

   g. Enter or select the following values for these fields, then click **Next**.
      – **Listen Address:** `host01.example.com`
      – **Listen Port:** `6000`
      – **External Listen Address:** (blank)
      – **External Listen Port:** (blank)
   h. Enter or select the following values for these fields and then click **Next**.
      – **Enabled:** Selected
      – **Tunneling Enabled:** Not selected

- **HTTP Enabled for This Protocol:** Selected
- **Outbound Enabled:** Selected

i. Enter or select the following values for these fields and then click **Finish**.
- **Two Way SSL Enabled:** Not selected
- **Client Certificate Enforced:** Not selected

j. Use the breadcrumbs to select **Summary of Servers**. Select **server2**. Click the **Protocols** > **Channels** tab. Click the **New** button. Go through the Create a New Network Channel wizard again, using the following values:

Field	Value
Name	`ReplicationChannel`
Protocol	`t3`
Listen Address	`host02.example.com`
Listen Port	`6000`
External Listen Address	(blank)
External Listen Port	(blank)
Enabled	Selected
Tunneling Enabled	Not selected
HTTP Enabled for This Protocol	Selected
Outbound Enabled	Selected
Two Way SSL Enabled	Not selected
Client Certificate Enforced	Not selected

k. In the Change Center, click **Activate Changes**.

5. Configure the cluster to use the new replication channel.
a. In the Change Center, click **Lock & Edit**.
b. In the Domain Structure, expand **Environment** and select **Clusters**.
c. Select **cluster1** in the Clusters table.
d. Click the **Configuration** > **Replication** tab.
e. Ensure that `ReplicationChannel` is in the **Replication Channel** field.
f. Click the **Save** button.
g. In the Change Center, click **Activate Changes**.

Note: If you did not type into the field, then no changes were made and the **Activate Changes** button will still say **Release Configuration**. If that is the case, click **Release Configuration**.

6. Start all cluster1 servers.
a. In the Domain Structure, expand **Environment** and select **Clusters**.
b. Select **cluster1** in the Clusters table.
c. Click the **Control** tab.
d. Select the check box in front of all cluster1 servers. Click **Start**. Click **Yes** when asked to confirm. Wait for the State of the servers to be RUNNING.
e. Do not close the web browser.

7. Verify that the replication channel started by looking at server1's output.
   a. Open a new Terminal window on host01 and navigate to the `logs` directory of server1. Then edit the `server1.out` file.
      ```
      $ cd /u01/domains/part1/wlsadmin/servers/server1/logs
      $ gedit server1.out
      ```
   b. Press **Ctrl + F**. In the Find pop-up window, enter **`ReplicationChannel`** in the **Search for** field and press **Enter**.
   c. You should see the message that this channel is listening on port `6000` for `t3`, `CLUSTER-BROADCAST`, and `http`.
   d. Close the Find window, and exit the editor.
   e. Close the Terminal window.
8. (Optional if you have time and are interested) Monitor the replication channel after session replication has taken place. To do this you need to target a data source to this cluster, deploy an application that uses in-memory session replication to this cluster, and run the application.

   Note: If the data source and application are already targeted to cluster1 you can skip those steps when you get to them.
   a. Return to the admin console.
   b. In the Change Center, click **Lock & Edit**.
   c. In the Domain Structure, expand **Services** and select **Data Sources**.
   d. In the Data Sources table, click the name of **datasource1**.
   e. Click the **Targets** tab.
   f. If the data source is already targeted to cluster1, move on to the next step. If not, select the check box for **cluster1**. Ensure **All servers in the cluster** is selected. Click **Save**.
   g. In the Domain Structure, select **Deployments**.
   h. In the Deployments table, click the **contacts** application.
   i. Click the **Targets** tab.
   j. If the application is already targeted to cluster1, move on to the next step. If not, select the check box next to **cluster1**. Ensure that the **All servers in the cluster** option button is selected. Click **Save**.
   k. In the Change Center, click **Activate Changes**.

   Note: If the data source and the contacts application were both already targeted to cluster1, instead click **Release Configuration**.
   l. Do not close the admin console.
   m. In the web browser, open a new window or tab.
   n. Enter the URL for the application running on server1:
      `http://host01.example.com:7011/contacts`

   Note: A proxy for this cluster is not set up, so you are accessing the application directly on server1.
   o. Click the **browse all contacts** link.
   p. Return to the admin console. In the Domain Structure, expand **Environment** and select **Servers**. In the Servers table, select **server1**.
   q. Click the **Protocols** > **Channels** tab.
   r. In the Network Channels table, select the **ReplicationChannel** channel.

s. Click the **Monitoring** > **Overview** tab.

t. Review the statistics for the channel. You should see that messages have been sent and received on the replication channel.

u. You can go view the statistics for the replication channel on **server2**, as well.

Practice Solution: Configuring a Replication Channel

There is no solution for this practice. Setting a replication channel is not needed in this cluster.

Practices for Lesson 16: WebLogic Server Security

Chapter 16

Practices for Lesson 16: Overview

Practices Overview

In this practice, you configure an external LDAP and set it as one of the authentication providers of the WebLogic Server security realm.

Practice 16-1: Configuring an Authentication Provider

Overview

In this practice, you configure an external LDAP, OpenDS. You then set this LDAP, which is installed on host02, as one of the authentication providers in your WebLogic domain. Finally, you configure the authentication provider control flags.

Note: The default embedded LDAP continues as one of the authentication providers and retains all the administrative users, groups, and roles. The external LDAP is used for all other users, groups, and roles.

Assumptions

You completed "Practice 13-2: Configuring a Cluster Proxy."

The administration server and the four servers in the dynamic cluster are currently running.

OHS is running.

Tasks

1. Create some LDAP users in the external LDAP system.
   a. Connect to host02.

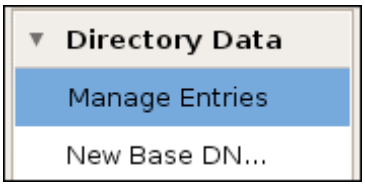
   Note: Notice that this is host02, not host01.

   b. Open a Terminal window. Navigate to the `bin` directory under the installed OpenDS LDAP system, and launch the OpenDS control panel:

   ```
   $ cd /u01/app/ldap/bin
   $ ./control-panel
   ```

   c. A dialog box appears warning that the Local Server is not running. Click **OK** to close it.
   d. Click the **Start** button.

   e. When prompted, enter the password, `Welcome1`, and click **OK**.
   f. After the dialog box indicates that the LDAP server has successfully started, click the **Close** button on the dialog box.
   g. In the left menu, select **Directory Data > Manage Entries**.



   h. Click the base node in the list on the left, **dc=example,dc=com**. Then right-click and select **New User**.

i. Enter or select the following values and then click **OK**.

Field	Value
First Name	`Larry`
Last Name	`Fine`
Common Name	`Larry Fine`
User ID	`larryf`
Password	`Welcome1`
Password (Confirm)	`Welcome1`
E-mail	`larry.fine@example.com`
Naming Attribute	`uid`

Note: Leave blank any fields not listed.

j. After the user has been successfully created, close the “Entry created” dialog box.

Note: It may be behind other windows.

k. Click the base node again, **dc=example,dc=com**. Right-click and select **New User**. Then enter or select the following values and click **OK**.

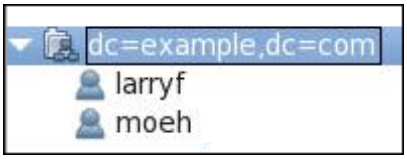
Field	Value
First Name	`Moe`
Last Name	`Howard`
Common Name	`Moe Howard`
User ID	`moeh`
Password	`Welcome1`
Password (Confirm)	`Welcome1`
E-mail	`moe.howard@example.com`
Naming Attribute	`uid`

Note: Leave blank any fields not listed.

l. After the user has been successfully created, close the “Entry created” dialog box.

Note: It may be behind other windows.

m. Confirm that you have two new users within the base DN.



2. Create a group in the external LDAP system and add a user to it.
   a. Click the base node, **dc=example,dc=com** again. Right-click and this time select **New Group**.
   b. For **Name**, enter `boss`.
   c. Ensure that **Static Group** is selected.
   d. Click the **Add Members** button.
   e. Select the **moeh** user and click **OK**.
   f. Click **OK** again.
   g. Close the "Entry created" dialog box.

   Note: It may be behind other windows.
   h. Close the Manage Entries dialog box.
   i. Exit the control panel.

   Tip: Click the "**X**" at the top right, or select **File** > **Exit**.
   j. Close the Terminal window.
3. Open the WebLogic Server administration console.
4. Add the external LDAP system as an authentication provider to the domain's security realm.
   a. In the Change Center, click the **Lock & Edit** button.
   b. In the Domain Structure, select **Security Realms**.
   c. In the Realms table, click **myrealm**.
   d. Click the **Providers** > **Authentication** tabs.
   e. Click the **New** button.
   f. Enter or select the following, then click **OK**.

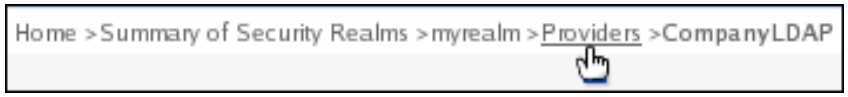
Field	Value
Name	`CompanyLDAP`
Type	`LDAPAuthenticator`

   g. In the Authentication Providers table, click the name of the new provider.
   h. Click the **Configuration > Provider Specific** tabs.
   i. Enter the following values, then click **Save**.

Field	Value
Host	`host02.example.com`
Port	`7878`
Principal	`cn=Directory Manager`
Credential	`Welcome1`
Confirm Credential	`Welcome1`
User Base DN	`dc=example,dc=com`
User Name Attribute	`uid`
Group Base DN	`dc=example,dc=com`
Static Group Name Attribute	`cn`

   Note: Any fields not mentioned should be left at their default values.

5. Adjust the authentication provider processing order and control flags.
   a. Use the breadcrumbs (at the top of the administration console) to return to the Authentication Providers table.



   b. Click the **Reorder** button.
   c. Select providers and use the arrow buttons (▲ ▼) to put the authentication providers in the following order:
      – CompanyLDAP
      – DefaultAuthenticator
      – DefaultIdentityAsserter
   d. Once the order is correct, click **OK**.
   e. In the Authentication Providers table, click the **CompanyLDAP** provider.
   f. Ensure that the **Configuration > Common** tabs are selected.
   g. Set the **Control Flag** to **Sufficient**.
   h. Click **Save**.
   i. Similarly, edit **DefaultAuthenticator** and set its **Control Flag** to **Sufficient** as well. Save that change.
   j. In the Change Center, click **Activate Changes**.

   Note: Notice the message that some items must be restarted. You need to stop and start all servers that are running. (The admin server and the managed servers need to access the new external LDAP.)
6. Navigate to the practice folder and execute the following script to disable the user cache.

```
$ cd /practices/part1/practice16-01
$ ./disable_cache.sh
```

7. Stop all the managed servers.
   a. In the Domain Structure, expand **Environment** and then select **Servers**.
   b. On the right, click the **Control** tab.
   c. Select the check box next to any managed server that is running. Click the **Shutdown** button and select **Force Shutdown Now**. When asked to confirm, click **Yes**.
   d. Wait a moment and refresh the screen. Ensure that the “State” column in the table confirms that all managed servers are shut down.

8. Stop and restart the administration server.
   a. Connect to host01.
   b. Find the Terminal window in which the admin server is running.
   c. Press **Ctrl + C** in the admin server window.

   Note: This has the same effect as Force Shutdown Now.
   d. Use the up arrow to bring back the command issued to start the admin server, and start it again.

      ```
      $ ./startWebLogic.sh
      ```
9. Access the WebLogic Server administration console again.
10. Restart all the managed servers in cluster2.
    a. In the Domain Structure, expand **Environment** and then select **Clusters**.
    b. Click on the name of **cluster2**.
    c. Click the **Control** tab.
    d. Select the check box next to all the managed servers in cluster2. Click the **Start** button. When asked to confirm, click **Yes**.
    e. Wait a moment and refresh the screen. Ensure that the “State” column in the table confirms that all these managed servers are RUNNING.
11. Deploy and start an application that uses role-based security. This application uses its deployment descriptors to:
    - Define a role called `manager`.
    - Protect certain resources, which only users in the `manager` role are allowed to access.
    - Map the `manager` role to a principal called `boss`.

    Therefore, any user in the `boss` group should be allowed to access the protected resources.
    a. Open a new Terminal window on host01 and navigate to the current practice directory. Copy the new application to the domain's application directory.

       ```
       $ cd /practices/part1/practice16-01
       $ cp timeoff.war /u01/domains/part1/wlsadmin/apps
       ```
    b. Return to the web browser running the admin console.
    c. In the Change Center, click **Lock & Edit**.
    d. In the Domain Structure, select **Deployments**.
    e. Above the Deployments table, click **Install**.
    f. Using the links, navigate to:
       `host01.example.com/u01/domains/part1/wlsadmin/apps`
    g. Select the option button in front of **timeoff.war** and click **Next**.
    h. Ensure that **Install this deployment as an application** is selected. Click **Next**.
    i. Select **cluster2**. Ensure that **All servers in the cluster** is selected. Click **Next**.
    j. Do not change any optional settings. Click **Next**.
    k. Select **No, I will review the configuration later**. Click **Finish**.
    l. In the Change Center, click **Activate Changes**.
    m. Return to the Deployments table. Select the check box in front of the **timeoff** application.

n. Click **Start** and select **Servicing all requests**. Click **Yes**.
o. Ensure the state of the timeoff application is now "Active."
p. Close the web browser. If any other web browsers are open, close them, too.

Why? You do not want to be logged in to anything.

12. Use the new application. Log in as a user that does not have the correct role to access the protected resources.
    a. Open the web browser.
    b. In the web browser, enter the URL to access the new timeoff application:
       `http://host01.example.com:7777/timeoff`
    c. The timeoff application opens.
    d. Click the **Request Time Off** link.

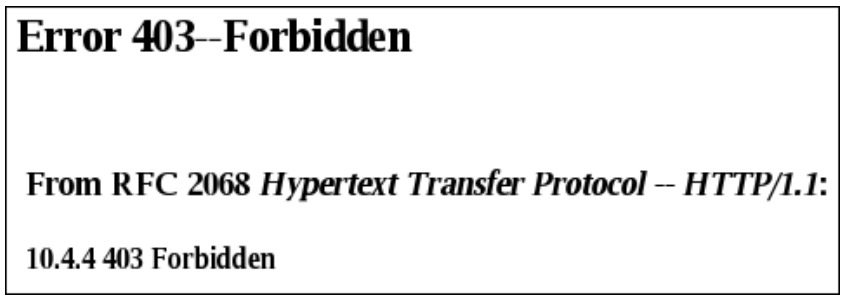
    e. Fill out the form and click the **Submit Report** button.
    f. Click the **Back to Home Page** link.

Note: This part of the application is not protected, so it is open to anyone.

    g. Now try the **Close an Office** link.

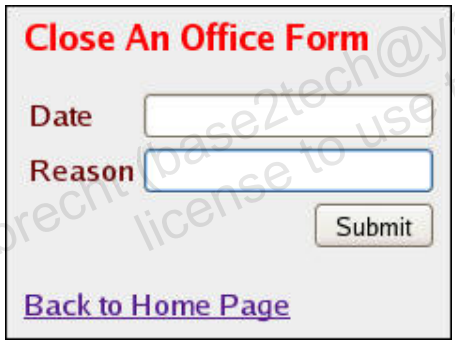
    h. When the "Log In" page displays, enter the username and password of the non-manager and then click **Log In**.
       – **User Name:** `larryf`
       – **Password:** `Welcome1`

    i. Since Larry does not have the correct role, you get an error.



j. Close the web browser.

13. Use the new application again. This time, log in as a user that does have the correct role to access the protected resources.
    a. Open the web browser.
    b. In the web browser, enter the URL to access the new timeoff application:
       `http://host01.example.com:7777/timeoff`
    c. The timeoff application opens.
    d. Click the **Close an Office** link.
    e. When the "Log In" page displays, enter the username and password of the manager and then click **Log In**.
       - **User Name:** `moeh`
       - **Password:** `Welcome1`
    f. This time, because Moe is a manager, you are taken to the "Close an Office Form."



Note: The user Moe (`uid: moeh`) was placed in the `boss` group. The `boss` principal was mapped to the role `manager` by the application's deployment descriptor. The protected resources (particular pages in the web application) allow the role `manager` to have access.

    g. Enter a **Date** and a **Reason** and click the **Submit Form** button.
    h. On the next page, click the **Back to Home Page** link.
    i. Close the web browser.

Practice Solution: Configuring an Authentication Provider

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

Assumptions

You completed “Practice 13-2: Configuring a Cluster Proxy.”

The administration server and the four servers in the dynamic cluster are currently running.

OHS is running.

Solution Tasks

1. Connect to host02.

   Note: Notice that this is host02, not host01.

2. Run the solution script.

   a. In a Terminal window, run the solution script.

   ```
   $ cd /practices/part1/practice16-01
   $ ./solution.sh
   ```

   Note: This script runs an executable under the `bin` directory of the LDAP server that imports users and a group. It then calls another executable that starts the LDAP server. Next, it calls a WLST script that creates the external LDAP authentication provider within the domain. Finally, it calls two more scripts: one stops all the servers in cluster2, the next starts them up again. (Servers must be rebooted for the authentication provider changes to take effect.) Stopping and starting the servers can take a while, so be patient.

   b. Close the Terminal window.

3. Connect to host01. Run a script to deploy an application to the dynamic cluster. Then stop and restart the administration server.

   a. In a Terminal window, run the deployment script.

   ```
   $ cd /practices/part1/practice16-01
   $ ./deploy.sh
   ```

   Note: If you get the WLContext.close() warning, ignore it.

   b. Close the Terminal window.

   c. Find the window in which the admin server is running. Press **Ctrl + C**.

   d. Once the server has stopped, use the up arrow to bring back the last command entered (`startWebLogic.sh`). Press enter to run it.

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

   Note: If you do not have a boot identity file, when prompted, enter the username and password (**`weblogic`** and **`Welcome1`**).

   e. Wait for the server to indicate it is in RUNNING mode.

Practices for Lesson 17: Backing Up a Domain and Upgrading WebLogic Server

Chapter 17

Practices for Lesson 17: Overview

Practices Overview

In this practice, you back up a domain. You then restore from that backup.

Practice 17-1: Backing Up and Restoring a Domain

Overview

In this practice, you back up the domain. You stop servers and remove the existing domain directories from one of the hosts. You restore the domain from the backup and restart the servers on that machine to ensure the backup is working properly.

You also set up the domain to automatically back up the configuration when changes are made.

Assumptions

You completed "Practice 13-2: Configuring a Cluster Proxy."

The administration server and the four servers in the dynamic cluster are currently running.

OHS is running.

Tasks

1. Lock the domain configuration by using the WebLogic administration console.
   a. In the Change Center, click **Lock & Edit**.

   Note: Although in the classroom, no other administrator will be making configuration changes to your domain, in the real world this step is recommended to keep other administrators from making changes to the domain configuration while the backup is being created.
2. Create a backup of the domain on host01.

   Note: You are just backing up the domain. This is just one of the tasks you would perform in an online backup.
   a. Open a Terminal window on host01. Navigate to the directory that holds the domain: `/u01/domains/part1`.

      ```
      $ cd /u01/domains/part1
      ```
   b. Create a backup of the domain.

      ```
      $ tar -zcpvf /home/oracle/wlsadminbackup.tarz wlsadmin
      ```

   Note: The options are:
   - `z`: Zip the archive
   - `c`: Create archive
   - `p`: Preserve file and directory permissions
   - `v`: Verbose output
   - `f`: File name
3. Use the admin console to set up automatic domain configuration backups.
   a. Return to the admin console.
   b. In the Domain Structure, click on the domain, **wlsadmin**.
   c. Ensure that the **Configuration** and **General** tabs are selected.
   d. Scroll down and click **Advanced**.
   e. Scroll down again and select **Configuration Archive Enabled**.
   f. Set **Archive Configuration Count** to **5**.
   g. Click the **Save** button.
   h. In the Change Center, click **Activate Changes**.
4. Make sure the cluster2 servers on host02 (machine2) are running.

a. In the Domain Structure, expand **Environment** and select **Clusters**.
b. In the Clusters table, select **cluster2**.
c. Click the **Control** tab.
d. Make sure the servers that run on machine2 are running. If not, select their check boxes, click **Start**. Then click **Yes**.

5. Stop all servers in the domain that are running on host01 (machine1).
   a. In the Domain Structure, expand **Environment** and select **Servers**.
   b. Click the **Control** tab.
   c. Select the check box in front of any managed servers that are currently running on machine1 (which is host01). Click **Shutdown** > **Force Shutdown Now**. Then click **Yes**. Wait for a little while and refresh the web browser. Ensure that the State of the servers is SHUTDOWN.
   d. Now, select the check box in front of the admin server. Click **Shutdown** > **Force Shutdown Now**. Then click **Yes**.

   Note: Notice the message that you no longer have an administration console, because the admin server is shutting down.
6. Find the Node Manager running on host01 and stop it.

   Note: You want to stop Node Manager before removing its directory.
   a. Find the Terminal window on host01 that is running Node Manager. Stop it by pressing **Ctrl + C** in the window. Do not close this Terminal window, you will use it later.
7. Simulate a "media failure" by renaming the domain directory. (You could delete it, but we will rename it, just in case the backup does not work.)
   a. Open a new Terminal window on host01. Rename the domain directory.

   Note: The command is shown on two lines due to space limitations. Enter it on one line.

```
$ mv /u01/domains/part1/wlsadmin
                                  /u01/domains/part1/wlsadminorig
```

8. Before restoring the domain on host01 and starting the admin server and some managed servers, notice that the servers on host02 (machine2) continue to run.
   a. Return to the web browser. Use it to access an application running on the two remaining cluster2 servers, which are both running on machine2. You will access them through the OHS cluster proxy. Depending upon which practices you have completed, you may have the contacts, supplies, or timeoff applications to try. Their URLs are:

   http://host01.example.com:7777/contacts

   http://host01.example.com:7777/supplies

   http://host01.example.com:7777/timeoff
9. Restore the domain on host01 from the backup.

   Note: You could alternatively use the Weblogic pack and unpack commands to back up your domain. The entire domain must be shut down prior to using pack and unpack.
   a. In the Terminal window on host01, navigate to the directory that contained the domain directory.

```
$ cd /u01/domains/part1
```

   b. Restore the domain directory.

```
$ tar -zxpvf /home/oracle/wlsadminbackup.tarz
```

   Note: The options are:
   - `z`: Unzip the archive

- x: Extract the archive
- p: Preserve file and directory permissions
- v: Verbose output
- f: File name

10. Restart the Node Manager on host01.
    a. Find the Terminal window on host01 in which Node Manager was running. Use the up arrow to retrieve the last command entered. Start Node Manager again by entering the command:
    ```
    $ ./startNodeManager.sh.
    ```
11. Start the admin server.
    a. Find the Terminal window in which the admin server was running. Use the up arrow to retrieve the last command entered. Start the admin server by entering that command:
    ```
    $ ./startWebLogic.sh
    ```
    Note: If the window has been closed, open a new Terminal window, navigate to the domain directory and start the admin server.
    ```
    $ cd /u01/domains/part1/wlsadmin
    $ ./startWebLogic.sh
    ```
    Note: Wait for the server to show it is in RUNNING mode.
12. Open the WebLogic administration console.
13. Start the two managed servers in cluster2 that run on machine1 (and that use the domain directories just restored).
    a. In the Domain Structure, expand **Environment**, and select **Servers**.
    b. Click the **Control** tab.
    c. Select the check box in front of the two managed servers in cluster2 that are on `machine1` (which is host01).

    Note: The servers are `cluster2server-1` and `cluster2server-3`.
    d. Click **Start**. Then click **Yes**.
    e. Wait for a little while and refresh the web browser. Ensure the State of the servers is RUNNING.

    Note: You have restored the “missing” domain files from a backup and restarted the servers running on that machine.

Practice Solution: Backing Up and Restoring a Domain

There is no solution to this practice.

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

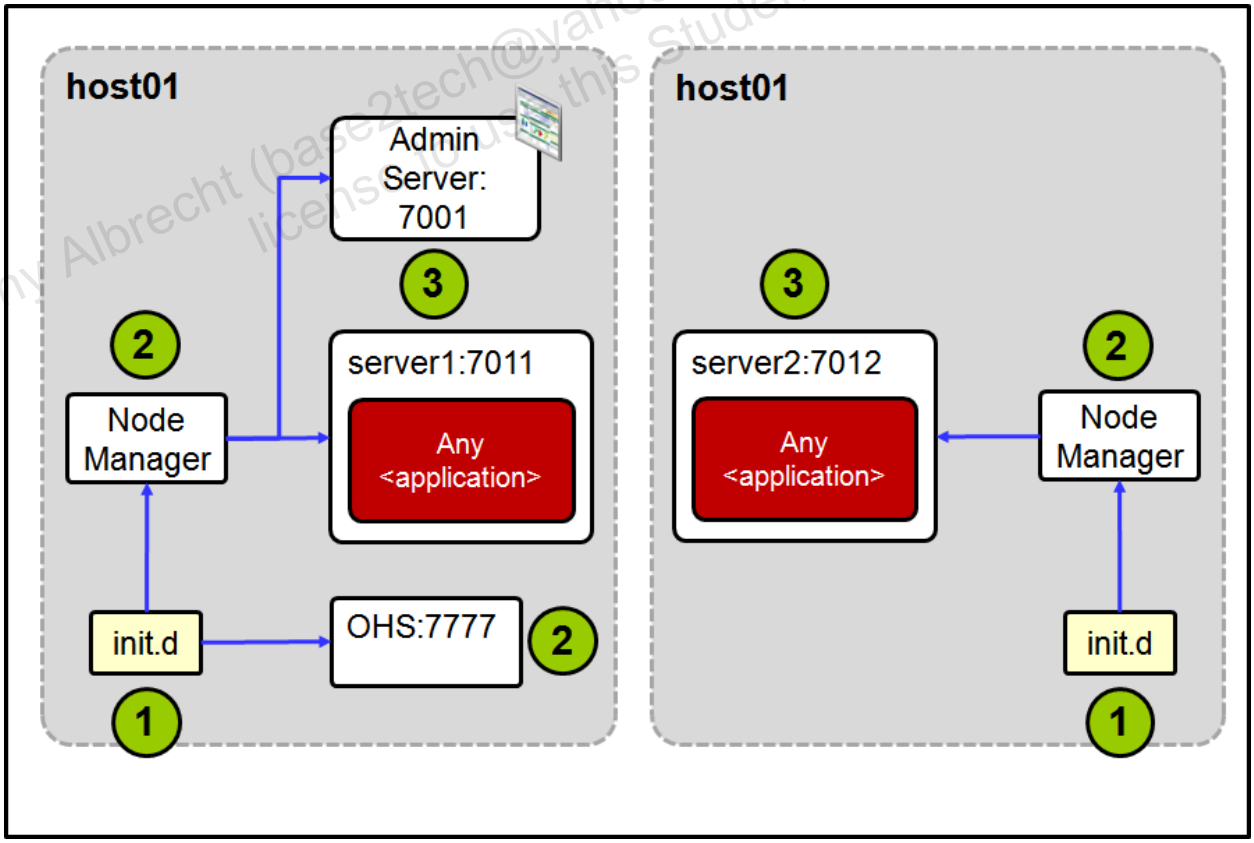
By default, the Node Manager processes responsible for the servers in a domain automatically restart any servers that they started if they shut down ungracefully. There is no mechanism in place by default to do the same for restarting the Node Manager processes in the event that they crash or the machine 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
   ```

   Notes: 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. Enter **`oracle`** and answer **`yes`** to any security questions as appropriate.

    ```
    $ 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

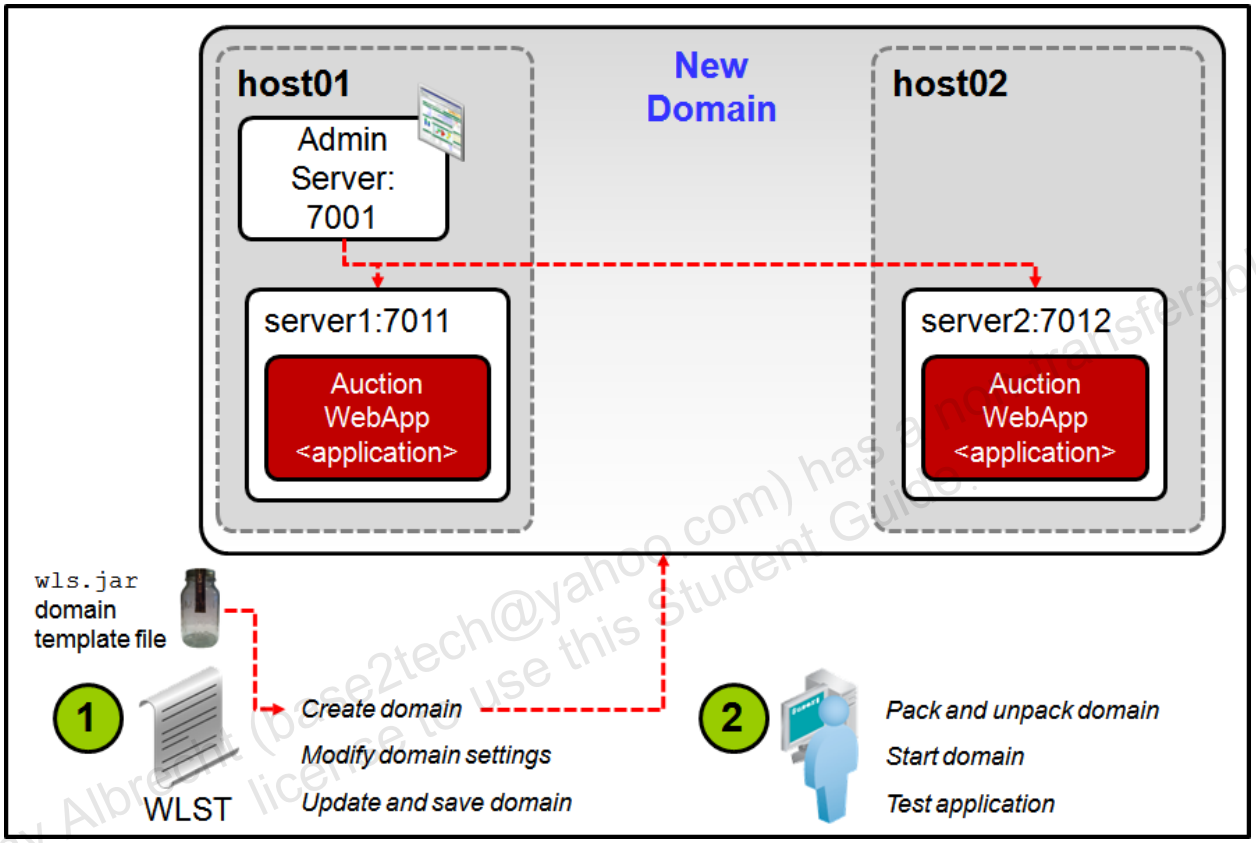
Practices Overview

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:

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

And add the following line below it:

```
cmo.setPassword('Welcome1')
```

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, remember 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 with `weblogic` and `Welcome1`.

```
$ ./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 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`	`weblogic`	The administrative user of the domain
`password`	`Welcome1`	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 that monitors server statistics:

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

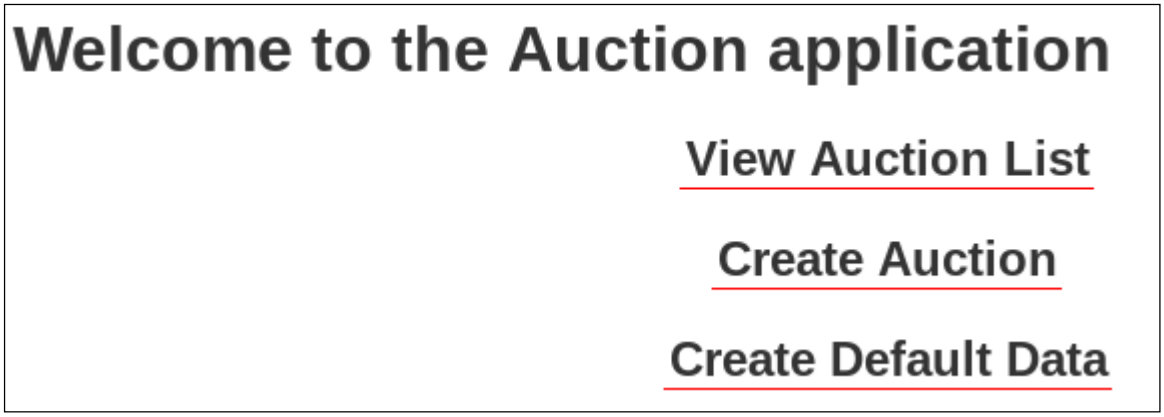
After a few seconds, the screen should clear and display the statistics it captures from the domain. If you have not run the application yet, your display should have zero values for everything. It is ok if there are some requests too. The script sleeps for 15 seconds, clears the screen, and then retrieves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



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 13: Working with the Security Realm

Chapter 13

Practices for Lesson 13: Overview

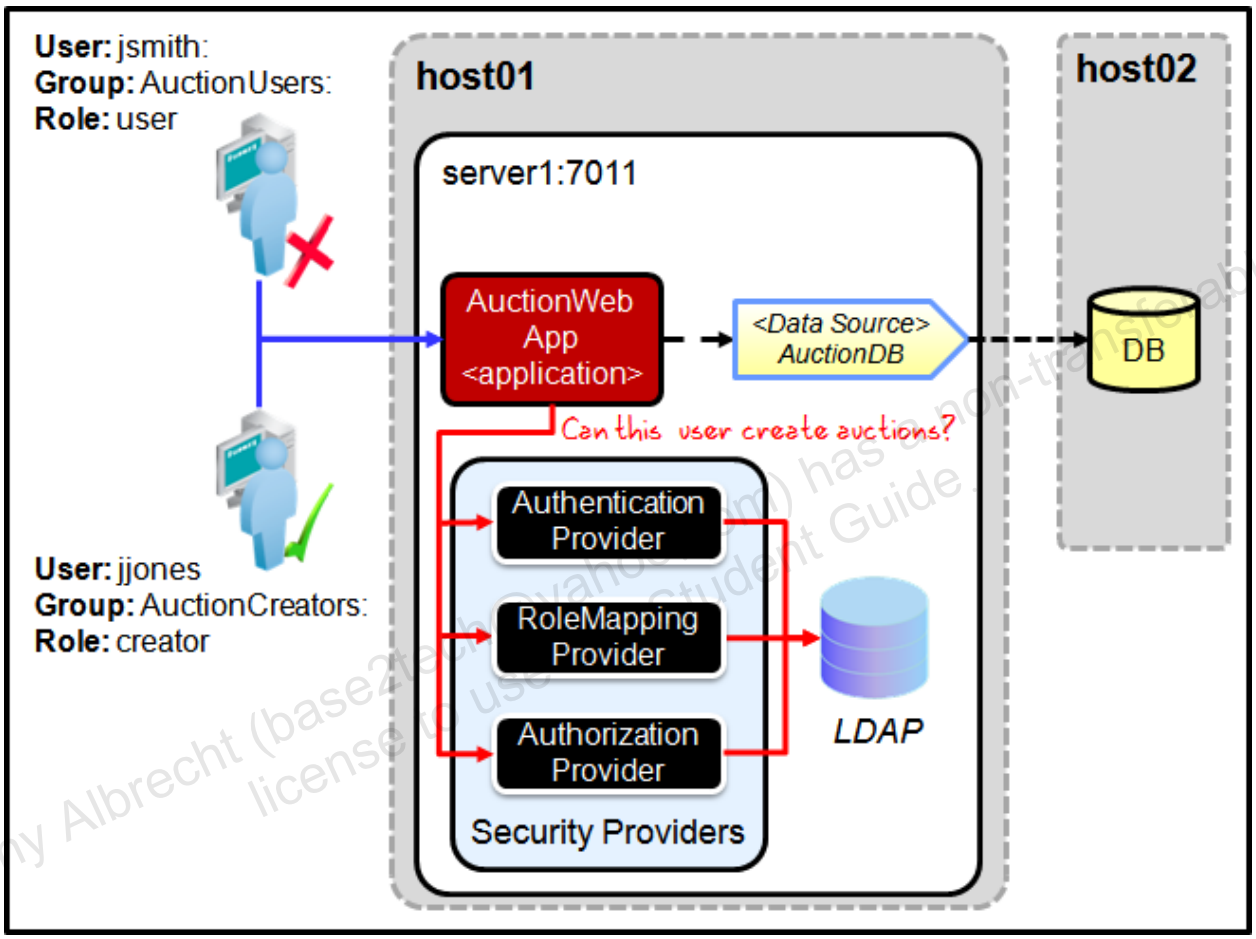
Practices Overview

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 13-1: 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 `practice13-01` folder.

      ```
      $ cd /practices/part2/practice13-01
      ```

   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 `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.

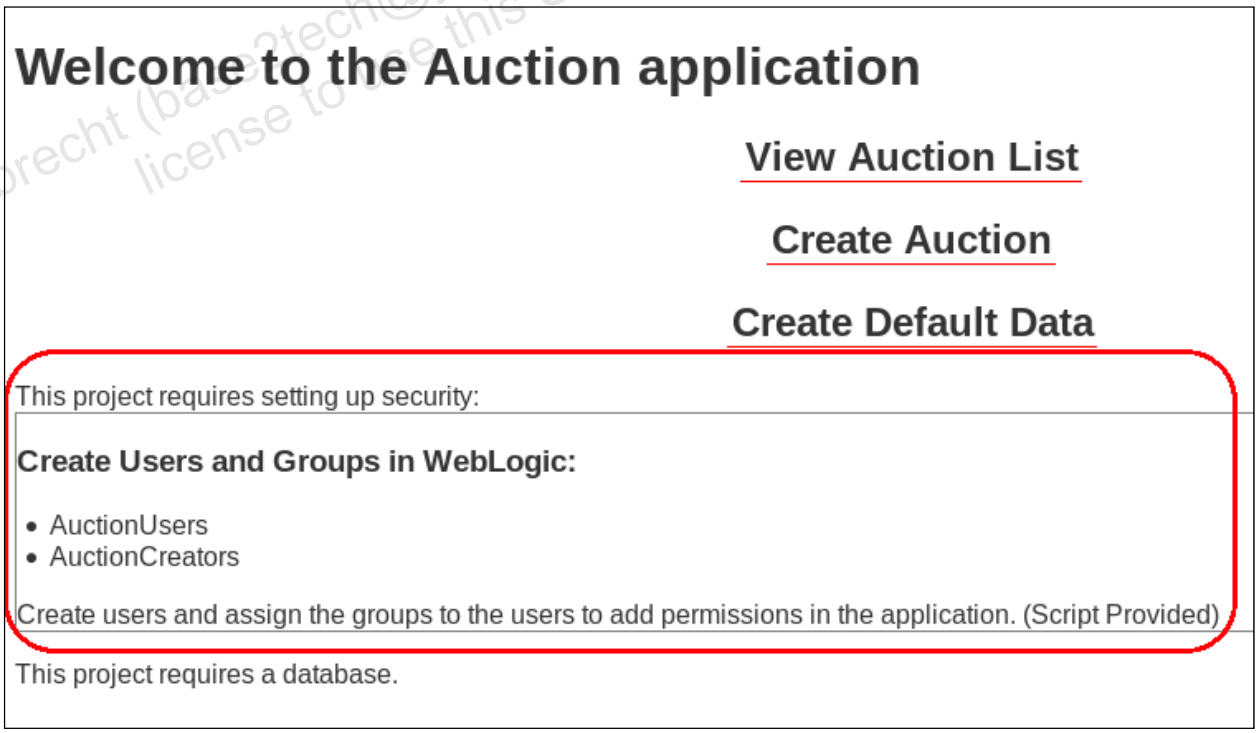
6. Test the unprotected application.

   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.

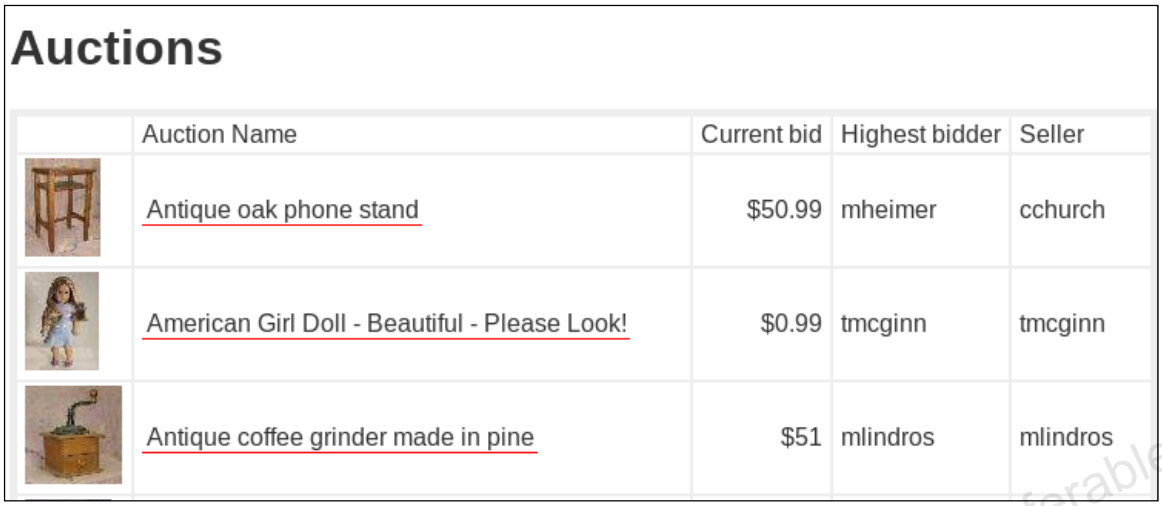
   a. Direct your web browser to the following URL:

      `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.

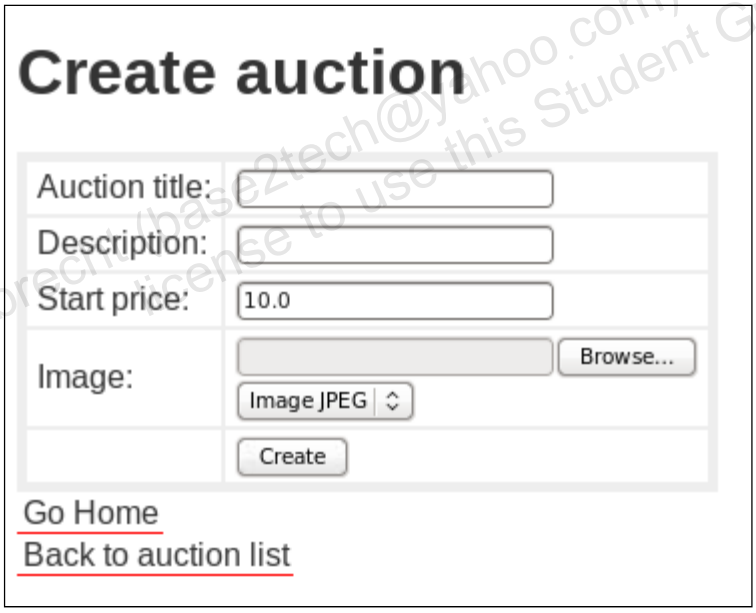


b. Click the *View Auction List* link to view the list of auctions stored in the database.



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.



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.

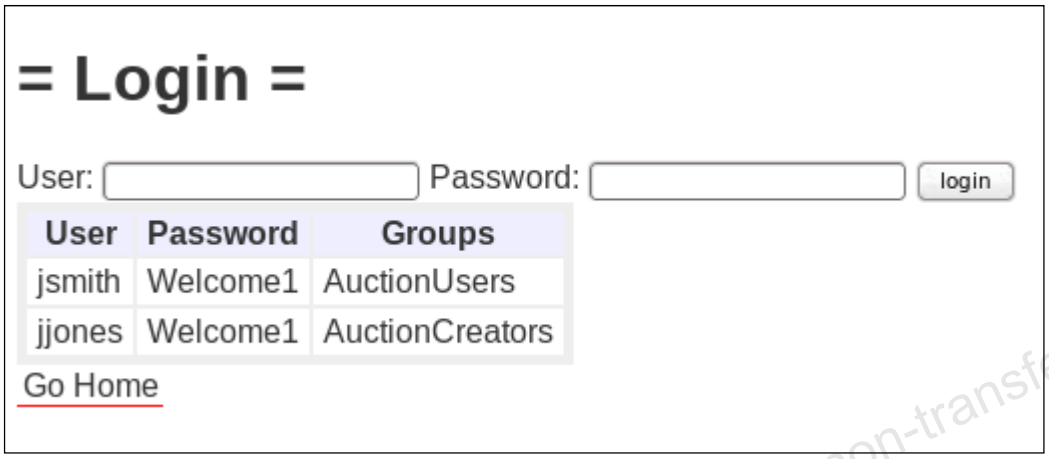
   c. Repeat the previous step to create the creator role and map it to the AuctionCreators group. Return to the Roles page using the breadcrumb trail at the top of the page by selecting Roles.

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 Tools > Start Private Browsing.
    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.



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.

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
   ```

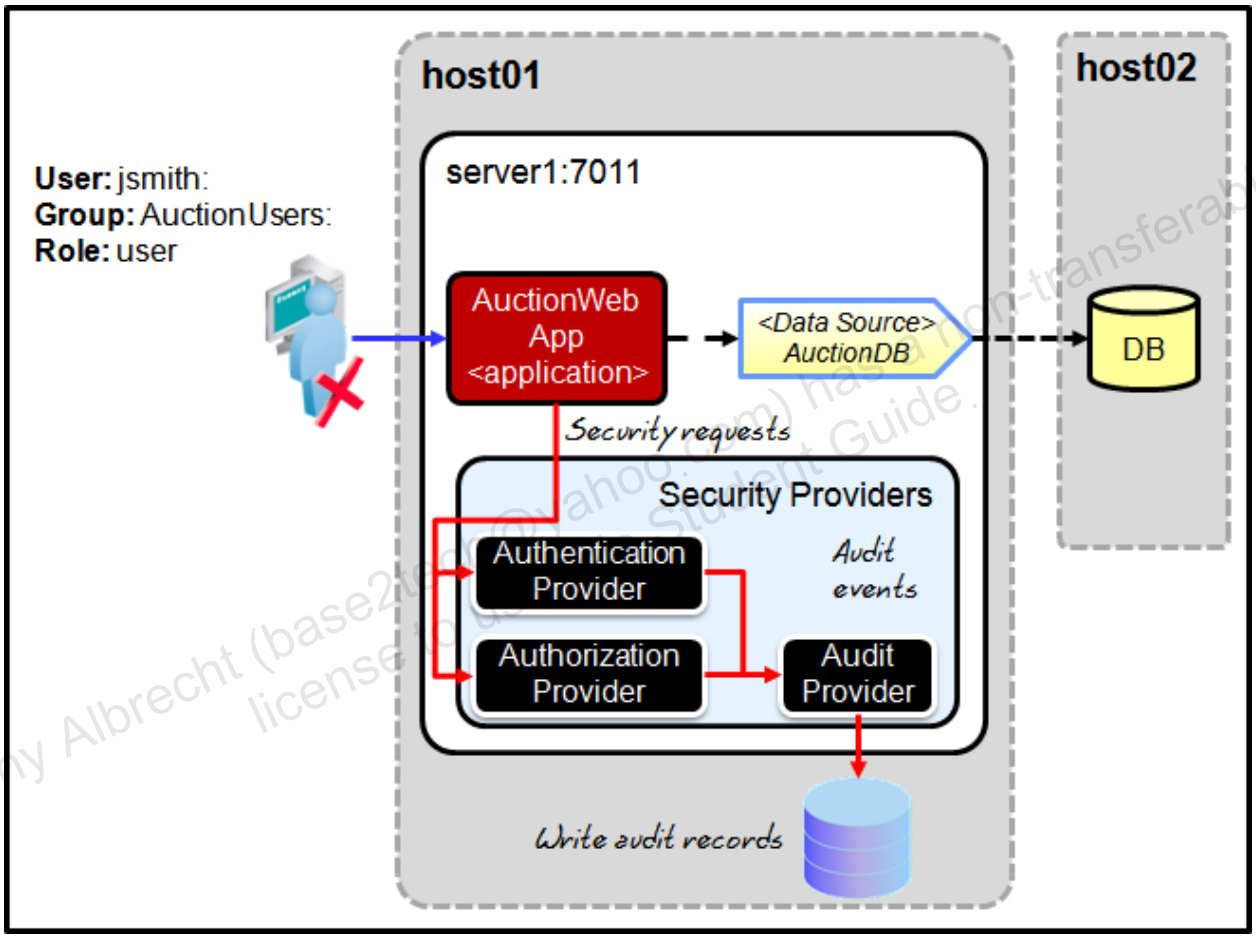
   The solution script performs the following:
   - Ensures that no previous servers are running on both machines
   - Restores the domain and practice to their original state
   - Starts the wlsadmin domain on host01
   - Deploys the solution applications used for this practice
   - Creates all the security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

Practice 13-2: 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 `practice13-02` folder.

      ```
      $ cd /practices/part2/practice13-02
      ```

   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 practice13-01, 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 15: Diagnostic Framework

Chapter 15

Practices for Lesson 15: Overview

Practices Overview

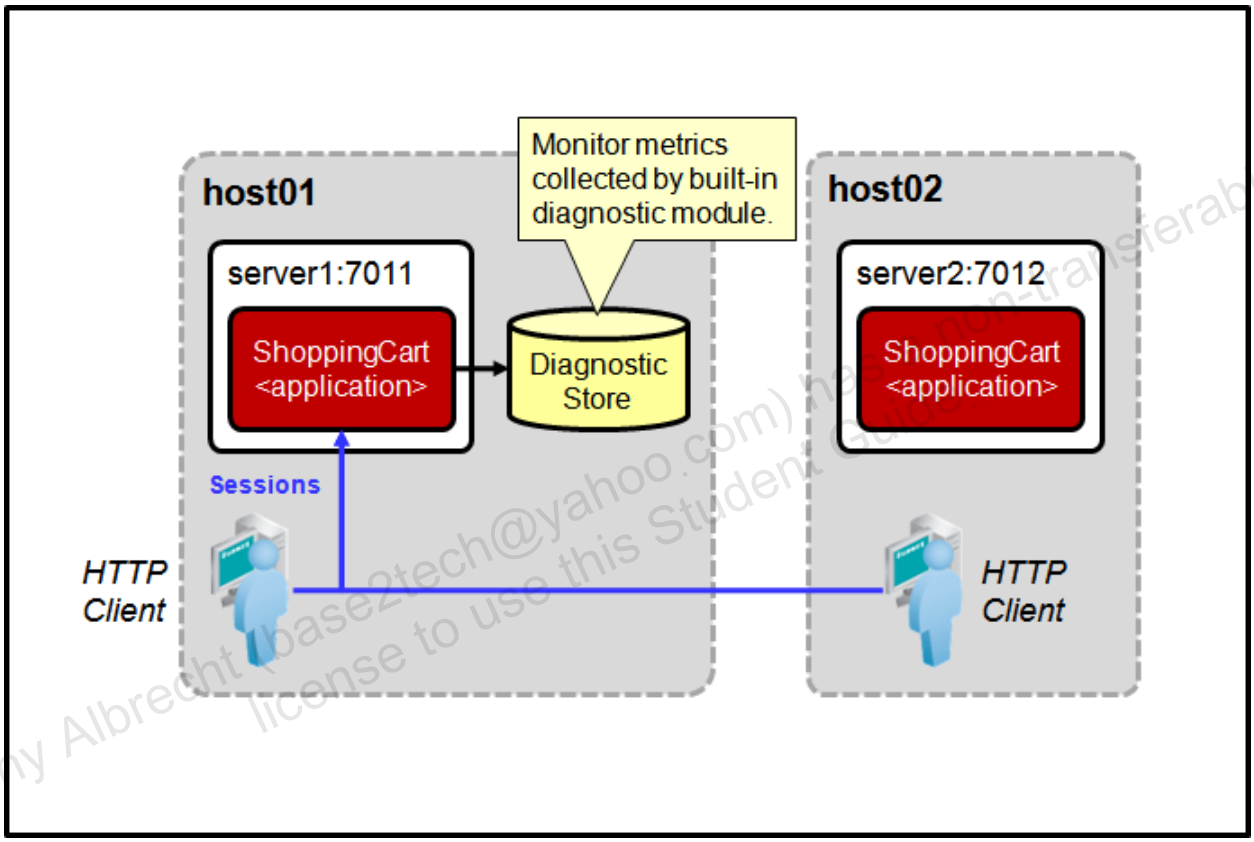
In the practices for this lesson, you use the administration console to configure WebLogic built-in diagnostic modules and metrics collected by them.

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:



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
      ```

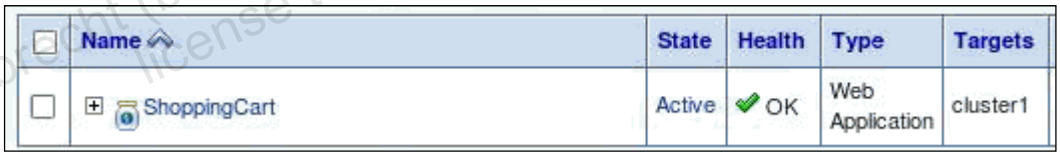
   d. Execute the `setup.sh` script to set up the initial practice environment:

      ```
      $ ./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`

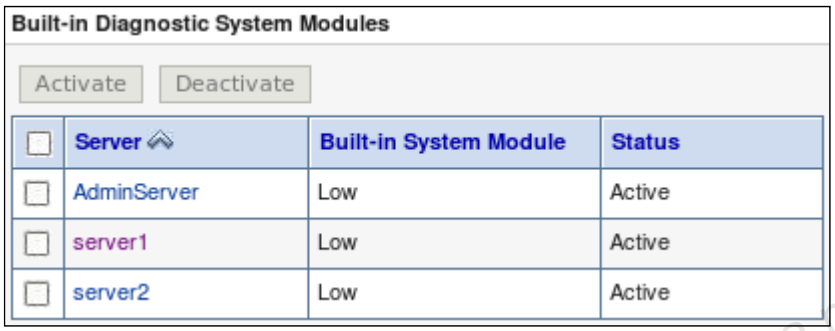
   b. Verify that the `ShoppingCart` application is deployed.



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.

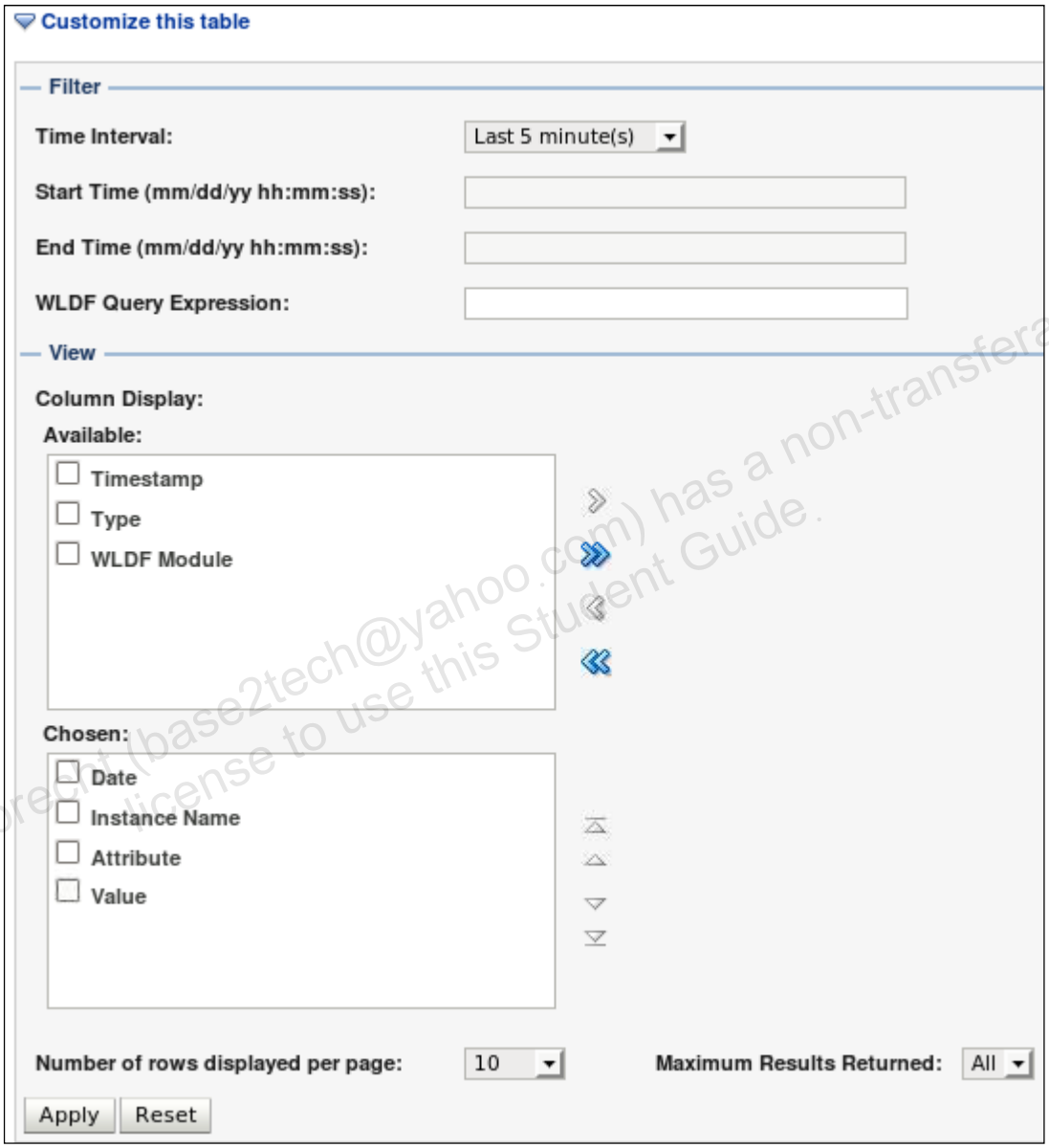


6. Configure a new setting for a server’s built-in module.
   a. All of the default built-in modules are set to Low. WebLogic allows for Low, Medium, or High settings for each module. Use the console to set the built-in module for `server1` from Low to Medium.
   b. Click `server1` to show the settings for its built-in module.
   c. Click Lock & Edit.
   d. Change its setting from Low to Medium.

   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 panel, 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.



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 panel, 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. Run the application to create more metric data.

o. Return to the page 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

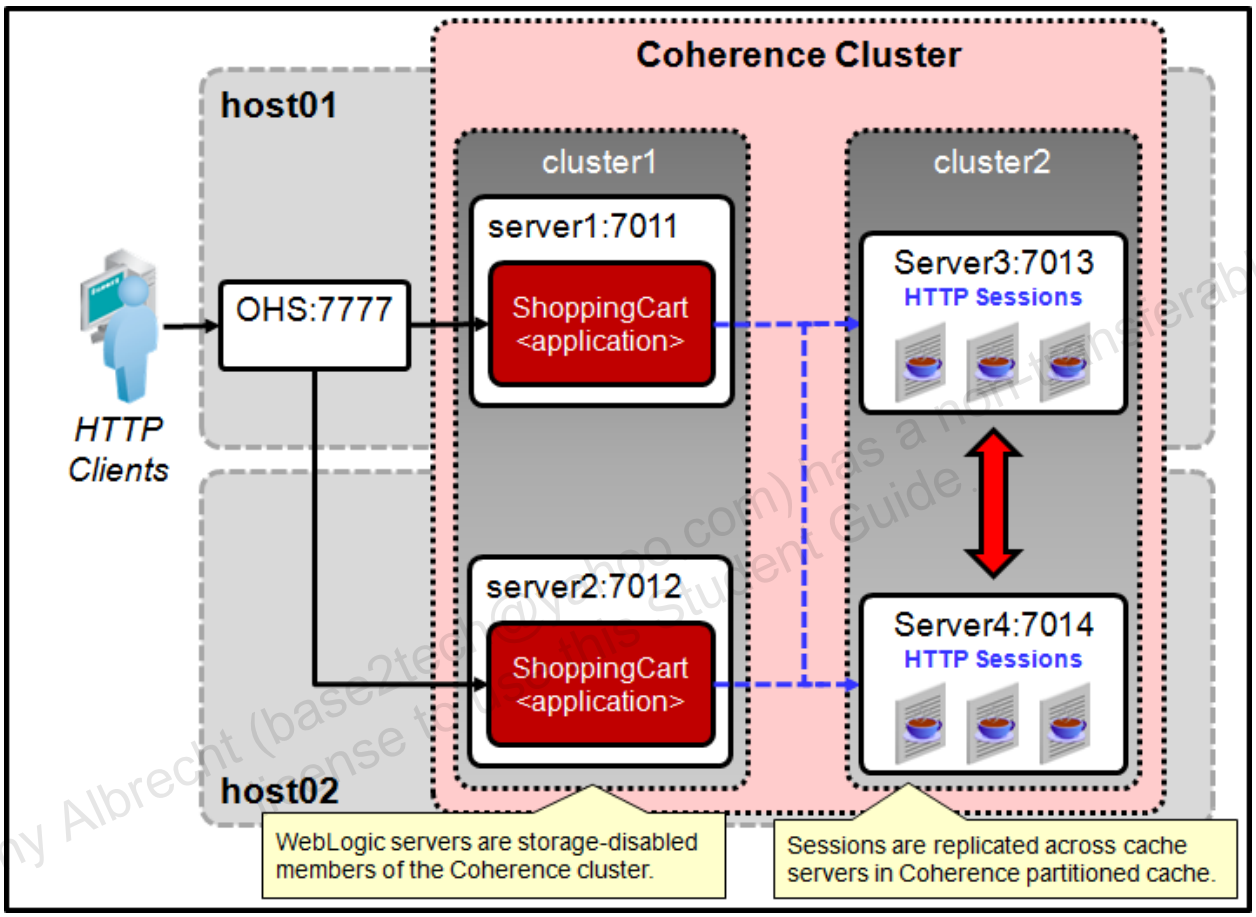
Practices Overview

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.



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
      - 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. Coherence libraries are automatically added to the WebLogic system classpath when you install WebLogic. In the `$WL_HOME/modules/features/weblogic.server.merge.modules_12.1.3.0.jar` file, the MANIFEST classpath includes both the `coherence.jar` and `coherence-web.jar` files in the classpath. These JAR files are located in the `$MW_HOME/coherence/lib` folder. Because these files are automatically included in the WebLogic system classpath, no further configuration is required for Coherence*Web to work unless you want to change something specific about the configuration.

b. You do not have to do this part, but 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.

c. 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 panel, 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 Panel, 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**:

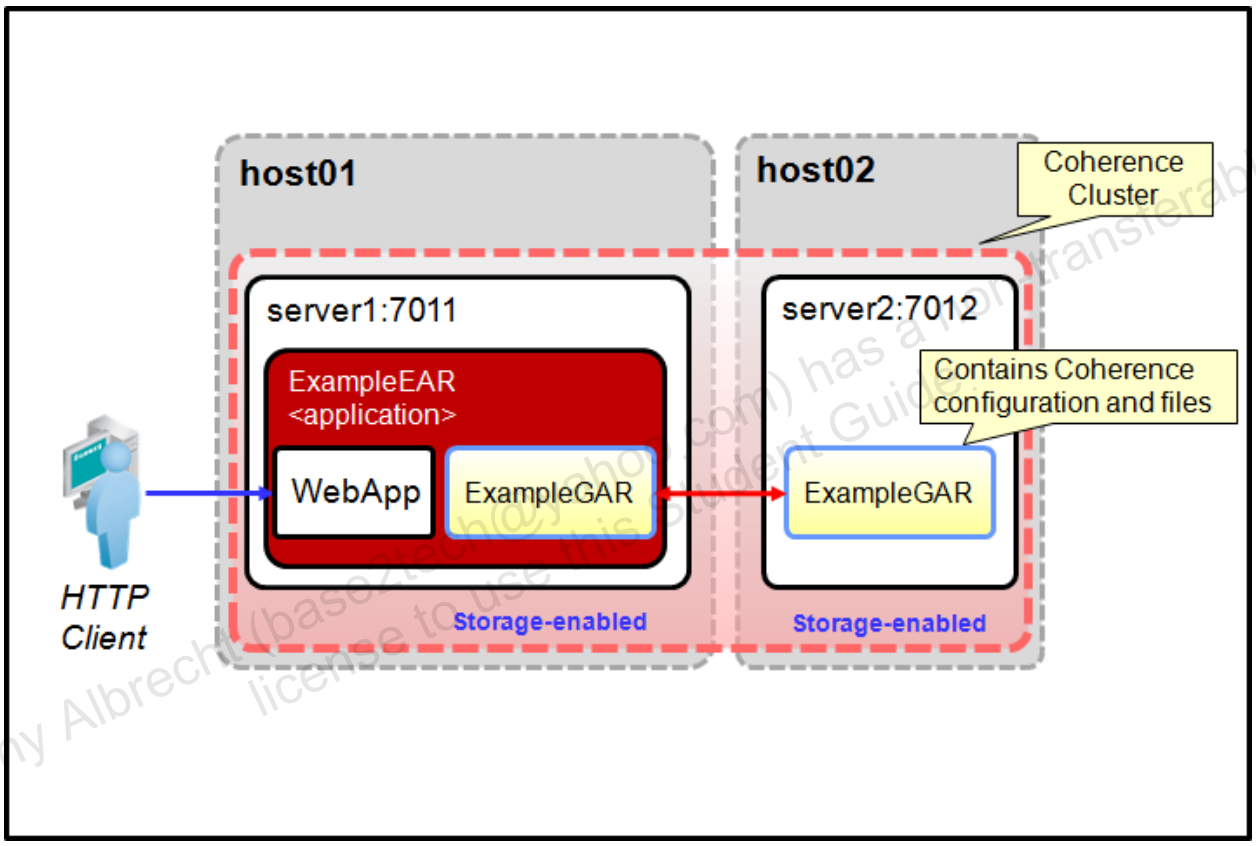
   ```
   $ ./solution.sh
   ```

   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
   - 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
      ```

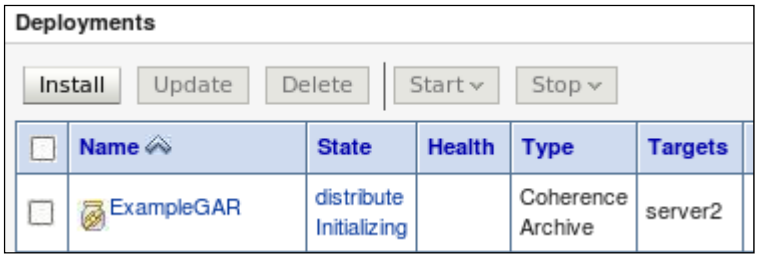
      This script performs the following:
      - Ensures that no previous servers are running on both the machines
      - Restores the domain and practice to their original state
      - Starts the `AdminServer` on host01
      - Ensures that no applications or libraries are deployed
3. Create Coherence 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.
   h. Click Save.
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.



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.

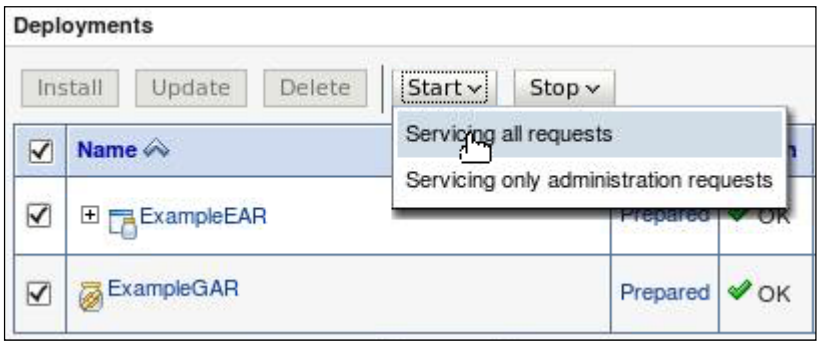
c. Click the Control tab.

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.



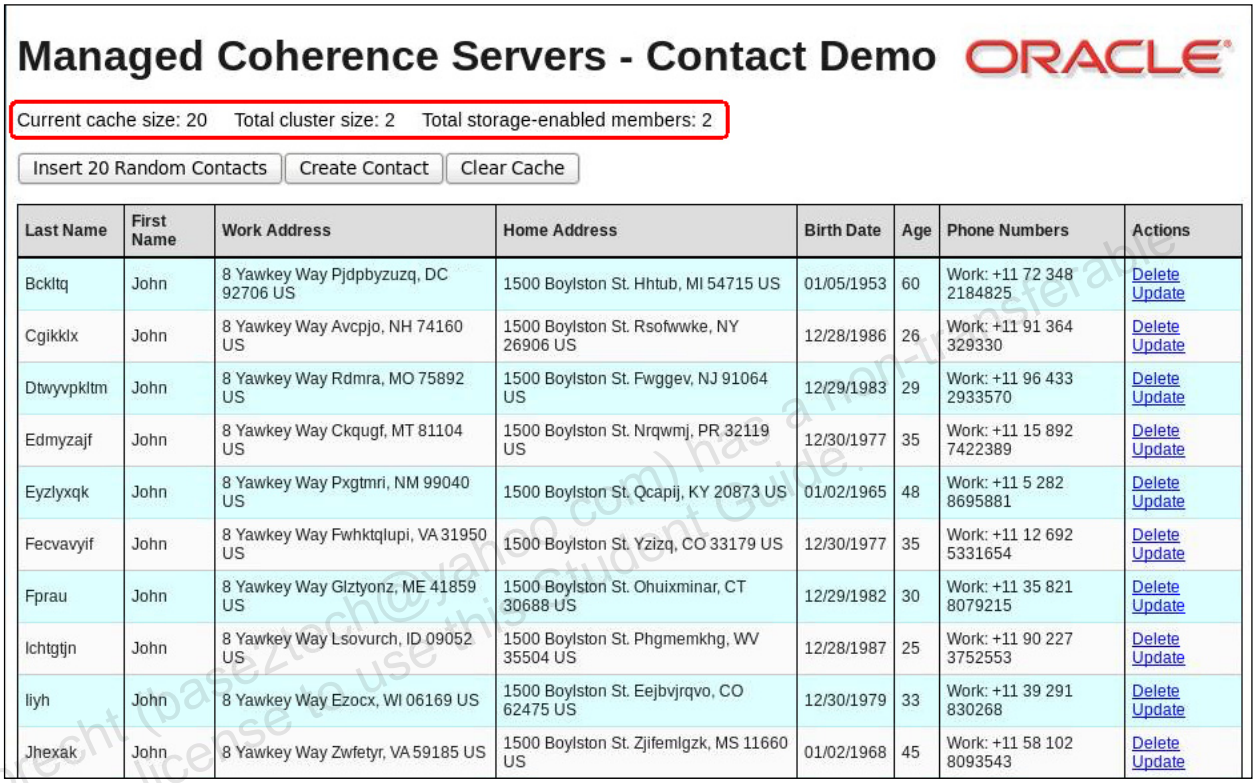
7. Test the managed Coherence server application.

   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 ORACLE

Current cache size: 20 Total cluster size: 2 Total storage-enabled members: 2

Insert 20 Random Contacts | Create Contact | Clear Cache

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 4: Tuning HotSpot JVM

Chapter 4

Practices for Lesson 4: Overview

Practices Overview

In the practices for this lesson, you monitor and tune parameters associated with the Java HotSpot JVM. First, you tune Java garbage collection, load test an application, and monitor garbage collection and application performance to perceive differences with different collectors. Next, you use Java command-line tools to monitor and manage JVM processes. Then, you use various graphical tools to monitor and manage JVM processes.

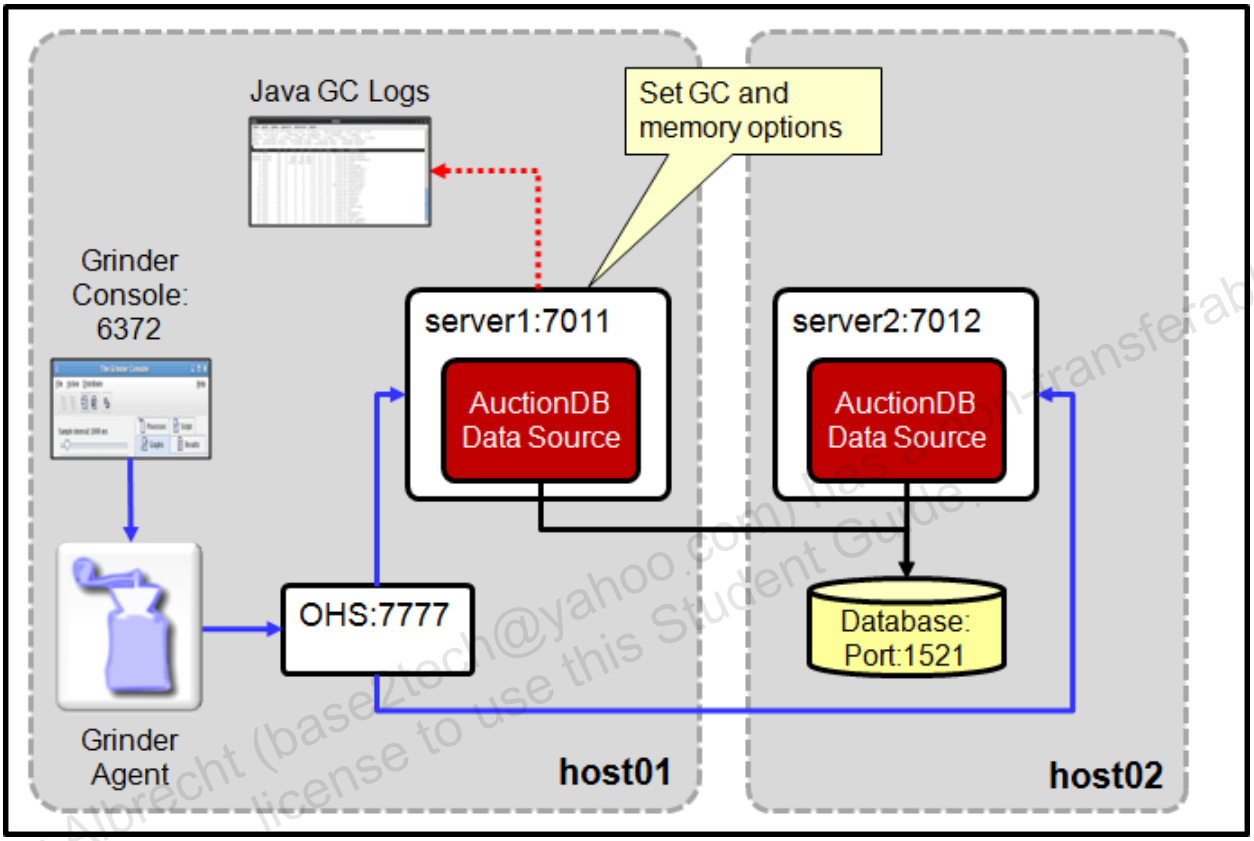
This lesson takes you on a journey of exploring the different tools available to you for diagnosing JVM performance issues. You may not be able to answer all the questions at first with the tools provided at the time. This lesson starts you off at the lowest common level, and then expands into more powerful tools that enable you to more easily identify GC and JVM performance issues and solutions.

Practice 4-1: Tuning JVM Garbage Collection

Overview

In this practice, you experiment with configuring different garbage collectors for an application, loading the application, and monitoring the performance results for comparison.

The following image depicts the architecture of the domain used for this practice.



About This Course's Approach

This practice provides only guidelines for how to accomplish the objectives of the lesson. You take lessons learned from the slides or previous practices and apply them for this practice on your own.

Dependencies

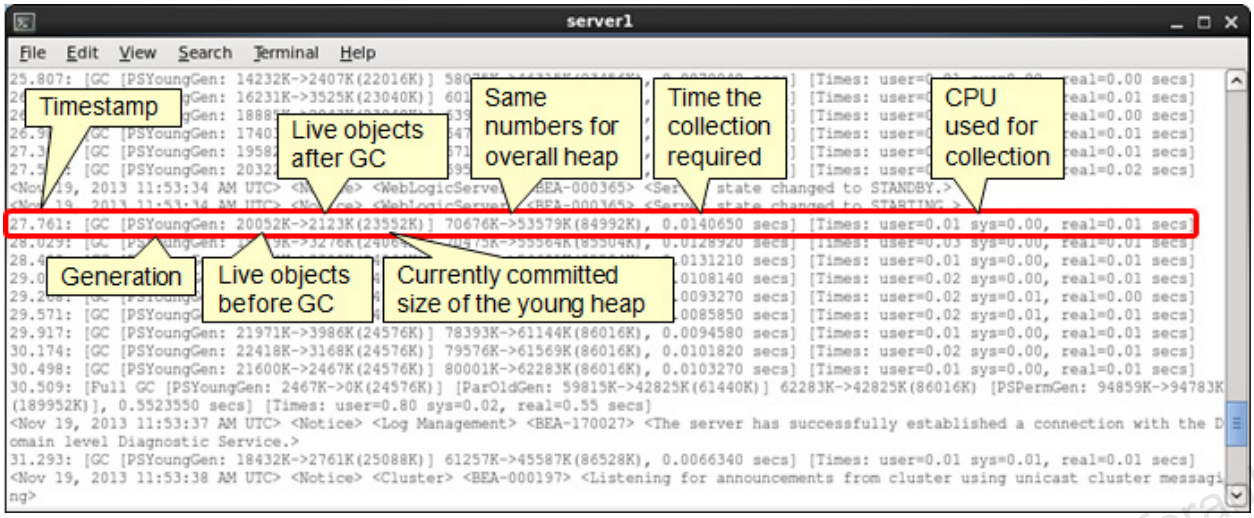
This practice depends on:

- Practice 2-1 for setting up the course environment.
- Practice 2-3 for configuring Grinder and establishing initial baseline performance numbers of the Auction application in this environment.

Reading GC Log Output Hints

Use these hints to help you read and analyze GC log data. Extrapolate these hints to all forms of collectors used in these practices.

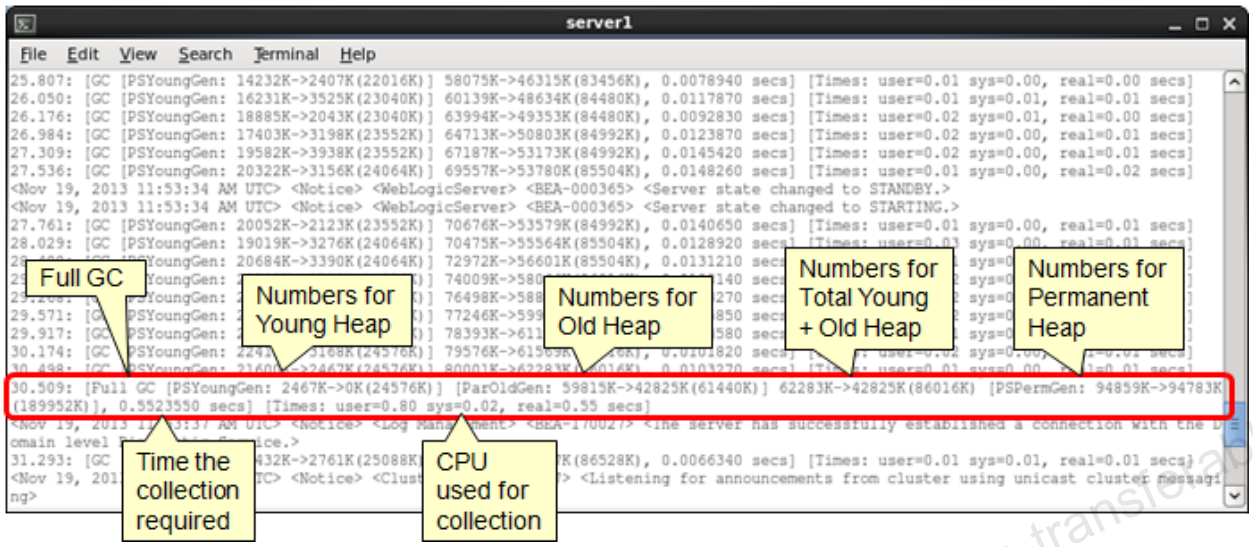
Default Minor GC on WebLogic



This screenshot shows a minor garbage collection log, which includes the following:

- A timestamp of when collection occurred.
- A minor collection is indicated by the term '`GC`' in the output.
- The collection occurred in the young generation as indicated by the term '`PSYoungGen`'.
- The collection consisted of:
  - A parallel collection using multiple threads, as indicated by the `P` in `PSYoungGen`.
  - An initial 20052KB of combined live objects before collection
  - An ending 2123KB of combined live objects after collection
- The total size of the young heap is 23552KB.
- The total size of the young and old heap combined before collection was 70676KB.
- The total size of the young and old heap combined after collection was 53579KB.
- The total committed size of the young and old heap combined is 84992KB.
- The collection took 0.0140650 seconds, or 14 milliseconds.
- The collection used approximately 0.02 seconds of CPU time.

Default Full GC on WebLogic



This screenshot shows a full garbage collection log, which includes the following:

- A timestamp of when collection occurred.
- A major collection is indicated by the term '`Full GC`' in the output.
- The collection occurred in the young generation as indicated by the term '`PSYoungGen`'.
- The collection occurred in the old generation as indicated by the term '`ParOldGen`'.
- The collection occurred in the permanent generation as indicated by the term '`PSPermGen`'.
- The collection consisted of:
  - A parallel collection of the young generation using multiple threads, as indicated by the `P` in `PSYoungGen`.
  - A parallel collection of the old generation using multiple threads, as indicated by the `Par` in `ParOldGen`.
  - A parallel collection of the permanent generation using multiple threads, as indicated by the `P` in `PSPermGen`.
- The collection took 0.5523550 seconds, or 552 milliseconds.
- The collection used approximately 1.37 seconds of CPU time.

Tasks

1. Connect to the **host01** and **host02** machines.
2. Set up the practice environment.
   a. Perform the following commands on **host01** to set up the environment for this practice:
      ```
      $ cd /practices/tune/practice04-01
      $ ./setup.sh
      ```
      This script does the following:
      - Restores the domain to its original state
      - Starts the wlsadmin AdminServer
      - Deploys the SimpleAuctionWebAppDb application to the cluster
      - Starts the server1 managed server on host01
      - Starts the server2 managed server on host02

   b. Wait for all servers to display that they are in the RUNNING state before continuing.
3. Explore the existing default memory and garbage collection settings for WebLogic managed servers.
   a. Navigate to the domain's `bin` folder on **host01** and open the domain's environment script.
      ```
      $ cd /u01/domains/tune/wlsadmin/bin
      $ gedit setDomainEnv.sh &
      ```

b. Explore the `setDomainEnv.sh` file and answer the following questions:

Hint: If you are having trouble figuring out the settings in the script file, you can review each managed server's console output for its start up command-line arguments and memory settings, and work your way backwards in the script file to see where the values are set. Some of the questions you may not be able to answer by looking only at the script. You will need to change some start up settings to get the answers. See the `resources/setUserOverrides.sh` script for possible settings. You can also review lecture slides 4-14 through 4-17, which explain memory settings for Java heaps.

It is OK if you cannot answer every question.

Question	Answer (include why)
What is the default minimum heap size?	
What is the default maximum heap size?	
What is the default permanent generation size?	
What is the default maximum permanent generation size?	
What are the default garbage collector settings? Include information for all generations.	
Do the servers start up in client or server JVM mode?	
What is the `setUserOverrides.sh` script?	This is a trick question because if you are familiar with WebLogic at all, then you have never heard of this file before WebLogic 12.1.2. This is a new file you can create in your `DOMAIN_HOME` folder that is called by the `setDomainEnv.sh` script to allow you to override and add settings to start your domain instead of modifying the other WebLogic scripts directly. **So keep this in mind when you are changing settings for this practice!**

4. Configure GC monitoring.
   a. Shut down server1 so that you can modify some settings.
   b. Create the `setUserOverrides.sh` file in your `DOMAIN_HOME/bin` folder. Be sure to place any environment variable settings in this file. There is a starter file in the practice `resources` folder if you prefer to not create the file yourself. Remember that you must create this file for each host if you want to monitor both managed servers because the domain folder is not a shared file system. However, for this practice, using the file for server2 is not recommended because it will take too long. There is a lot of work to do for this course and the goals are accomplished by focusing on one server. You will modify this same file, directly in the `DOMAIN_HOME/bin` folder for the remaining practices in this lesson.
   c. Experiment by setting the different settings in the table below and starting the servers with the new settings. Remember to start the servers using the `startAdmin.sh`, `startServer1.sh`, and `startServer2.sh` scripts. Try answering the questions in step 3b again.

Setting	Setting
`-XX:+PrintCommandLineFlags`	`-XX:+PrintGC`
`-XX:+PrintGCDetails`	`-XX:+PrintGCTimeStamps`
`verbose:gc`	

   d. If you do not have Grinder running, then start the Console and Agent now. For convenience, a script has been created to make it easy to start Grinder by using a single command. Execute the following steps to run Grinder:
      – Open a new terminal window on **`host01`**.
      – Execute the following commands to run Grinder:

```
$ cd /practices/tune/practice02-03
$ runGrinder.sh
```

      The script opens two terminal windows and automatically starts the Console and the Agent processes for you. Now all you must do is use the Console to control your tests.

      IMPORTANT! This script causes Grinder to reuse your recorded load test and `grinder.properties` settings from `practice02-03`. You can continue to use these files when using Grinder for other practices, unless a practice specifies different instructions. If you do not have these files available, then run Grinder from within the `practice02-03/solution` folder and use the files in that location instead.
   e. Keep the garbage collection monitoring settings in place and run your Grinder load again. Make sure that both managed servers are running for your tests. Record the performance numbers for this test to ensure that garbage collection monitoring has not impacted your baseline numbers. Keep these performance metrics available for this practice to compare with other load test metrics.

f. Find the garbage collection output in the server console window and examine the garbage collection metrics that were recorded during your load test. Use those metrics to answer the following questions:

Hint: Use the garbage collection examples in the beginning of this document to understand garbage collection output. You can also refer to lecture slides 4-20 through 4-21 to help answer these questions.

It is ok if you cannot answer every question.

Question	Answer (include why)
Pause Time: Does the collector introduce pauses that stop the world?	
Memory Footprint: What is the total memory allocated versus the total memory available to the application?	
Parallel: Does the garbage collector make use of multiple CPUs?	
CPU Usage: What is the percentage of CPU spent in garbage collection?	
Concurrency: Is there a contention for a CPU between application threads and the garbage collector? If so, can this be overcome by use of a system with more CPUs?	

g. What you should have noticed is that WebLogic and the application run well using the default memory settings. Garbage collection occurs very infrequently, and should only be minor collections. This is denoted by the log entries that start with “GC.” Full garbage collections start with “Full GC” and are much slower.

h. Because the application is running well with the default memory settings, you must lower the heap size of the server to force a performance issue with garbage collection. Try setting the heap size and maximum heap size to 90 MB and run Grinder again to see how the application performs. Remember that you must also set the maximum permanent generation size when making custom memory settings in your `setUserOverrides.sh` script. Try to answer the following questions:

Hint: There are no concrete answers to these questions because every student is literally running a different experiment based on whatever test they recorded with Grinder. Use the garbage collection examples in the beginning of this document to understand garbage collection output. You can also review the *Course Developer Corner* section at the end of this practice for more details about what was encountered during the development of this practice.

Question	Answer (include why)
Does the application perform faster or slower than before? If so, why do you think that is?	
Do you notice anything different about the application's garbage collection process? Is it using more CPU? Is it affecting overall application performance?	
How far apart are garbage collections? Minor GCs? Full GCs?	

Note: You may notice that when you press Ctrl + C to restart a server, the JVM displays more garbage collection information, much like the following example when the serial collector was used:

```
Heap
 def new generation   total 15360K, used 12654K
[0x00000000ece00000, 0x00000000edea0000, 0x00000000edea0000)
  eden space 13696K,  92% used [0x00000000ece00000,
0x00000000eda5b880, 0x00000000edb60000)
  from space 1664K,   0% used [0x00000000edd00000,
0x00000000edd00000, 0x00000000edea0000)
  to   space 1664K,   0% used [0x00000000edb60000,
0x00000000edb60000, 0x00000000edd00000)
 tenured generation   total 34176K, used 34176K
[0x00000000edea0000, 0x00000000f0000000, 0x00000000f0000000)
   the space 34176K, 100% used [0x00000000edea0000,
0x00000000f0000000, 0x00000000f0000000, 0x00000000f0000000)
 compacting perm gen  total 101120K, used 101045K
[0x00000000f0000000, 0x00000000f62c0000, 0x0000000100000000)
   the space 101120K,  99% used [0x00000000f0000000,
0x00000000f62ad4d8, 0x00000000f62ad600, 0x00000000f62c0000)
No shared spaces configured.
```

i. Set the server's heap size to 50 MB and run the test again. Answer the questions again for this heap size.

j. Change the garbage collection algorithm for server1 and server2 to two of the settings in the following table (one at a time), restart the managed servers for each setting, run your Grinder load tests again for each setting, record the performance results, and compare them to the metrics from the last baseline test. Try to answer all of the questions in 4f for each run. Review the *GC Algorithms* slide in this lesson's presentation for details about each setting. There is not enough class time to try to run them all.

Setting	Setting
`-XX:+UseSerialGC`	`-XX:+UseParNewGC` (use with the concurrent collector)
`-XX:+UseParallelOldGC`	`-XX:+UseConMarkSweepGC` (use with the ParNew collector)
	`-XX:+UseG1GC`

5. Shut down the environment.

   a. When you are done with this practice, execute the following command to shut the environment down.

   ```
   $ cleanup.sh
   ```

   b. If any dialog windows appear asking if you want to close a terminal window, click OK to close the window.

   c. You should notice that the AdminServer, server1, and server2 terminal windows and the servers that were running in them are all shut down now. The Grinder Console and Agent applications should still be running.

Course Developer Corner:

You should notice that when you change to a heap size of 50 MB that the garbage collector struggles to keep enough free memory available for the JVM to run. You should also notice more frequent full garbage collections that heavily use the CPU. In some cases, all the tests will complete, and in other cases you may have to stop the test because the JVM is failing to keep up with the load.

I was impressed with the G1 collector when run in a 50 MB heap. Not only was it able to finish all the tests, but it also managed some decent performance numbers even in such a constrained environment. I never even set any threshold parameters for it to follow. Did you notice similar results?

My baseline test for a default 256-512 MB heap and parallel throughput collector settings were:

11.2 ms average response time, 72.5 average / 281 peak TPS

My 50 MB G1 collector test run produced the following numbers, which were very on par with much less memory:

19.0 ms average response time, 75 average / 265 peak TPS

The primary focus of this practice is to get you used to setting and analyzing JVM garbage collection performance of a WebLogic Server by using JVM command-line options. The goal in tuning the garbage collector and Java heap spaces is to minimize the occurrence of full GC events (and minimize the frequency of GC events in general) while maintaining acceptable pause times. For finer details and explanations of each aspect of garbage collection output, see the *D82344GC10 Java Performance Tuning with Mission Control and Flight Recorder* course.

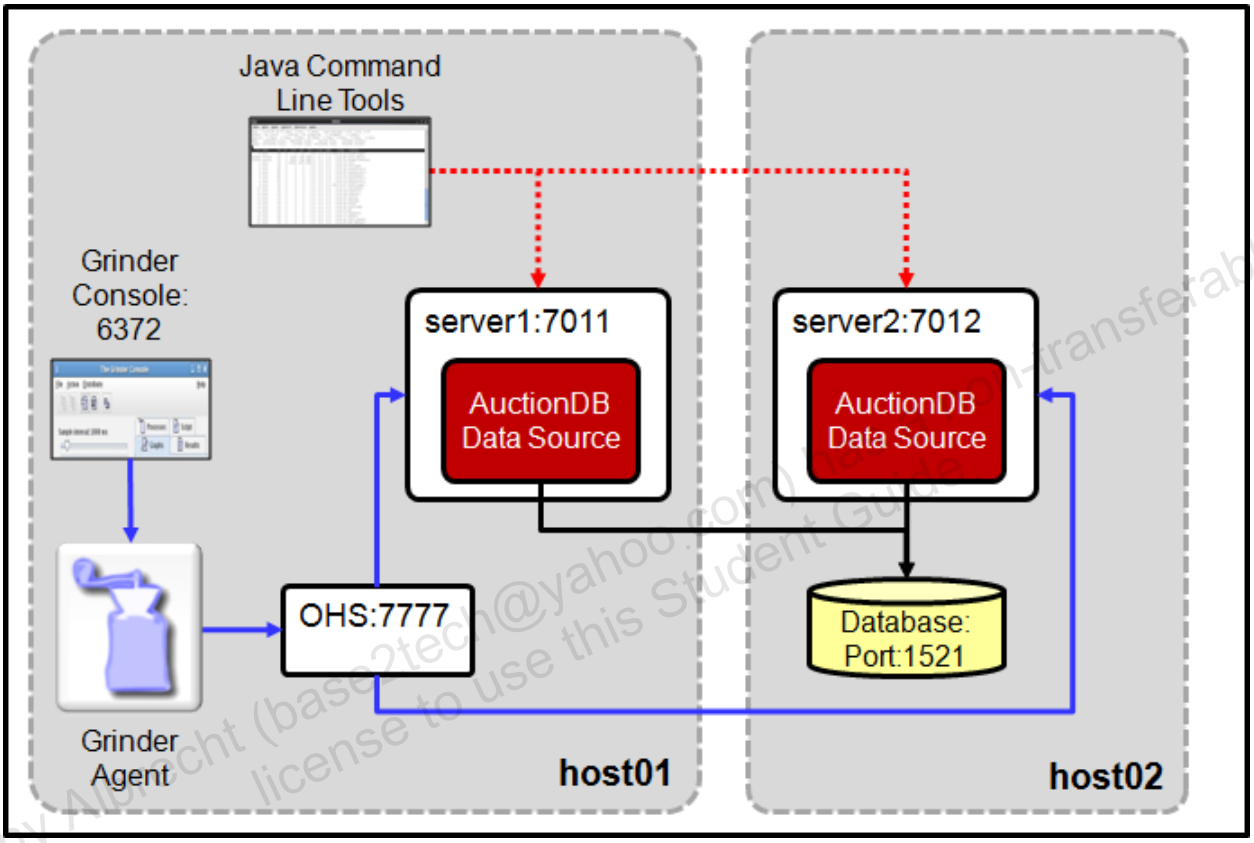
Practice Solution: Tuning JVM Garbage Collection

There is no solution for this practice. No practices depend on this practice. However, the `setUserOverrides.sh` script is introduced in this lesson. This script is used in other practices, but is supplied by only this practice for this lesson. Other practices in this lesson retain your `setUserOverrides.sh` file if it exists in your domain; otherwise a default version of the file is placed in the domain.

Practice 4-2: Using Command-Line Tools

Overview

In this practice, you use Java command-line tools to obtain and analyze performance-related metrics of running WebLogic Server JVM processes. These tools may be used freely throughout the rest of the practices in this course if they seem appropriate and suitable to you. The following image depicts the architecture of the domain used for this practice.



About This Course's Approach

This practice provides only guidelines for how to accomplish the objectives of the lesson. You take lessons learned from the slides or previous practices and apply them for this practice on your own.

Dependencies

This practice depends on:

- Practice 2-1 for setting up the course environment.
- Practice 2-3 for configuring Grinder and establishing initial baseline performance numbers of the Auction application in this environment.
- The `setUserOverrides.sh` script in `/practices/tune/practice04-01/resources`. When you run the `setup.sh` script, it detects your existing `setUserOverrides.sh` script and ensures it is in place for you again. If you did not have an existing `setUserOverrides.sh` script in place, then it places the default file in the domain for your convenience.

IMPORTANT! You can now leave Grinder running and reuse it for practices that use the Auction application, or run it separately for each practice when needed the same way you did for practice 4-1.

Tasks

1. Connect to the **`host01`** and **`host02`** machines.
2. Set up the practice environment.
   a. Perform the following commands on **`host01`** to set up the environment for this practice:
      ```
      $ cd /practices/tune/practice04-02
      $ ./setup.sh
      ```
      This script does the following:
      - Restores the domain to its original state. This includes your existing `setUserOverrides.sh` script.
      - Starts the wlsadmin AdminServer
      - Deploys the SimpleAuctionWebAppDb application to the cluster
      - Starts the server1 managed server on `host01`
      - Starts the server2 managed server on `host02`

   b. Wait for all servers to log that they are in the RUNNING state before continuing.
3. Use `jps` to analyze JVM processes.
   a. Read this note on `jps`: The main purpose of `jps` is to find the process ID of a running JVM. It is often referred to as the `lvmid`, or local virtual machine identifier, because Java is a platform-independent language. Essentially, this is Java's way of not tying itself to an operating system construct, such as a process ID (PID), in case there is another mechanism for identifying JVMs outside of PIDs. However, because all the operating systems these days use PIDs, the `lvmid` correlates to the PID of the running process that is executing the JVM.

b. Review the following `jps` usage description and execute `jps` on **`host01`**. After `jstatd` is running on **`host02`**, then run `jps` on **`host01`** to obtain JVM information on **`host02`**.

```
jps [-q] [-mlvVJ] [<hostname>[:<port>]]
```

Option	Description
`-q`	Suppresses all output except for the lvmids of JVM processes. This option cannot be used with the other options.
`-m`	Displays the arguments passed to the main method of the running JVM
`-l`	Displays the full package name of the main class or full path of the application's JAR file
`-v`	Displays the arguments passed to the JVM
`-V`	Displays the arguments passed to the JVM using a flags file
`-J`	Used to pass options to the underlying JVM run with `jps`
`hostname`	Specifies the hostname of the remote host where `jstatd` is running so you can use `jps` to retrieve lvmids of JVMs remotely on that host
`port`	Specifies the port of the remote host where `jstatd` is running. This port should match the port used to run `jstatd`.

Example usage with WebLogic on the same machine:

```
$ jps -lv | grep server1
```

Example usage with WebLogic on a remote machine (after `jstatd` is running on the remote machine):

```
$ jps -lv host02:4001 | grep server2
```

c. Review the `jstatd` usage description below and execute `jstatd` on **`host02`**.

The `jstatd` command creates an RMI network hub for receiving Java command-line tool remote requests.

```
jstatd [-nr][-p port][-n rminame][-Joption]
```

Option	Description
`-nr`	Tells `jstatd` to not create an RMI registry if one is not already running
`-p port`	The default RMI registry port for `jstatd` is 1099. Use this to specify a different port. Your remote clients must also specify this port.
`-n rminame`	The default name for `jstatd` in the RMI registry is `JStatRemoteHost`. Use this to specify a different name for this particular `jstatd` instance.
`-Joption`	Used to pass options to the underlying JVM run with `jstatd`

`Policy file`	The `jstatd` tool does not authenticate remote clients, but does enforce security to dissuade attackers from accessing JVM processes. You can use `jstatd`'s use of `RMISecurityPolicy` to run `jstatd` for this practice. You do this by creating a policy file, such as this example:
	`jstatd.all.policy` file:
	`grant codebase "file:${java.home}/../lib/tools.jar" {`
	`  permission java.security.AllPermission;`
	`};`
	Note: A copy of this file is available in the practice folder's `resources` folder for your convenience.

Example usage (**Hint:** You will need to create the `jstatd.all.policy` file as described in the table above):

```
$ jstatd -J-Djava.security.policy=jstatd.all.policy -p 4001
```

Note: This command should be run on **`host02`**.

d. Reflect on why you would use `jps` and `jstatd`. Could you obtain the same information using operating system commands? If so, what would the differences be?

Hint: Use the information for the tool commands in steps 4b and 4c to answer the questions. **It is OK if you cannot answer every question.**

Example OS remote method by using SSH:

```
$ ssh host02 "ps -ef | grep java | grep server2"
```

e. Experiment with these commands to perform the following tasks:

- Find the PID of each JVM running on `host01`.
- Find the PID of each JVM running on `host02`, but do it from `host01`.
- Find the PID of server1 by using `jps` and `grep`.
- Find the PID of server2 by using `jps`, `jstatd`, and `grep`.

f. Experiment with these commands on your own for 5-10 minutes.

4. Use `jcmd` to manage JVM processes.
   a. Read this note on `jcmd`: The `jcmd` tool is the JRockit `jrcmd` tool ported for use with the HotSpot JVM. It is used to send diagnostic commands to the JVM. This is very useful for obtaining runtime information about your WebLogic Server instances.
   b. Review the following `jcmd` usage description.

```
jcmd [options]
jcmd [pid | main-class] PerfCounter.print
jcmd [pid | main-class] command [arguments]
jcmd [pid | main-class] -f file
```

Option	Description
`-l`	Displays the list of running Java processes including their PID, main class, command-line arguments
`-h -help`	Displays a help message
`pid`	Specifies a particular PID (that represents a running JVM) to which `jcmd` will send commands
`main-class`	Specifies a particular main Java class (running in a JVM) to which `jcmd` will send commands. If multiple Java processes use the same main class, then commands are sent to all matching processes.
`command [arguments]`	The `jcmd` command to send to the JVM. Arguments included depend on the particular command being used.
`-f file`	Performs commands read from the specified file

Here are some examples of using `jcmd` (for reference, 19719 refers to a WebLogic Server instance's PID):

Example	Description
`jcmd`	Displays the PID and class names for running JVM processes
`jcmd 19719 help`	Displays the available `jcmd` commands for the process with the specified PID
`jcmd 19719 help VM.flags`	Displays detailed `jcmd` help for an available command
`jcmd 19719 Thread.print`	Displays a thread dump of a JVM
`jcmd 19719 GC.class_histogram`	Displays a list of classes ordered by those that are using the most memory
`jcmd 19719 VM.command_line`	Displays the JVM command line. Use this to display the command line used to start WebLogic Server.
`jcmd 19719 VM.system_properties`	Displays the system properties that are set for a JVM. Use this to display system properties set for WebLogic Server.
`jcmd 19719 VM.version`	Displays the version of the JVM running in the specified process. Use this to know what JVM version is running your WebLogic Server instances.
`jcmd 19719 PerfCounter.print`	Displays a bunch of performance counters for the JVM. Use this to get a dump of numbers you can use to analyze WebLogic / JVM performance.

c. Experiment with `jcmd` to perform the following tasks:
   - Find the PID of each JVM running on `host01`.
   - Find the PID of server1 by using `jcmd` and `grep`.
   - Display the available help commands for the JVM running server1.
   - Display detailed help for one of the available commands for the JVM running server1.
   - Print a thread dump of the JVM running server1.
   - Print a class histogram for the JVM running server1. Which classes seem to be using the most memory? Why?
   - Print the command-line arguments for the JVM running server1. Why is this useful? What does it help you avoid?
   - Print the system properties for the JVM running server1. What parameters do you find useful in the display? Why?
   - Print the JVM version of the JVM running server1. Why is it useful to know what version of the JVM is running WebLogic?
   - Execute the `PerfCounter` command to view performance data related to the JVM running server1. What type of information does the tool provide?

d. Experiment with these commands on your own for 5-10 minutes.

5. Use `jinfo` to analyze JVM processes.

a. Read this note on `jinfo`: The `jinfo` tool is used to display configuration information for a Java process, core file, or a remote debug server. This is very useful for obtaining runtime information about your WebLogic Server instances.

b. Review the following `jinfo` usage description.

```
jinfo [option] pid
jinfo [option] executable core
jinfo [option] [server-id@]remote-hostname-or-IP
```

Option	Description
`-flag name`	Displays the name and value of the specified command-line argument
`-flag [+|-]name`	Enables or disables the specified boolean command-line argument
`-flag name=value`	Sets the specified command-line argument to the specified value
`-flags`	Displays command-line arguments passed to the JVM
`-sysprops`	Displays Java system properties for the JVM
`-h -help`	Displays a help message

Here are some examples of using `jinfo` (for reference, 19719 refers to a WebLogic Server instance's PID):

Example	Description
`jinfo -help`	Displays the `jinfo` usage statement
`jinfo 19719`	Displays the command-line arguments and system properties that are set for a JVM. Use this to display system properties set for WebLogic Server.
`jinfo -flags 19719`	Displays the JVM command line. Use this to display the command line used to start WebLogic Server.
`jinfo -sysprops 19719`	Displays the system properties that are set for a JVM. Use this to display system properties set for WebLogic Server.

c. Experiment with `jinfo` to perform the following tasks:
   - Display the `jinfo` help message.
   - Print the command-line arguments for the JVM running server1. Why is this useful? What does it help you avoid?
   - Print the system properties for the JVM running server1. What parameters do you find useful in the display? Why?

d. Experiment with these commands on your own for 5 minutes.

6. Use `jstat` to view JVM performance statistics.
   a. Read this note on `jstat`: The `jstat` tool is used to display performance statistics for an instrumented HotSpot JVM process. This is very useful for obtaining runtime information about your WebLogic Server instances.

b. Review the following `jstat` usage description.

```
jstat -help | -options
jstat option [-t] [-h lines] vmid [interval [count]]
```

The first example uses a single general command-line option, either `-help` or `-options`. When a general option is used, no other options are valid. General options are described in the following table:

Option	Description
`-help`	
`-options`	Displays a list of options available with the `jstat` command. The description of these options is listed in the next table.

Output options and other arguments are described in the following table:

Option	Description
`-class`	Displays class loader statistics
`-compiler`	Displays JIT compiler statistics
`-gc`	Displays garbage collector statistics
`-gccapacity`	Displays size statistics for the generations of the heap
`-gccause`	Displays garbage collector statistics, plus the cause of the last and current GC events
`-gcnew`	Displays young generation statistics
`-gcnewcapacity`	Displays size statistics for the young generation
`-gcold`	Displays old and permanent generation statistics
`-gcoldcapacity`	Displays size statistics for the old generation
`-gcpermcapacity`	Displays size statistics for the permanent generation
`-gcutil`	Displays garbage collection statistics
`-printcompilation`	Displays compilation method statistics
`-t`	Includes a timestamp column in the output
`-h n`	Displays a column header every n number of rows
`-Joption`	Passes Java options to the java application launcher

`vmid`	Specifies the vmid of the JVM process to use. The format for vmid is:
	`[protocol:][//]`
	`lvmid[@hostname][:port][/servername]`
	See the documentation for the description of each aspect of the URL.

`interval`	Time interval in between samples
`count`	The number of samples to display

Several of `jstat`'s command options produce displays with headings that can be cryptic at first glance. Execute the following command to view the documentation for `jstat` to see descriptions of the columns for each option:

```
$ firefox resources/jstat.html &
```

Find the examples of using `jstat` in the documentation and review them. The `jstat` tool is very useful for analyzing Java heap and garbage collection statistics. The table below shows more examples of using `jstat`:

Example	Description
`jstat -gcutil -h 10 19719 2000`	Displays a list of garbage collection statistics, including the size and performance of each space in the heap. Prints a header every 10 lines, and prints a new row every 2000 ms.
`jstat -gcutil -h 10 7429@host02:4001 2000`	Displays the same statistics as the previous command above, but is reporting statistics for a remote JVM running on host02. **Note:** For reference, 7429 refers to the PID for server2 obtained by using `jps` and `jstatd`.
`jstat -gcnew 19719 2000 10`	Displays statistics for the young generation, and prints a new row every 2000 ms for a count of 10.
`jstat -gcnewcapacity 19719 2000 10`	Displays capacity-related statistics for the young generation, and prints a new row every 2000 ms for a count of 10.
`jstat -gccause -t -h 25 19719 2000`	Displays garbage collection statistics, including the reason for GC if any. Prints a header every 25 rows, and time in between rows is 2000 ms. Includes a timestamp in the output.

c. Experiment with `jstat` to perform the following tasks. You can optionally run Grinder to cause the numbers to have more meaning:
   - Display the `jstat` help message.
   - Display the list of `jstat` options.
   - Display garbage collection and heap statistics for the JVM running server1.
   - Display garbage collection and heap statistics for the JVM running server2 from the `host01` machine. Hint: Requires using `jstatd`, which may still be running from an earlier step in this practice.
   - Include timestamp information in your output.

d. Experiment with these commands on your own for 5 minutes.

7. Use `jstack` to view and analyze JVM thread stack traces.
   a. Read this note on `jstack`: The `jstack` tool is used to display stack traces of Java threads for a Java process, core file, or a remote debug server. This is very useful for obtaining runtime information about your WebLogic Server instances. Thread dumps are displayed in the console window where the command is executed, rather than the server's logs or console window. You should redirect the output of `jstack` to different files for comparisons. This is because a thread dump is only a single snap shot in time of the state of a server. Thread dumps have to be analyzed over a series of thread dumps to track any trends in how threads are executing or having issues. From a performance perspective, `jstack` provides only information pertaining to threads that are stuck, long-running, locked, or busy. They do not provide performance numbers or metrics to measure performance. It is up to you to analyze what the threads are doing and to determine if they are stuck. And from an administrative perspective, you would not be expected to understand the methods listed in stack traces because they are written by developers. You would be focused mainly on trends of thread deadlocks or overtly active threads. Our application is functioning well at this point, so you should not detect any locked threads or performance issues at this point. When you are working with stuck thread behavior in this course, you will have a chance to use `jstack` again.
   b. Review the following `jstack` usage description.

```
jstack [option] pid
jstack [option] executable core
jstack [option] [server-id@]remote-hostname-or-IP
```

Option	Description
`-F`	Force a thread dump when "`jstack [-l] pid`" does not respond.
`-l`	Display a long listing that includes information about locks, such as a list of owned `java.util.concurrent` ownable subscribers.
`-m`	Display a mixed mode listing of Java and native C/C++ frames in the thread dump.
`-h -help`	Display a help message.

Here are some examples of using `jstack` (for reference, 19719 refers to a WebLogic Server instance's PID):

Example	Description
`jstack -l 19719`	Displays a thread dump of Java threads running in the JVM running WebLogic server1. Prints a detailed long listing.
`jstack -m 19719`	Displays a thread dump of Java threads running in the JVM running WebLogic server1. This prints native methods as well as Java methods.

c. Experiment with these commands on your own for 10 minutes.

8. Leave the environment running for the next practice.

Course Developer Corner:

What you should have noticed with this practice is that there are several different Java command-line commands available to you. Some of these commands provide overlapping features, such as getting the process information for a running JVM, or the arguments passed to the JVM main class. You may naturally have a better feel for one or more tools than other tools initially. The most commonly used tool is `gcutil`, because it supplies good details of the Java heap space.

During this practice, you were asked what the difference was between using `jps` to get JVM information remotely versus using operating system commands. The first thing that you should have noticed is that you have to use SSH to execute commands on the other host. SSH is much harder to set up than `jstatd` and the command to get information is easier. Of course if you have access to the remote server you could just run the local `jps` command instead. The `jps` command only requires that Java is installed on both machines, and that the `jstatd` process is running.

The primary focus of this practice is to get you used to obtaining and analyzing JVM performance-related metrics of a running WebLogic Server using JVM command-line options.

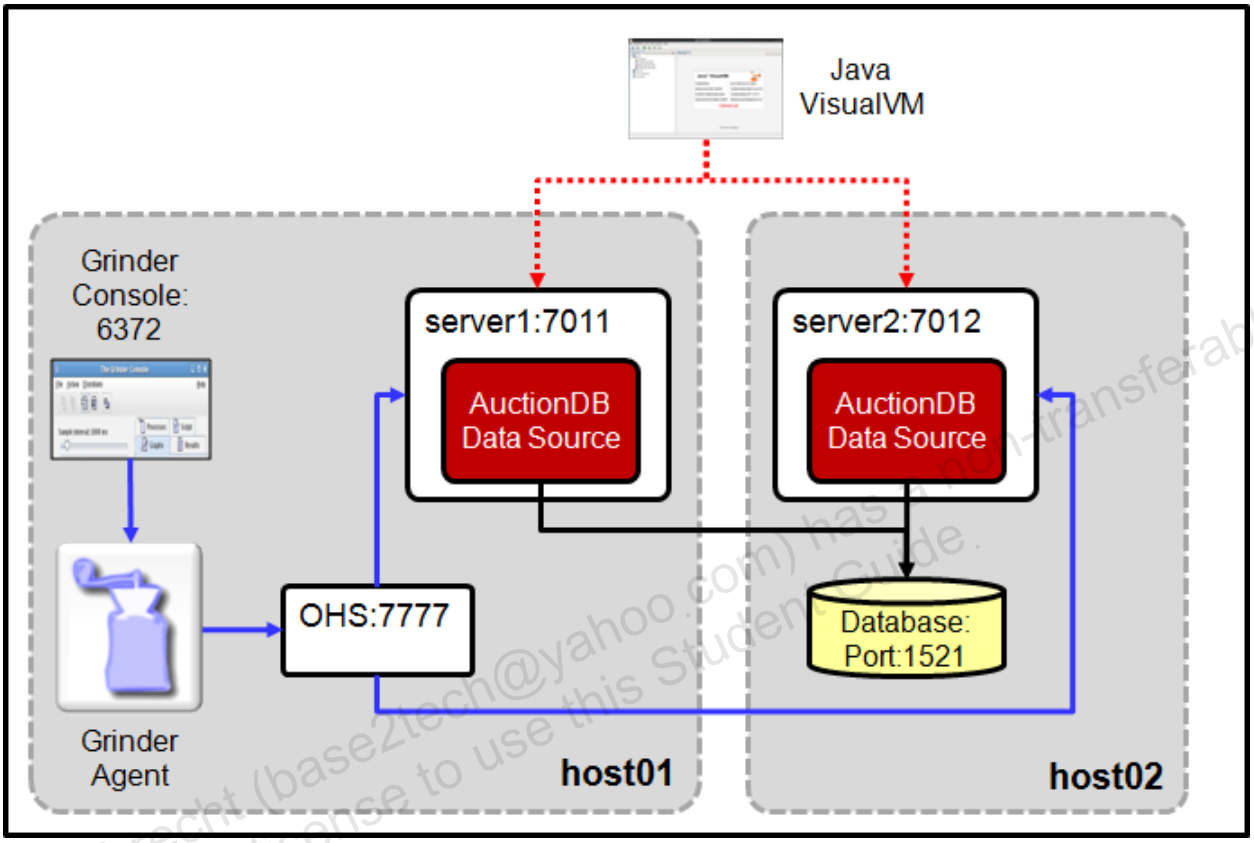
Practice Solution: Using Command-Line Tools

There is no solution for this practice. Other practices in this lesson depend on the experience gained in this practice. Additionally, the experience gained in this practice can be applied in subsequent lessons.

Practice 4-3: Using Java VisualVM

Overview

In this practice, you use Java VisualVM to monitor your running WebLogic Server instances graphically. The following image depicts the architecture of the domain used for this practice.



About This Course's Approach

This practice provides only guidelines for how to accomplish the objectives of the lesson. You take lessons learned from the slides or previous practices and apply them for this practice on your own.

Dependencies

This practice depends on:

- Practice 2-1 for setting up the course environment.
- Practice 2-3 for configuring Grinder and establishing initial baseline performance numbers of the Auction application in this environment.
- The `setUserOverrides.sh` script in `/practices/tune/practice04-01/resources`. When you run the `setup.sh` script, it detects your existing `setUserOverrides.sh` script and ensures it is in place for you again. If you did not have an existing `setUserOverrides.sh` script in place, then it places the default file in the domain for your convenience.

Tasks

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

2. Set up the practice.
   a. Perform the following commands on **host01** to set up the environment for this practice. If your servers are still running from the previous practice, then you should skip running the `setup.sh` script.

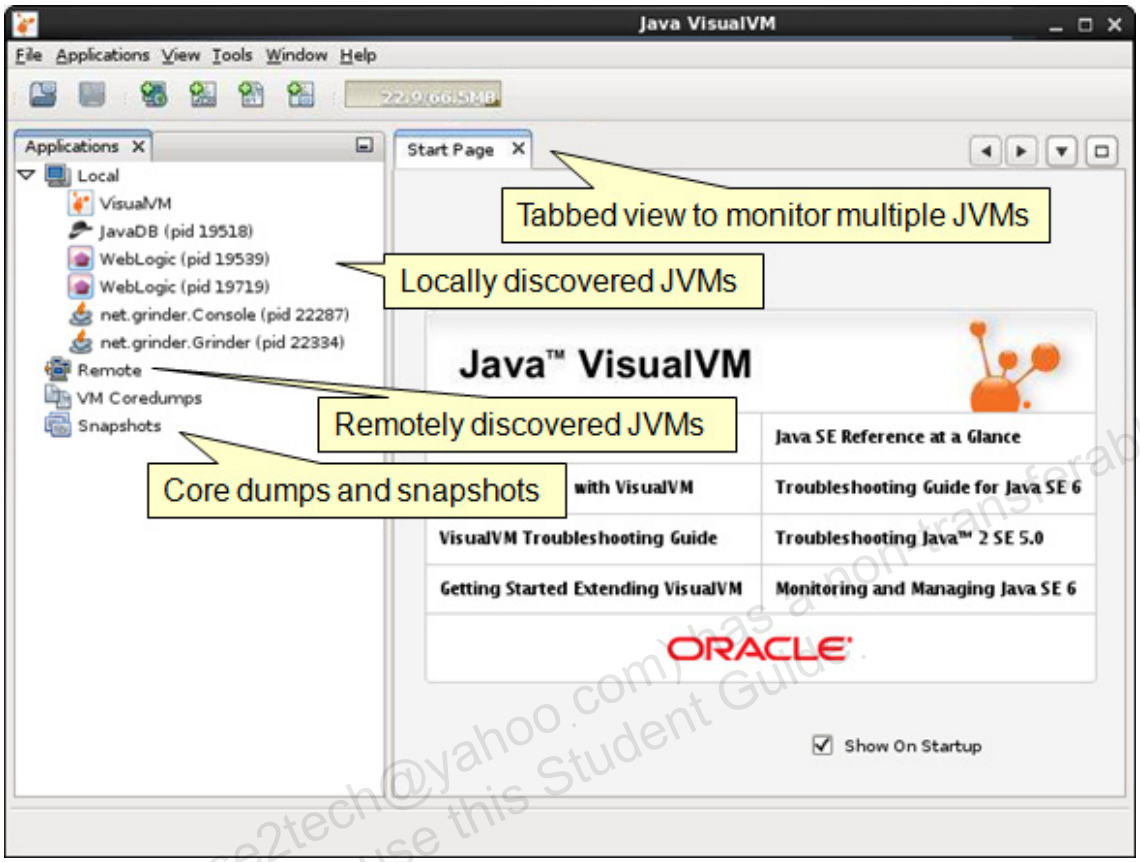
      ```
      $ cd /practices/tune/practice04-03
      $ ./setup.sh
      ```

      This script does the following:
      - Restores the domain to its original state. This includes your existing `setUserOverrides.sh` script.
      - Starts the wlsadmin AdminServer
      - Deploys the SimpleAuctionWebAppDb application to the cluster
      - Starts the server1 managed server on `host01`
      - Starts the server2 managed server on `host02`

   b. Wait for all servers to log that they are in the RUNNING state before continuing.
3. Start Java VisualVM.
   a. The script to start VisualVM is located in `/u01/app/jdk/bin`. This directory is already in your path, so you can start VisualVM from any location. Execute the script to start the tool:

      ```
      $ jvisualvm
      ```

b. Verify that the GUI appears.



4. Connect to local WebLogic managed server with VisualVM.
   a. In the left-hand panel, double-click one of the WebLogic JVMs in the list. The managed server is likely the last entry because it started later. This opens the JVM tab.
   b. In the Overview tab, verify it is server1 by reviewing the JVM Arguments subtab and checking that `-Dweblogic.Name=server1`.

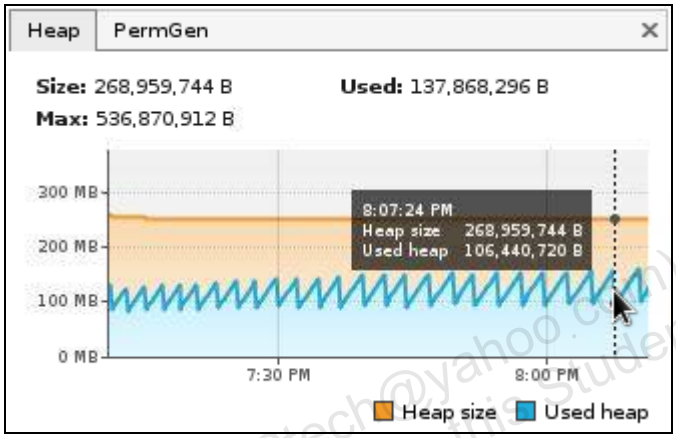
c. Review the descriptions of the JVM tabs in the following table:

Tab	Description
Overview	Provides an application snapshot where you can see the PID, host, main class, main class arguments, JVM information, Java version, Java Home, and JVM flags. There are additional subtabs that let you view JVM arguments and system properties.
Monitor	Provides a graphical multiple chart view of the JVM process, including CPU and GC utilization, memory use, class count, and thread count. This page also displays the PID, uptime, and some buttons for performing a GC and a heap dump. Hovering the mouse over a chart causes a popup window to appear with more detailed information.
Threads	Provides a list of all JVM threads and allows you to view threads in timeline, table, or detailed mode. Each thread is displayed and is color coordinated to distinguish between running, sleeping, waiting, and monitoring threads. You can select some threads and right-click to get more thread details. You can choose to see all threads, some threads, or just one thread.
Sampler	Provides an interface for analyzing runtime behavior of your Java applications. You set CPU and memory settings here and get calculated metrics of the following, which is analyzed from collected stack trace data over several intervals: • CPU: Displays which Java methods or threads are using the most CPU. • Memory: Displays which Java methods or threads are using the most memory.
Profiler	Provides an interface for analyzing runtime behavior of your Java applications. You set CPU and memory settings here and get real time metrics of the following, which is captured in real time as methods are executed: • CPU: Displays which Java methods or threads are using the most CPU. • Memory: Displays which Java methods or threads are using the most memory.

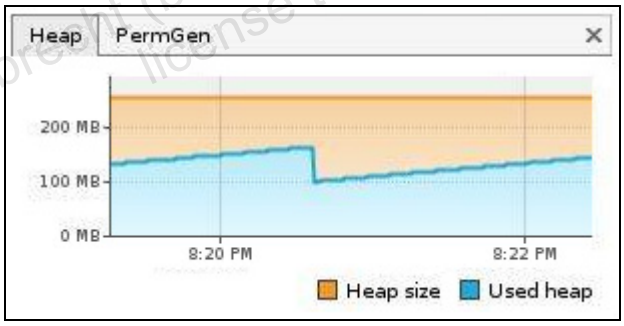
Note: The Sampler and Profiler tabs seem similar but work very differently. Because Java VisualVM is a generic tool for all Java processes, it is not necessarily the right tool for every job, such as profiling huge application code bases, such as WebLogic. The Sampler tab is faster, gathers information from thread dumps at regular time intervals. The Profiler tab instruments the byte code of the JVM, and in the case of an application server like WebLogic, this instrumentation can take a long time, and oftentimes can simply just not work when profiling is complete. It is not recommended to use the profiling tool for WebLogic. The Sampler is faster but less accurate. In any case, understanding which methods or threads are using the most memory or CPU is also something that interests a developer more because they are the ones who have to go back to the code and try to improve the results.

5. Use VisualVM to answer the following questions. You may run Grinder to load the application for some of the questions.
   a. What is the operating system architecture?
   b. Is the server running in production mode?
   c. Find the graphic charts for CPU, heap, and thread monitoring. What are the characteristics of these metrics? You should see a gradual increase in memory, followed by occasional declines as garbage collections occur. If the heap appears to be too jagged, meaning that there are frequent memory spikes and drops, then memory may not be sized properly or there could be some other problem.

      The following image shows a server that may be doing too much garbage collection. Then again, it could be ok. It depends on how much work the server is doing. Is the CPU heavily used for garbage collections? In reality, this is normal, at rest garbage collection. It only looks jagged because it encompasses a time period of about an hour.



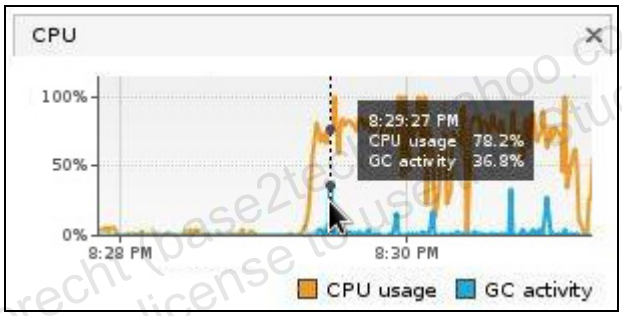
Here is an example of another chart showing the heap and garbage collection in a shorter time period:



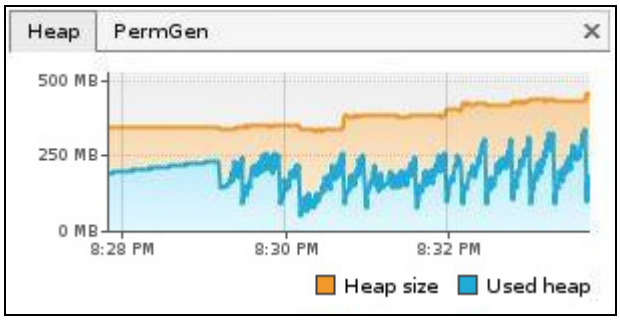
6. Use VisualVM to connect to server2 on `host01`.
   a. Configure VisualVM to connect to `jstatd`, running on `host02`, so that it can remotely monitor the server2 managed WebLogic Server JVM.

      Hint: You should have already successfully run the `jstatd` command in Practice 4-2, step 3c and it should still be running. If not, then start it again.
   b. Open server2 so it has its own monitoring tab.
   c. Verify that it is the JVM for server2 by checking its server name property.
   d. Notice that server1 and server2 each have their own tabs.
   e. You can drag the server2 (or any tab) off the main VisualVM window so it creates its own window. This makes it so you can view both servers side by side.
   f. Alternatively, you can dock the tab in the main window so both tabs are visible at the same time for side-by-side viewing as well.
   g. Run Grinder and watch the monitoring tabs for each server to see how each behaves. Alternatively, you can stop remotely monitoring server2 and start another instance of VisualVM locally on `host02` so you can see CPU metrics.

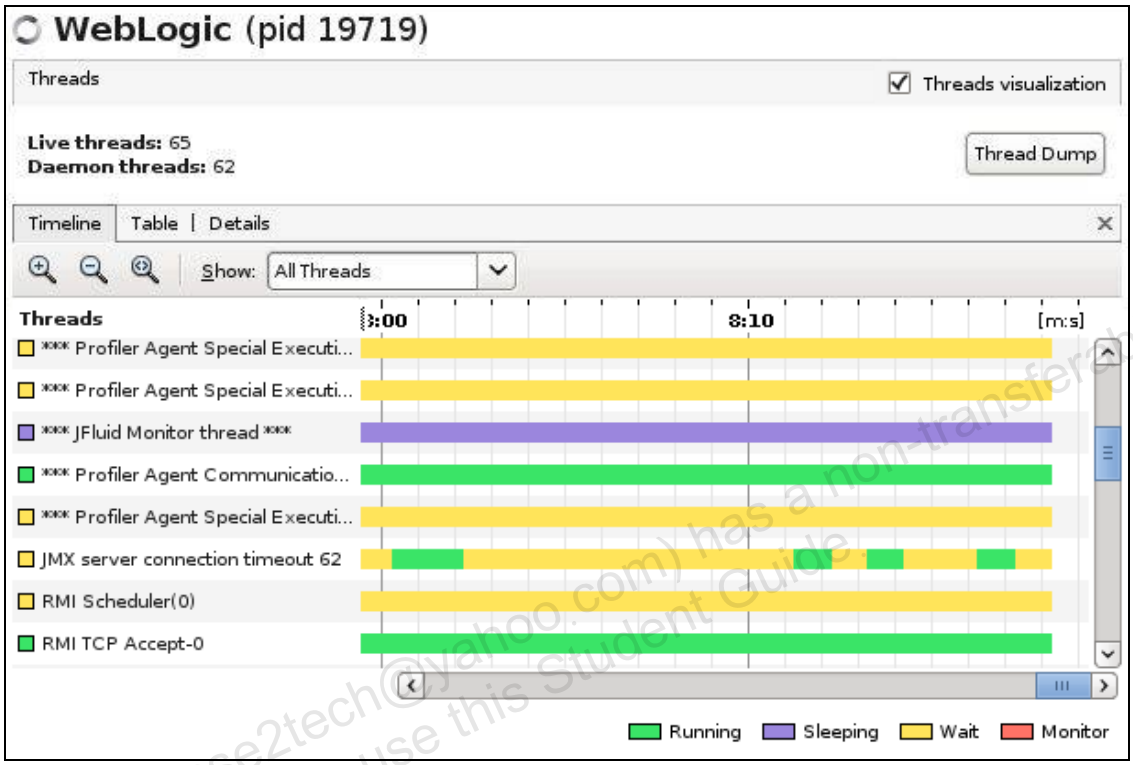
      While the application is loaded by Grinder, you can see the CPU reacting to the workload. The orange line represents the CPU utilization, while the blue line represents the CPU utilized to perform garbage collection.



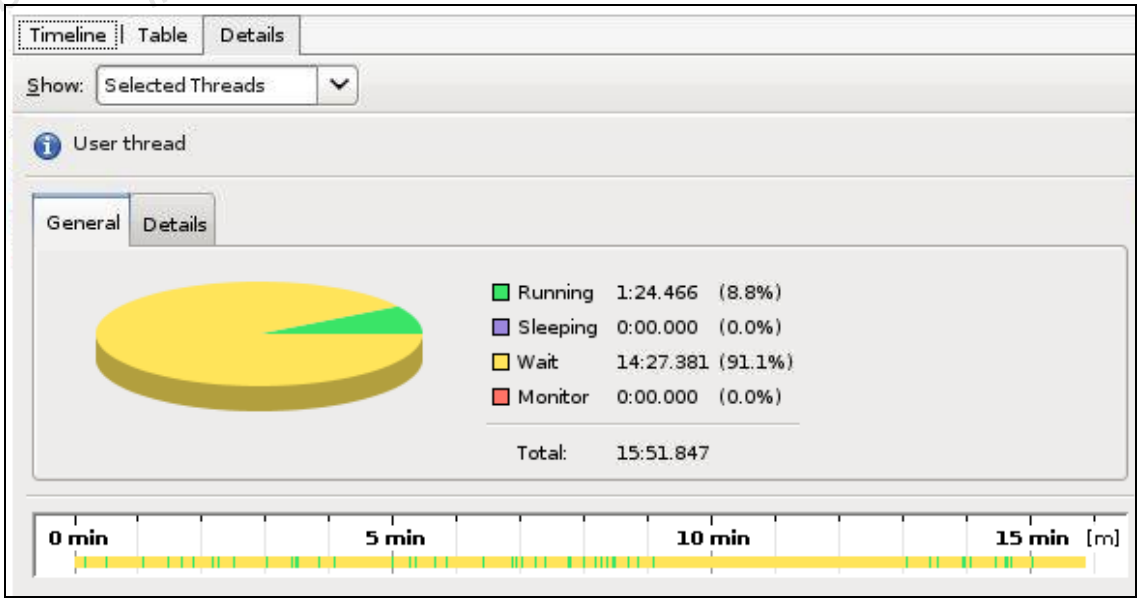
Here is an example of heavier garbage collection activity while the application is under load. Notice how the heap size is growing toward the maximum 512 MB setting.



7. Use VisualVM to monitor server1 threads.
   a. Use the Threads tab to view the active threads in the server1 JVM. The legend on the lower right-hand side shows the color scheme used to denote which threads are running, sleeping, waiting, or monitoring.



   b. You can right-click a thread in the list and select “Thread Details” to view more information about that particular thread. The following image shows a thread that spent 91.1% of its lifetime in a Wait state, and 8.8% in a Running state.



8. Experiment with the Sampler tab to view some metrics.
9. Avoid using the Profiler tab.
10. Use VisualVM under load.
    a. Set the heap size to 50 MB again in the `setUserOverrides.sh` file and restart server1.
    b. Run Grinder to load the application.
    c. Use VisualVM to monitor CPU, heap, and threads.
    d. What do you notice about CPU activity in this scenario?
    e. How does GC CPU utilization compare to overall CPU utilization?
    f. How often does garbage collection appear to be happening?
    g. What are the basic memory and garbage collection activities in this scenario?
    h. Are there more or fewer threads running now?
11. Experiment with VisualVM on your own for 5 minutes.
12. Leave the environment running for the next practice.
    The next practice uses Grinder and the same environment so it is best to leave it running.

Course Developer Corner:

You saw how the server functioned when you set the heap size to a very small size by using Java command-line tools and options. This practice lets you see the same thing graphically using VisualVM. The primary focus of this practice is to get you used to using VisualVM graphical tools to capture performance metrics, analyzing the results, and determining the current capabilities of your system.

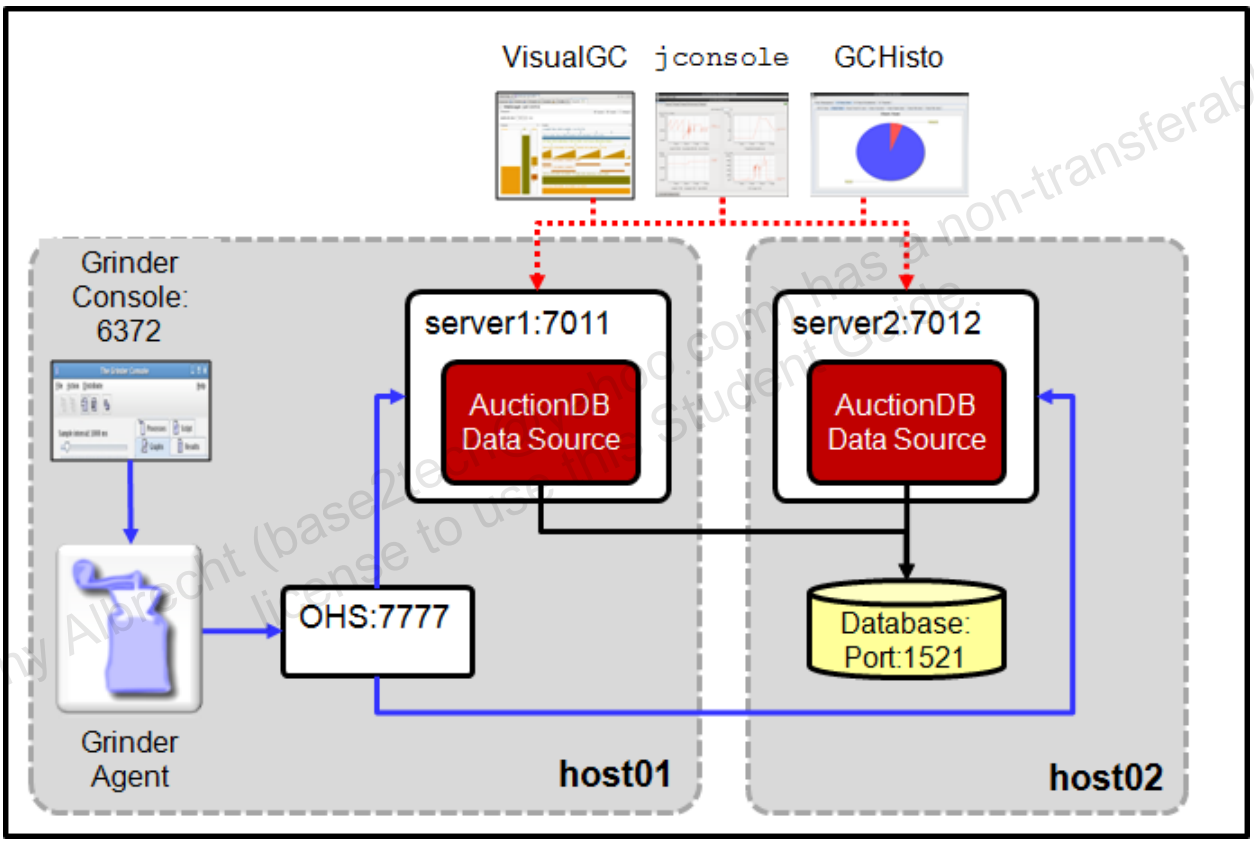
Practice Solution: Using Java VisualVM

There is no solution for this practice. Other practices in this lesson depend on the experience gained in this practice. Additionally, the experience gained in this practice can be applied in subsequent lessons.

Practice 4-4: Using VisualGC, jconsole, and GCHisto

Overview

This practice continues your journey into exploring some of the graphical tools available to you for gaining visibility into your JVM processes that are running WebLogic Server. You will install VisualGC as a plug-in into VisualVM, and use it to graphically visualize how garbage collection is really taking place in the application. Next, you run jconsole and connect it to your running WebLogic server instances to view some of the same metrics, but also to browse the live MBean tree of your WebLogic server. Then you install and run GCHisto to view garbage collection history of your running WebLogic Server. The following image depicts the architecture of the domain and tools used for this practice.



About This Course's Approach

This practice provides only guidelines for how to accomplish the objectives of the lesson. You take lessons learned from the slides or previous practices and apply them for this practice on your own.

Dependencies

This practice depends on:

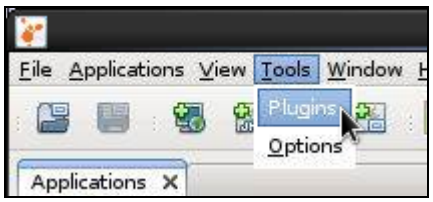
- Practice 2-1 for setting up the course environment.
- Practice 2-3 for configuring Grinder and establishing initial baseline performance numbers of the Auction application in this environment.
- Practice 4-3 for using VisualVM.
- The `setUserOverrides.sh` script in `/practices/tune/practice04-01/resources`. When you run the `setup.sh` script, it detects your existing `setUserOverrides.sh` script and ensures it is in place for you again. If you did not have an existing `setUserOverrides.sh` script in place, then it places the default file in the domain for your convenience.

Tasks

1. Connect to the **`host01`** and **`host02`** machines.
2. Set up the practice.
   a. Perform the following commands **on host01** to set up the environment for this practice. If your servers are still running from the previous practice, then you should skip running the `setup.sh` script.

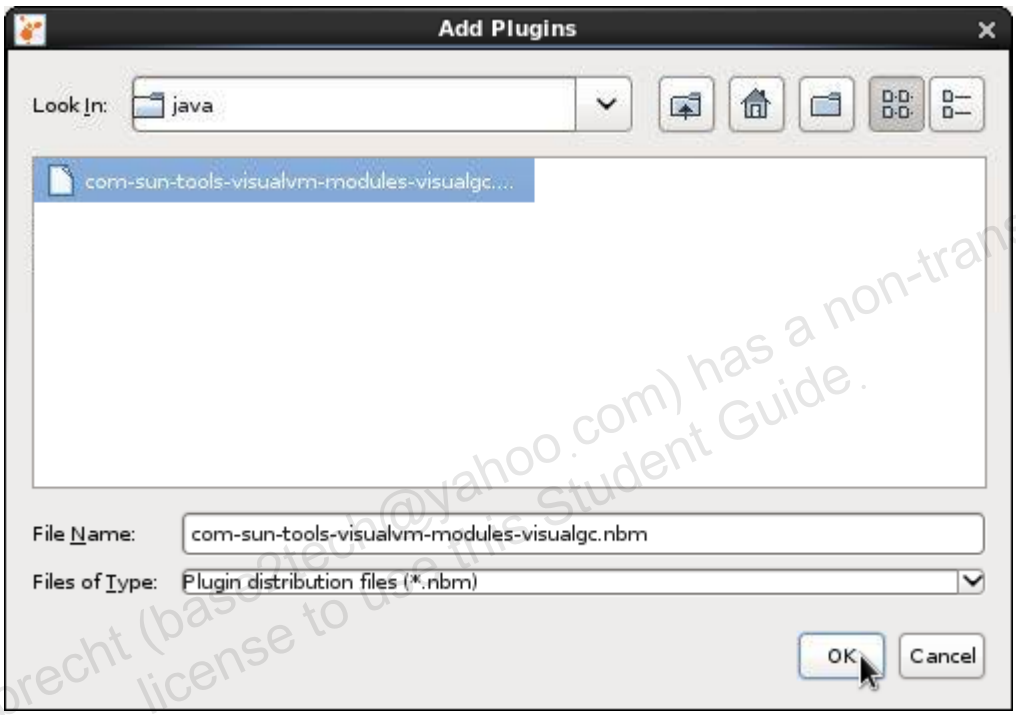
      ```
      $ cd /practices/tune/practice04-04
      $ ./setup.sh
      ```

      This script does the following:
      - Restores the domain to its original state. This includes your existing `setUserOverrides.sh` script.
      - Starts the wlsadmin AdminServer
      - Deploys the SimpleAuctionWebAppDb application to the cluster
      - Starts the server1 managed server on `host01`
      - Starts the server2 managed server on `host02`
3. Wait for all servers to log that they are in the RUNNING state before continuing.
4. Use VisualGC to monitor garbage collection.
   a. First, you have to install the VisualGC plug-in for VisualVM. The plug-in file is located in the `/install/java` folder, and the file is named `com-sun-tools-visualvm-modules-visualgc.nbm`. Within Visual VM, select Tools > Plugins.

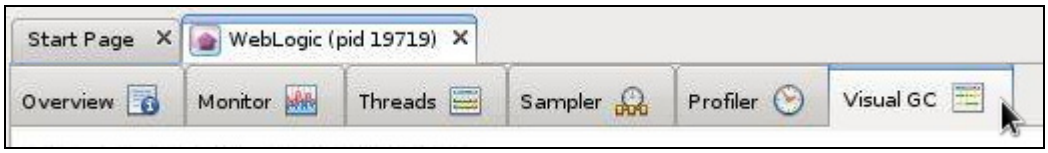


b. The Plugins dialog window appears. Click the Downloaded tab and click Add Plugins.

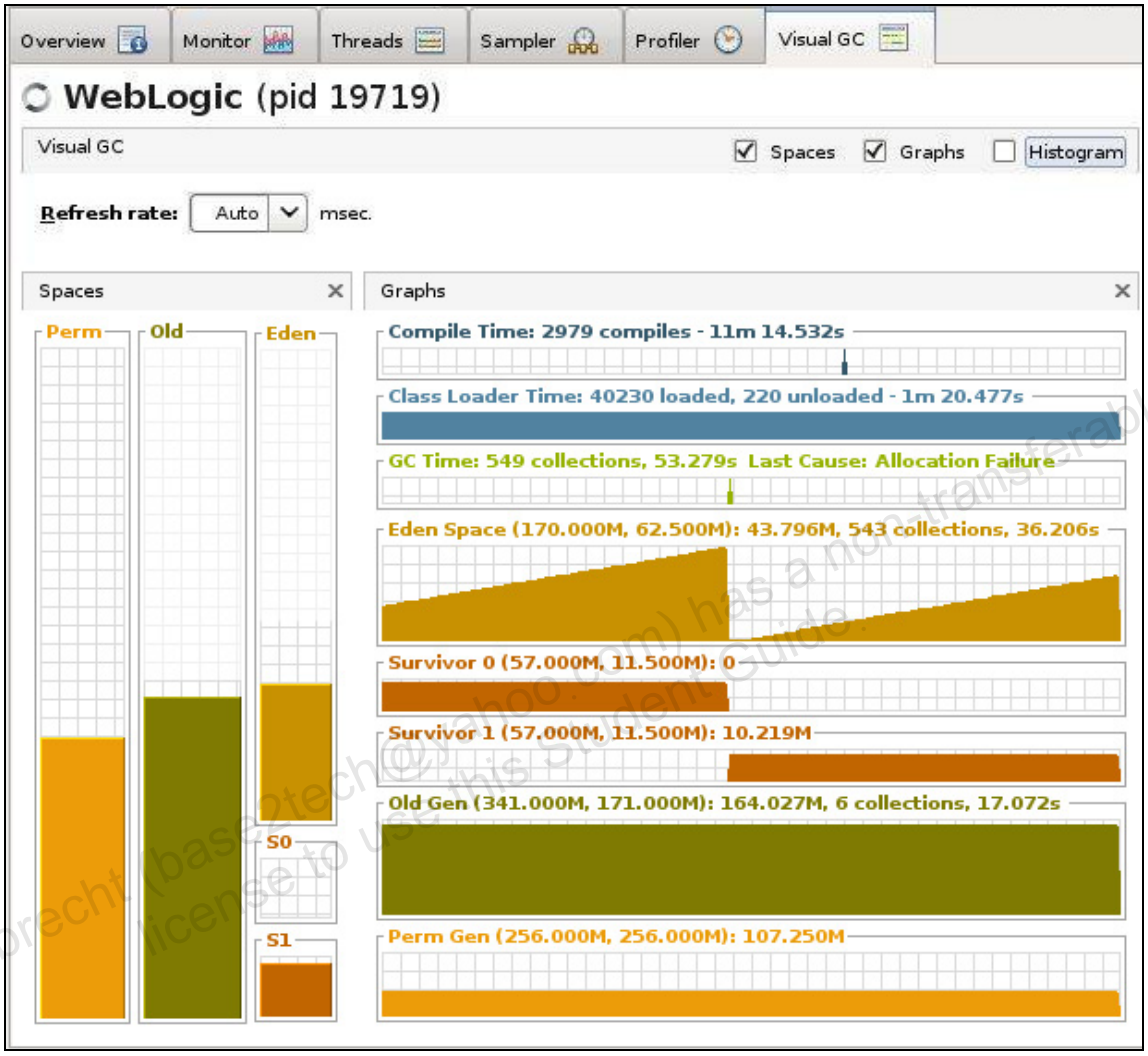
c. Browse to the file location and click OK to install the plug-in.
Note: Your view may be slightly different.



d. Click Install and complete the installation wizard to install the VisualGC plug-in.
e. Click Close to return to VisualVM.
f. Close any tabs you may have open for monitoring WebLogic JVMs. They must be reopened in order for the VisualGC tab to appear.
g. After you reconnect to the server1 JVM, click the VisualGC tab that now appears on the end.



h. Now you should be able to visually see your Java heaps and how they are being used. You can change the refresh rate, but it refreshes automatically at an acceptable rate.



i. Watch the display until you see a garbage collection take place. It usually takes a minute or two. You will know when it takes place because Eden space gets cleared to 0 bytes, and you will see objects move from one Survivor space to the other. Note the numbers of the heap sizes for every space. They provide the minimum, maximum, and used space sizes. Note what the maximum sizes of each heap space are on paper. They should be similar to the sizes shown in the screenshot above.

j. Run Grinder again with server1's memory settings set to 90MB again so you can see garbage collection activity using VisualGC. You will have to reconnect to the JVM again to monitor it. Maximizing the VisualVM window gives the best viewing results.

k. Now note the heap sizes for every space again and compare them to your previous sizes. How are they different? How does this affect the performance of your application?

l. Note the number of collections reported for each space. How many collections were there for each? How much time did it take collectively to perform those collections?

m. Did your application show any errors? What can you ascertain from this?

n. You should notice that although the application is running fine that the garbage collector is working very hard to keep the application running. Just as you saw with command-line tools, this visual representation provides you with an instant and intuitive way of knowing how garbage collection is working. You get more detailed information from the command-line, but get fast results with a visual tool.

o. Close all VisualVM instances.

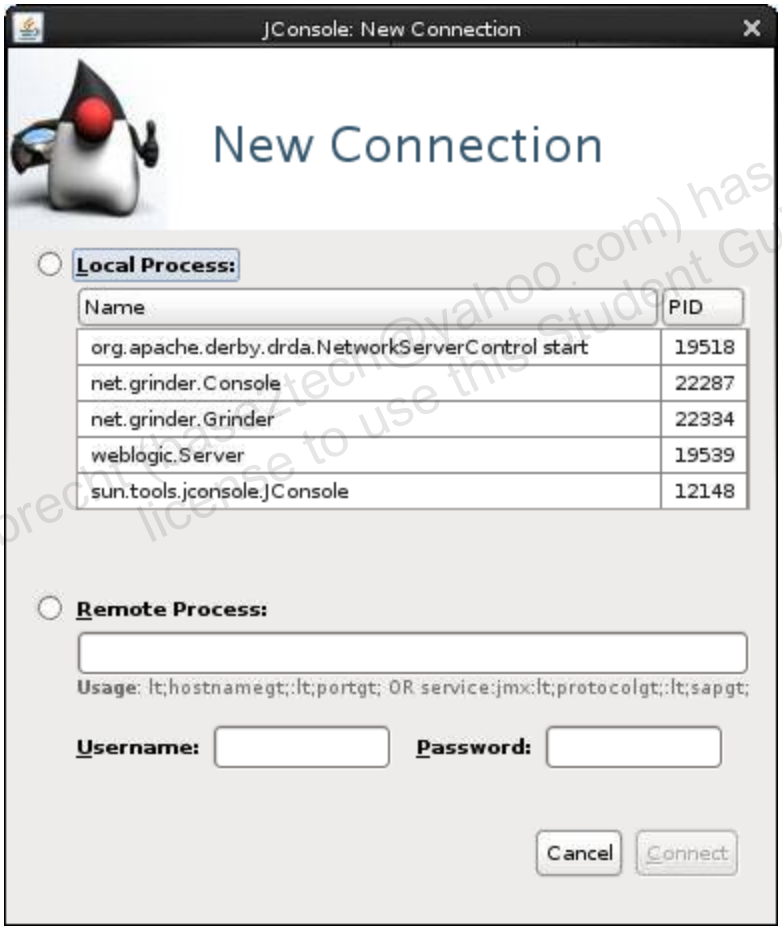
5. Use jconsole to monitor and analyze your JVMs running WebLogic Server.

a. Verify that your domain is running and that server1 is running with a heap size of 50 MB.

b. The jconsole client is included with the JVM. It is located in the Java `bin` directory, which is in your path. Execute the following command to start jconsole:

```
$ jconsole
```

c. The *New Connection* window appears. This window displays local JVMs that jconsole has discovered, and optionally allows you to connect to a remote JVM.

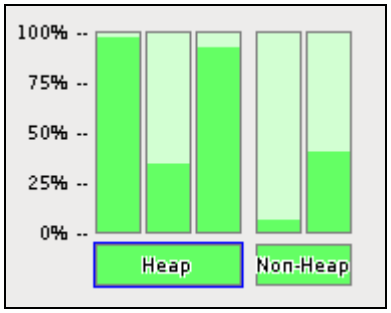


d. Select the server1 JVM and click Connect.

e. You should get an SSL failure because you have not configured an SSL connection for jconsole. Click Insecure to continue on a non-SSL connection.

f. The interface for jconsole is very similar to other consoles, such as VisualVM. There is a tab (on the bottom) for the JVM process, and there are multiple tabs for each functional area, including overview, memory, threads, classes, VM summary, and a new one called MBeans.

g. Click the Memory tab to see the heap in more detail. Notice that just like VisualGC, there is a graphical representation of the Java heap. You can place the cursor over a space to see which space the graphics represent. You can click each space to change the overall chart to display the graph for that space.

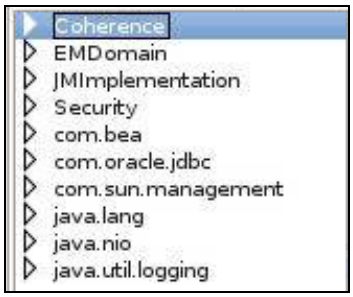


h. Click through each tab and experiment with each. The capabilities are very similar to what you have already used with other tools so there are no detailed instructions for this part.

i. Click the MBeans tab to display the list of MBeans for this JVM.

j. You should see a list of MBeans similar to what is shown in the following image. The MBeans that start with `com.bea` are WebLogic Server runtime MBeans that contain the runtime information of server1. You should notice within the `com.bea` node that there are two server1 nodes. The first one represents WebLogic configuration MBeans that contain the configuration associated with server1. This data is persisted in the domain's `config.xml` file. The second one represents the server's runtime properties. This is where metrics associated with the running server are maintained. When the server JVM is gone, this information is also gone unless it was persisted in some way.

Note: Your view may be slightly different.



k. Browse to server1 (try each one to find the correct attributes) > AuctionImageServlet > SimpleAuctionWebAppDb > ServletRuntime > server1_/SimpleAuctionWebAppDb > Attributes. You should see a list of attributes for the MBean.

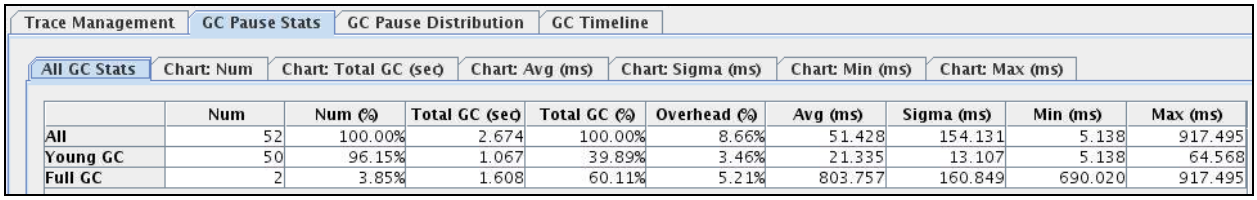
l. Run Grinder to load the application. Let it run for a few moments to generate some data.

m. Click Refresh on the page in jconsole to update attribute data values. You should see some real numbers in your display. If you keep refreshing the screen, you can see the numbers change as the server tries to work through the load that is generated by Grinder. This includes execution average time, total execution time, invocation count, and more. This is one way to view WebLogic Server runtime statistics, but is still a lot of effort to try to access information that should be more easily available. But still, it does provide the information you need.
n. Double-click attributes to expand them into a graphical chart. You may need to run Grinder again to see worthwhile results.
o. Experiment with jconsole on your own for 5 minutes.

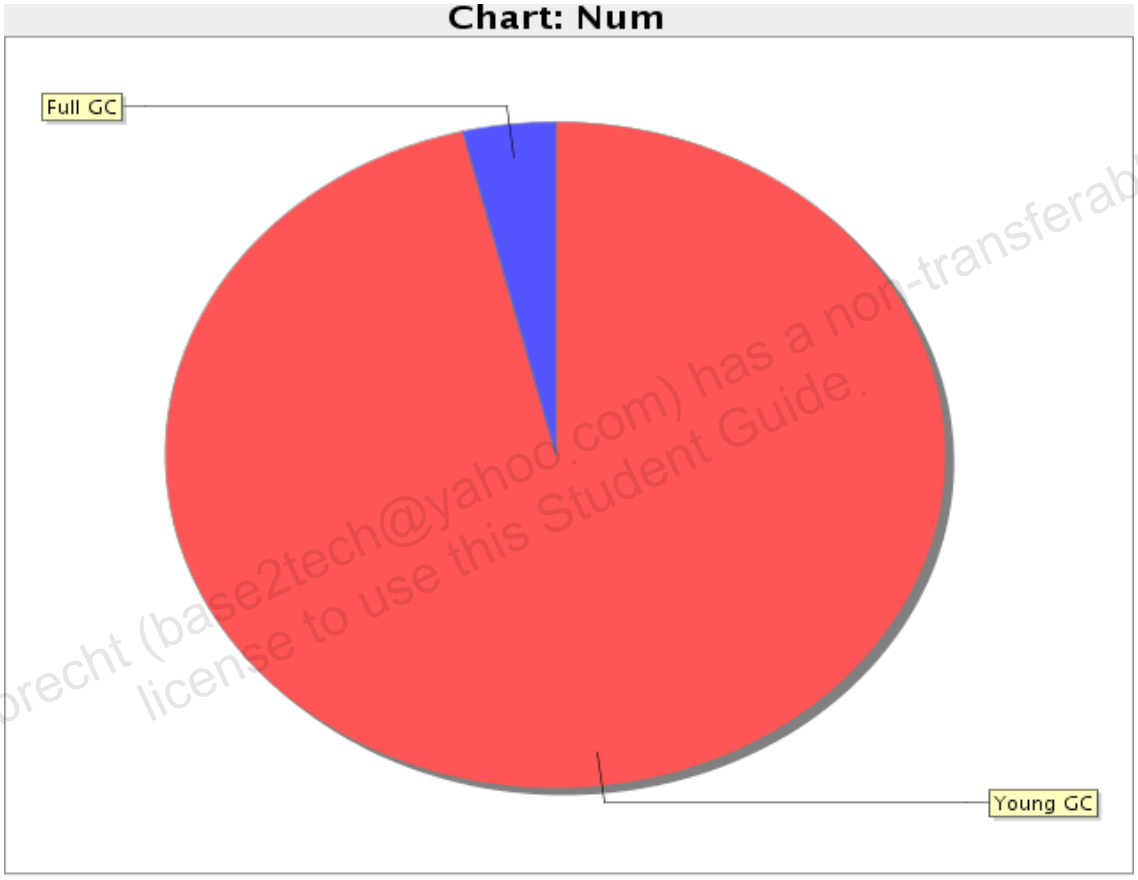
6. Use GCHisto to graphically analyze GC log output.
   a. First, you have to locate the GCHisto program. The program is located in the `/install/java/GCHisto` folder, and the file is named `GChisto.jar`.
   b. Change your `setUserOverrides.sh` file again to print GC statistics again. This time, include the following option to specify a particular file, `/tmp/gc.log`, for GC statistics to get written (keep memory settings to 90MB):
      ```
      -Xloggc:/tmp/gc.log
      ```
   c. Restart server1 to realize your changes.
   d. Verify that the `/tmp/gc.log` file exists and is getting GC statistics written to it.
   e. Execute the following command to start GCHisto:
      ```
      java -jar /install/java/GCHisto/GCHisto.jar
      ```
   f. Run Grinder again to load the system and generate some real logs. Let it run for a few moments before continuing.
   g. Configure GCHisto to point to your GC log file by selecting *Dynamic HotSpot GC Log* and clicking Add. If you select *HotSpot GC Log*, then you only get a single historic snapshot of actual data. The dynamic option allows GCHisto to give you updated metrics.

   h. Browse to your GC log file and select it for GCHisto to use.
   i. Experiment with Real Playback and Fast Playback options.

j. Use the GCHisto tabs to view the various charts to review garbage collection history of server1. GCHisto reads the `gc.log` file and parses the GC records to create human readable charts.



	Num	Num (%)	Total GC (sec)	Total GC (%)	Overhead (%)	Avg (ms)	Sigma (ms)	Min (ms)	Max (ms)
All	52	100.00%	2.674	100.00%	8.66%	51.428	154.131	5.138	917.495
Young GC	50	96.15%	1.067	39.89%	3.46%	21.335	13.107	5.138	64.568
Full GC	2	3.85%	1.608	60.11%	5.21%	803.757	160.849	690.020	917.495



k. Experiment on your own with GCHisto for 5 minutes.

l. Close GCHisto.

7. Reflect for a moment about the tools you have learned so far.
   a. Now that you have used command-line tools and graphical tools to monitor and analyze JVM performance characteristics, what are your thoughts when comparing each of the tools you used?
   b. When would you use each tool? What benefits do you see using command-line tools or graphical tools? Do you have a preference? Why?
   c. Several of the tools provide similar, if not identical, capabilities. Which of these tools do you think provides you with the best of these capabilities? Why?
8. Leave the environment running for the next practice.
   The next practice uses Grinder and the same environment so it is best to leave it running.

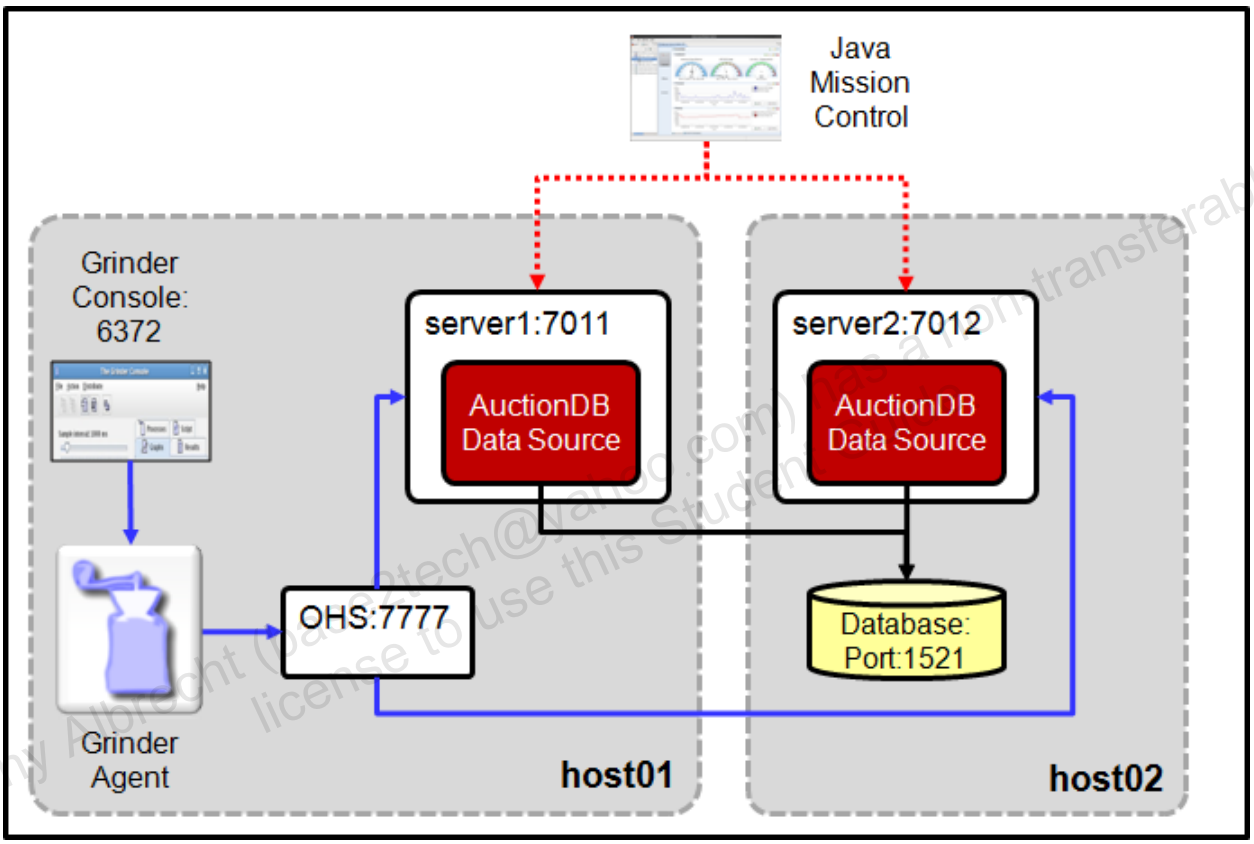
Practice Solution: Using VisualGC, jconsole, and GCHisto

There is no solution for this practice. No practices depend on this practice.

Practice 4-5: Using Mission Control

Overview

The primary diagnostic tools that were part of the JRockit JVM implementation are now part of the HotSpot JVM. The first of these tools is Mission Control. In this practice, you use Mission Control to monitor and analyze your WebLogic servers. You can compare how this tool works with the other tools you used in previous practices. The following image depicts the architecture of the domain and tools used for this practice.



About This Course's Approach

This practice provides only guidelines for how to accomplish the objectives of the lesson. You take lessons learned from the slides or previous practices and apply them for this practice on your own.

Dependencies

This practice depends on:

- Practice 2-1 for setting up the course environment.
- Practice 2-3 for configuring Grinder and establishing initial baseline performance numbers of the Auction application in this environment.
- Java Development Kit version 1.7.0 update 40 or later. Mission Control and Flight Recorder features are not present before this release.
- The `setUserOverrides.sh` script in `/practices/tune/practice04-01/resources`. When you run the `setup.sh` script, it detects your existing `setUserOverrides.sh` script and ensures it is in place for you again. If you did not have an existing `setUserOverrides.sh` script in place, then it places the default file in the domain for your convenience.

Tasks

1. Connect to the **`host01`** and **`host02`** machines.
2. Set up the practice.
   a. Perform the following commands on **`host01`** to set up the environment for this practice. If your servers are still running from the previous practice, then you should skip running the `setup.sh` script.

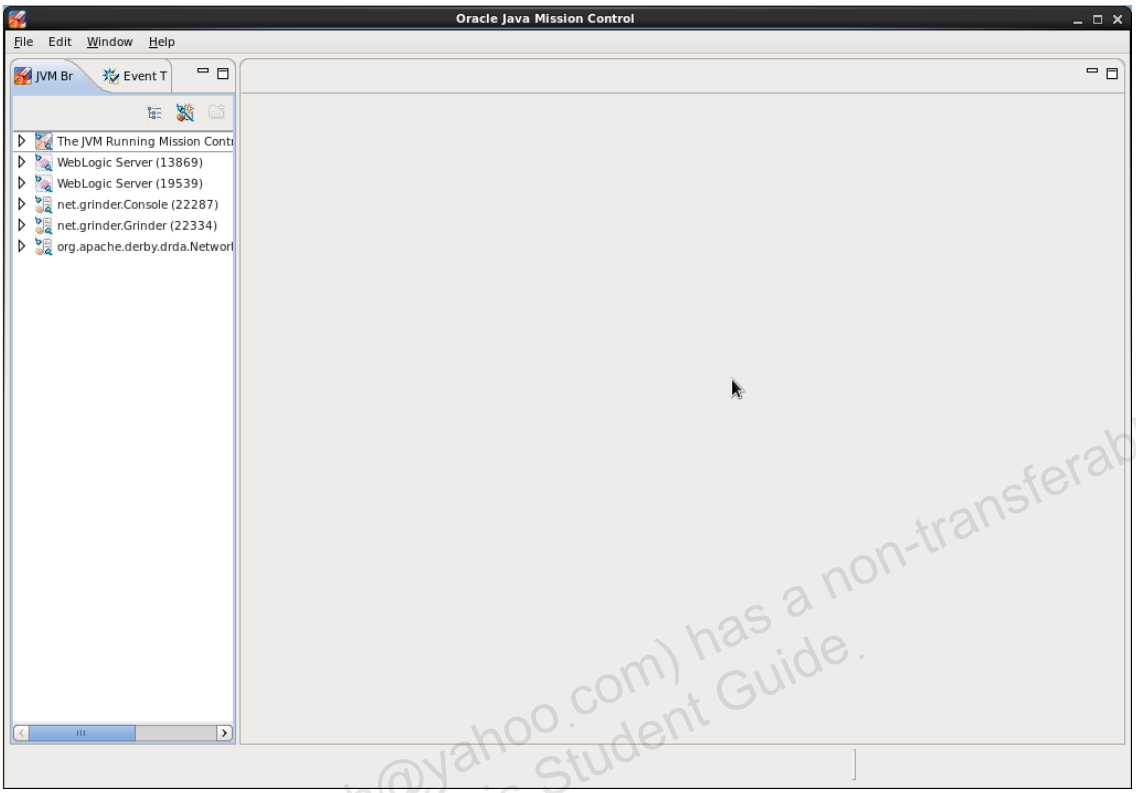
      ```
      $ cd /practices/tune/practice04-05
      $ ./setup.sh
      ```

      This script does the following:
      - Restores the domain to its original state. This includes your existing `setUserOverrides.sh` script.
      - Starts the wlsadmin AdminServer
      - Deploys the SimpleAuctionWebAppDb application to the cluster
      - Starts the server1 managed server on `host01`
      - Starts the server2 managed server on `host02`
3. Wait for all servers to log that they are in the RUNNING state before continuing.
4. Change your `setUserOverrides.sh` file again to turn on Mission Control and Flight Recorder by using the following parameters (memory settings should stay at 90 MB):

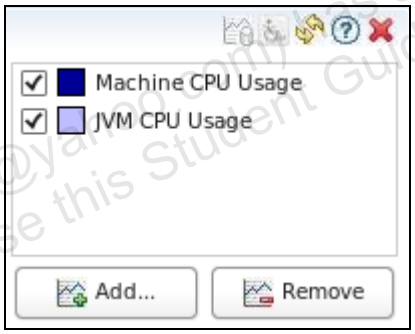
   ```
   -XX:+UnlockCommercialFeatures
   -XX:+FlightRecorder
   ```
5. Restart server1 to realize your changes.
6. Start Java Mission Control (JMC).
   a. The script to start JMC is located in `/u01/app/jdk/bin`. This directory is already in your path, so you can start JMC from any location. Execute the script to start the tool:

      ```
      $ jmc &
      ```

b. Verify that the GUI appears. You may have to close the initial Welcome page.



7. Connect to local WebLogic managed server with JMC.
   a. In the left-hand panel, right-click one of the WebLogic JVMs in the list and select *Start JMX Console*. This opens the JVM tab. Ensure that it is the server1 managed server JVM using the *System* tab on the bottom of the tool and find *VM Arguments* in the *Server Information* section of the screen. You can also find this information by checking `weblogic.Name` in the *System Properties* section. If it is not the server1 JVM, then close it and try another JVM.
   b. Review the descriptions of the JVM tabs in the following table:

Tab	Description
Overview 	Provides a graphical dashboard of gauges and charts that provide heap, CPU, fragmentation, and server information (in a separate tab on the bottom of the screen). You can use the icons in the upper right hand of the screen to manage and customize this page. Overview controls:  Dashboard controls:  Chart controls:  These controls allow you to view more specific data that is focused on your monitoring requirements. JMC provides a button to easily allow you to revert to the default settings.
MBean Browser 	Displays the MBean tree of the running server. Because you are connected to a managed server, you can only view runtime data. The default page displays the `java.lang.OperatingSystem` MBean, which displays important performance metrics for the running JVM.

Triggers	The Triggers tab enables you to configure threshold conditions for MBean metrics. You can also configure actions for JMC to take when these conditions are met, such as logging a message, dumping a Flight Recording, sending an email, and starting a Flight Recording. You can then constrain the condition to take place only during time periods. The tool enables you to import and export rules, and to perform CRUD operations on your own rules.
System	This tab displays server information, JVM statistics, and system properties by default. You can customize what is shown on this page.
Memory	This tab displays JVM garbage collection and memory statistics by default. You can customize what is shown on this page. This tab displays garbage collection statistics for the young and tenured generations. Metrics are displayed in a table format. This tab displays each of the memory heap spaces, the associated usage numbers, and graphical bars to show real time memory usage in the JVM much like VisualGC. The format is slightly different, but the information is similar.
Threads	This tab displays a live threads table with CPU, deadlock detection, and allocated bytes details per thread. It also displays stack traces for selected threads. You can customize what is shown on this page.

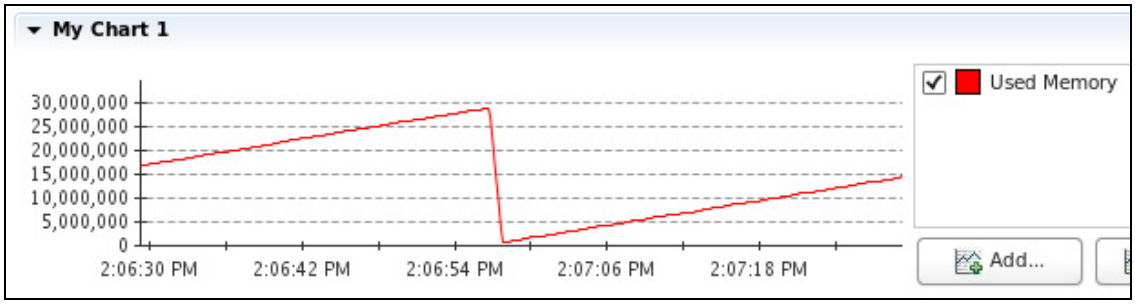
Overview Tasks

8. Use Mission Control to answer the following questions. You may run Grinder to load the application for some of these.
   a. What is the execution time average for the AuctionImageServlet? Where did you find this information in JMC?
   b. What is the current and maximum **JVM** CPU usage? Where is this information found?
9. Use Mission Control to perform the following tasks. You may use Grinder to load the application for some of these.
   a. Add a new dial on the Overview tab that shows the machine CPU utilization. Hint: The attribute is found in the Operating System node.
   b. Remove the Live Set and Fragmentation dial.
   c. Run Grinder and monitor the CPU, JVM CPU, heap, and garbage collection metrics by using JMC. What are the characteristics of these metrics? Are they the same as when you monitored the application using VisualVM?
   d. Try to create your own dial and graph. Select attributes of your choosing.
10. Reset the display to its default settings.

MBean Browser Tasks

11. Create a chart from an MBean attribute.
    a. Open java.lang > MemoryPool > Eden Space > Usage > used.
    b. Right-click the value and select Visualize.
    c. In the Create Chart dialog, click Add Chart.
       Note: You can add attributes to existing charts from this interface.

d. Click OK.
e. Click on the Overview tab and review the chart you just created.



12. Export data from an attribute.
    a. Return to the *Eden Space* attribute. Open java.lang -> MemoryPool -> Eden Space.
    b. Click any attribute in the display.
    c. Press Ctrl + A to highlight the entire table.
    d. Right-click the table and select *Clipboard Settings*.
    e. Make sure *Copy as CSV* is selected.
    f. Right-click the table and select Copy.
    g. Open a text editor and paste the data into it.
    h. Examine the data briefly.
13. View JVM options with the Operations tab.

    Within the MBean Browser tab, select an attribute and click the *Operations* tab to display available operations for the attribute.
    a. Open `com.sun.management`, choose the HotSpotDiagnostic MBean, and then choose `getVMOption` within the *Operations* tab.
    b. Enter the following as parameter 0 (p0): **ThreadStackSize**
    c. Click the Execute button. The value set for that option in the JVM should be displayed.
    d. Enter the following: **MaxTenuringThreshold**
    e. Click the Execute button. The value set for that option in the JVM should be displayed.

14. Set JVM options with the Operations tab.

    Note: This is an example of setting an attribute value using an MBean operation. There are many operations available for all different kinds of MBeans. Just remember that you cannot set, or write, to WebLogic Server runtime MBeans because they are read-only. All changes to WebLogic Server MBeans must occur through the WebLogic change management mechanism, where the configuration is locked, changes are made, and then changes are activated. However, you can freely change JVM values for the JVM running WebLogic to monitor and tune the JVM.

    a. Select `setVMOption` within the *Operations* tab.
    b. Enter the following values for p0 and p1 to dynamically enable printing GC data for this JVM: **PrintGC**, **true**
    c. Click Execute. The PrintGC option is now turned on.
    d. Switch to the server1 terminal window where server1 is running. You should now see GC data in the server's output window.
    e. Turn off PrintGC using the values: **PrintGC**, **false**

15. Turn on a notification for this JVM.
    a. Select `com.sun.management` > GarbageCollectionAggregator in the MBean Tree.
    b. Click the Notifications tab.
    c. Note that this particular MBean has two notification types.
    d. Check the Subscribe box.
    e. Wait for 2 or 3 events to occur. Depending on how your JVM is configured, this could take a couple of minutes.
    f. Deselect the Subscribe button.
    g. Examine the data that was collected.

16. Set Triggers for the JVM.
    a. Create a trigger for an attribute by clicking the Triggers tab. Choose an attribute that you know will hit a conditional threshold, such as CPU Usage.
    b. Configure a condition.
    c. Configure an action, but do not configure a constraint. Be sure to choose an action that you can verify in this environment, such as application alert, console output, or log to file. Note that if you select console output, it refers to the console where Mission Control was executed, not the server1 JVM. You may also want to set the *Limit Period* to something like 5 seconds so you can see the alerts appear without waiting a full 60 seconds for every notification.

Runtime Tasks

You can run Grinder during your exploration of the Runtime tab to make your results more interesting.

17. Explore each of the System, Memory, and Threads tabs on your own for a few minutes.
18. Enable CPU Profiling, Deadlock Detection, and Allocation in the Threads tab to see JMC populate all the columns.

Finishing Up

19. Leave the environment running for the next practice.
    The next practice uses Grinder and the same environment so it is best to leave it running.

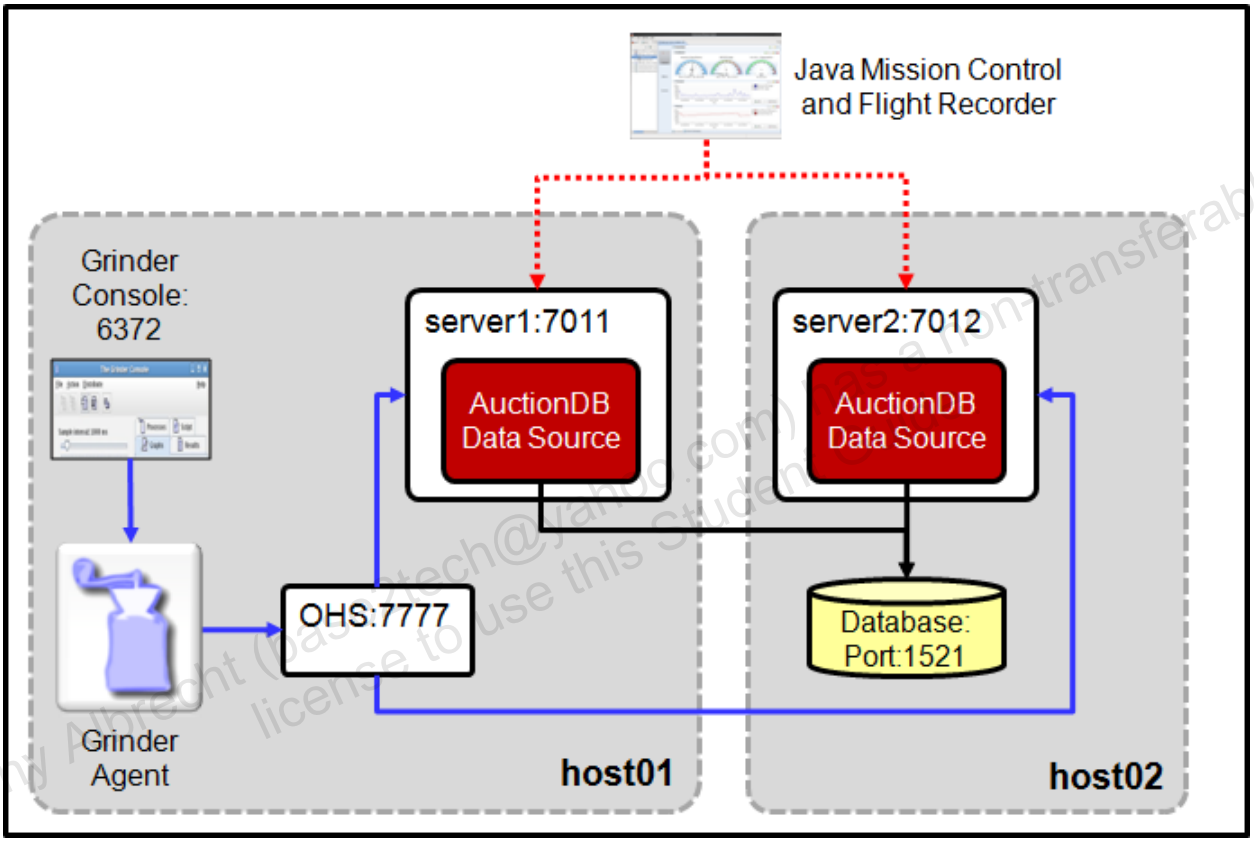
Practice Solution: Using Mission Control

There is no solution for this practice. Other practices in this lesson depend on the experience gained in this practice. Additionally, the experience gained in this practice can be applied in subsequent lessons.

Practice 4-6: Using Flight Recorder

Overview

The primary diagnostic tools that were part of the JRockit JVM implementation are now part of the HotSpot JVM. The next tools originating from JRockit that is now part of the HotSpot JVM is the Java Flight Recorder (formerly known as the JRockit Flight Recorder). In this practice, you use Java Flight Recorder (JFR) to monitor and analyze your WebLogic servers. You can compare how this tool works with the other tools you used in previous practices. The image below depicts the architecture of the domain and tools used for this practice.



About This Course's Approach

This practice provides only guidelines for how to accomplish the objectives of the lesson. You take lessons learned from the slides or previous practices and apply them for this practice on your own.

Dependencies

This practice depends on:

- Practice 2-1 for setting up the course environment.
- Practice 2-3 for configuring Grinder and establishing initial baseline performance numbers of the Auction application in this environment.
- Practice 4-5 for using Mission Control
- Java Development Kit version 1.7.0 update 40 or later. Mission Control and Flight Recorder features are not present before this release.
- The `setUserOverrides.sh` script in `/practices/tune/practice04-01/resources`. When you run the `setup.sh` script, it detects your existing `setUserOverrides.sh` script and ensures it is in place for you again. If you did not have an existing `setUserOverrides.sh` script in place, then it places the default file in the domain for your convenience.

Tasks

1. Connect to the **`host01`** and **`host02`** machines.
2. Set up the practice.
   a. Perform the following commands on **`host01`** to set up the environment for this practice. If your servers are still running from the previous practice, then you should skip running the `setup.sh` script.

      ```
      $ cd /practices/tune/practice04-06
      $ ./setup.sh
      ```

      This script does the following:
      - Restores the domain to its original state. This includes your existing `setUserOverrides.sh` script.
      - Starts the wlsadmin AdminServer
      - Deploys the SimpleAuctionWebAppDb application to the cluster
      - Starts the server1 managed server on host01
      - Starts the server2 managed server on host02
3. Wait for all servers to log that they are in the RUNNING state before continuing.
4. Configure and restart server1 if Mission Control and Flight Recorder have not been enabled.
   a. Change your `setUserOverrides.sh` file again to turn on Mission Control and Flight Recorder by using the parameters below (memory settings should be increased to 128MB to avoid out of memory exceptions that cause the recording to fail):

      ```
      -XX:+UnlockCommercialFeatures
      -XX:+FlightRecorder
      ```

   b. Restart server1 to realize your changes.

5. Start Mission Control and start Flight Recorder.
   a. If not started already, start Mission Control.
      ```
      $ jmc &
      ```
   b. Find the server1 managed server in the JVM browser.
   c. Expand the arrow next to the JVM to show the available options.
   d. Right click the Flight Recorder node and select *Start Flight Recording*. This causes the Start Flight Recording dialog window to appear. This is where you configure the settings for how you want Flight Recorder to work.
      Note: If you received an error, you may not have enabled the Flight Recorder features. Review step 4 to ensure you completed all the steps and try again.
   e. Name your recording.
   f. You have two options for recording:

Option	Description
Time fixed recording	Records JVM events for the specified period of time. This is useful for reproducible scenarios that you know will occur within a certain time interval.
Continuous recording	Records JVM events continuously for the duration of the JVM's lifecycle. You can truncate data by setting either a maximum size of the data file, or a maximum age for events stored in the data file. This is useful for constantly monitoring your JVM, and preserving a certain time period of events leading up to an issue.

   g. Select *Timed fixed recording*, and set a recording time of 2 min.
      Note: Because we have the heap set so low you can cause out of memory exceptions when running this recording. This is why we limit the recording to two minutes.
   h. Flight Recorder comes with two templates by default:

Template	Description
Continuous - on server	This template is tuned for very low performance overhead and is used for continuous production use. It incurs approximately 1% overhead.
Profiling - on server	This template provides a balance of information that is captured and performance overhead. It captures more data than the continuous template, but incurs approximately 2% overhead.

   i. Because you are running a time-based recording, select *Profiling - on server* as the *Event settings* value.
   j. Click Next. This displays a list of settings for this recording.
   k. Review each setting for a moment to learn what they do.

l. Set the *Exceptions* setting to *All exceptions, including errors*.
m. Check the *Heap Statistics*, *Class Loading*, and *Allocation Profiling* boxes.
n. Click Next.
o. Navigate to Java Application > Java Exception and ensure that the Enabled check box is checked.
p. Click Finish.

6. Use application and wait for recording to finish.
   a. Run Grinder with server1's heap set to 128MB to load the application with some traffic. Flight Recorder will record JVM events related to this traffic.
   b. Do not wait for Grinder to complete its load test. The five minute recording time may elapse before Grinder is done. While Grinder is running, you should see a progress meter in the Mission Control lower right-hand corner. Continue with the next step when you see the recording appear in Mission Control.
7. Explore the Flight Recording.
   a. When the recording process is complete, Mission Control downloads and opens the recording in its Event tab.
   b. You should see the following tab groups, or icons, along the left-hand side of the screen. Follow along in your Flight Recorder with each topic. Take a few moments to experiment with each topic and tab.

General	
Sub Tab	**Description**
Overview	Provides general high-level information, such as CPU and heap metrics
JVM Information	Provides information about the JVM that was recorded
System Properties	Provides all the system properties that are set for the JVM that was recorded
Recording	Provides a list of recordings and their event types and settings available to each recording

Memory	
Sub Tab	**Description**
Overview	Provides high-level memory usage, garbage collection, and thread allocation statistics
Garbage Collection	Provides a graphic and textual snapshot of garbage collection activity for the JVM that was recorded. Note that in the Heap graph that GC pause times are superimposed with heap use. The Heap and Garbage Collections graphs also have several tabs you can view.
GC Times	Provides high-level statistics about garbage collections that occurred in the JVM that was recorded. You can use the values here to see the longest GC pauses and in which spaces they occurred.
GC Configuration	Provides data for how the heap and generations are configured and sized.
Allocations	Provides information about objects that were allocated within the JVM, both in thread local allocation bytes (TLAB) and outside thread local space. You can use this to see how large and frequent allocations are in the JVM over time.
Object Statistics	Provides details about objects that are allocated in the heap during the selected time period, the top growing objects, and the percentage of heap they use

The Code tab group is more for developers that want to see how well their code is optimized and working within the server JVM.

Code (More for Developers)	
Sub Tab	**Description**
Overview	Provides high-level information about which packages and classes are spending the most time executing in the JVM
Hot Methods	Provides information about which methods are spending the most time executing in the JVM, and where they were called in the stack. Developers can use this information to isolate exactly where their code is taking the most time, and to see if there are ways to optimize for better performance.
Call Tree	Provides the most common sampled stack traces for the selected time period, which developers can use to find performance bottlenecks.
Exceptions	Provides information regarding exceptions thrown within the JVM. Developers can use the data in this tab to investigate the classes that throw the most exceptions, and see if there are any problems that cause excessive errors.
Compilations	Provides information about code generation changes that occurred during the recording. Developers can use the statistics in this page to see where the JIT compiler is working the most and see if their code is the most efficient for what it is doing.
Class Loading	Provides information about class loading and unloading. Developers can use this information to determine which classloaders are involved, and which classes are loaded and unloaded by each.

Threads	
Sub Tab	**Description**
Overview	Provides a high-level view of how CPU and threads are used by the JVM and the system.
Hot Threads	Provides a list of threads that spend the most time executing. You can review the stack trace for the thread to get more details about what it was executing at the time.
Contention	Provides information about threads that compete for a synchronization lock, blocked threads, and details about each thread.
Latencies	Provides information about when threads are not executing code. You can see how much time your threads are waiting, blocked, or sleeping. Note that if there is no load on the application that a lot of threads waiting is not unusual.
Thread Dumps	Shows the thread dumps that were captured during the selected recording period. You can use this to view the stack traces of JVM threads over a period of time to see how the application is functioning.
Lock Instances	Provides details about thread locking that occurred during the recorded time period. You can see the address and class of the synchronized object, and the amount of time the lock was in place.

I/O	
Sub Tab	**Description**
Overview	Provides information on file and socket reads and writes. You can use this information to see how I/O intensive your JVM is, and where I/O occurs the most.
File Read	Provides information on file reads that occurred during the selected recording time period. You can view file reads by thread, by event, over time, and related stack traces.
File Write	Provides information on file writes that occurred during the selected recording time period. You can view file writes by thread, by event, over time, and related stack traces.
Socket Read	Provides information on socket reads that occurred during the selected recording time period. You can view socket reads by thread, by event, over time, and related stack traces.
Socket Write	Provides information on socket writes that occurred during the selected recording time period. You can view socket writes by thread, by event, over time, and related stack traces.

System	
Sub Tab	**Description**
Overview	Provides information about the hardware where the JVM is running
Processes	Provides information for all processes in the system
Environment Variables	Provides a list of the environment variables that were set when the JVM was started. This is useful for when you want to investigate if there are issues with an incorrect environment for your JVM.

Notice that when you click the Events icon 

that the left-hand pane displays a tree of event types that you can include in the view of charts on the right-hand pane. Experiment with each of the settings on the left and see how they change the view on the right. At this point, you will not see any data associated with WebLogic Server because that data is captured either through WebLogic Diagnostic Framework (WLDF) notifications or opening a Flight Recorder recording contained in a WebLogic diagnostic image.

Events	
Sub Tab	**Description**
Overview	Provides a detailed breakdown of event data for a particular time period. You can select a period of time to analyze, see the producers of the events, and view details of the different event types.
Log	Provides details for events. You can sort by column data, Duration for example, to see which events took the longest to execute.
Graph	Provides activity graphically by which thread performed the work.
Threads	Provides performance metrics on a per thread basis.
Stack Traces	Provides stack traces for threads.
Histogram	Provides tools for analyzing events grouped by a certain property, which can also be used with an operative set of events with matching properties.

c. Now that you have followed along in Mission Control to explore these topics in this practice, take a few minutes to explore parts of the interface that interest you before going to the next step.

8. Capture and View WebLogic Server Flight Recorder data.

   You can make WebLogic take certain actions when a WLDF watch condition is reached, or if certain server scenarios take place, that will cause WebLogic to create a diagnostic image of the running server. This provides useful data that you can use to discover causes of performance problems that occur. You can also manually create a diagnostic image for analysis as shown here.

   a. Login to the WebLogic administration console.
   b. Navigate to Diagnostics > Diagnostic Images.
   c. Select server1 and click *Capture Image* to cause WebLogic to dump a diagnostic image of the server to disk.
   d. Specify the location and timeout values on the page and click OK to trigger the image creation process. This will dump data associated with the default WLDF Low built-in module that is configured.
   e. Within a terminal window, navigate to `/u01/domains/tune/wlsadmin/servers/server1/logs/diagnostic_images`.
   f. Locate the file with the `.jfr` extension. This is the Flight Recorder recording associated with your diagnostic image. A copy of it is also contained in the diagnostic image zip file, called `FlightRecorder.jfr`.
   g. Within Mission Control, select File > Open File and browse to this flight recording file to open it in the Mission Control tool.
   h. Click the Events icon to display the Event Types View in the left-hand pane.
   i. Check the WebLogic Server check box to display WebLogic Server events in the right-hand pane.
   j. Use the drag controls in the Overview tab to isolate a section of the recording time. The Interval, start, and end times change to reflect the selected time frame of the recording.
   k. The charts and tables on the page change to reflect the data for the selected time interval. You can view this data, and then pick and choose what event type data to include by using the left-hand pane. This enables you to correlate how certain WebLogic operations relate performance-wise to other metrics, such as CPU Load.
   l. Click the Log tab to show more details about events.
   m. Click the Duration column to see which events are taking the longest to execute.
   n. The *Execution Context Identifier* (ECID) is an internal representation of related requests within Oracle products. You can create an operative set within Flight Recorder to show only the events related to certain data, such as an ECID value. Right-click an event entry of Event Type *Servlet Invocation* in the *Event Log* section and select *Operative Set* > *Add Related Events* > *With ECID*=<*value*> to create an operative set.
   o. Check the *Show Only Operative Set* check box to display those events by themselves. This allows you to see where the most time is spent for certain operations. Again, this may be more developer focused, because it drills down to the code level, but there may be times that this provides information that is useful to administrators as well.
   p. Note the data contained in the *Event Attributes* section. If one of your operative sets has a *Pool* attribute, it should be a database connection pool name. If it is, then you should also be able to see the SQL statement associated with the request. This can help with performance problems where database requests are taking too long. You can

find the long running requests in WebLogic, then track it down to the database call, and uncover which SQL statement is taking a long time to execute. Then database administrators (if this is not also you) can investigate those queries and tables to see where a problem may exist. Perhaps a table wasn't indexed properly? This is one way to identify the issue and the cause faster.

q. Take a few minutes and experiment with the settings on this page and the other tabs.

9. Experiment with Flight Recorder commands.
   a. Review the commands in the following table and use them for the following step.

Command	Description
`jcmd pid JFR.start [options]`	Start a Flight Recorder recording from the command line
`jcmd pid JFR.check [options]`	Check the status of recordings for the specified process ID
`jcmd pid JFR.stop [options]`	Stop a recording with a specific identification number. Recording 1 is stopped by default.
`jcmd pid JFR.dump [options]`	Dump the data collected so far by the recording with a specific number. Recording 1 is dumped by default.

   The documentation reference page for these diagnostic commands is located in `/practices/tune/practice04-06/resources/app_command_ref.html`. Reference it to experiment with these commands.

   Note: You can also configure the JVM to create a recording when the JVM exits. This is useful troubleshooting issues that cause the server to crash. Additionally, you can configure conditional triggers in Mission Control that cause a flight recording to get created when the condition is met. This is useful for capturing data that leads up to an issue, while simultaneously minimizing the amount of data that gets recorded.

10. Close all Grinder applications, JVM tools, and terminal windows.

Course Developer Corner:

You may have noticed that the Flight Recorder and Mission Control tools both offer loads of settings and features. Each tool can probably support a course of their own, and unfortunately, we are really only able to show the high-level features of the products.

Mission Control provides several of the same features as the other graphical tools. You may have also noticed that it combines a lot of features within a single program and interface, and provides more features by default. When you are monitoring your WebLogic environments, the tools you use will be up to you. There are many tools available to monitor your JVMs. Covering them all goes beyond the scope of this course.

Practice Solution: Using Flight Recorder

There is no solution for this practice. Although no other practices depend on this lesson, the experience gained can be applied in subsequent lessons.