

Lab Guide

Oracle WebLogic 12c for Administrators

Version 1.0

Christopher Parent, Instructor
Copyright 2015 LearnWebLogicOnline.com

Table of Contents

Lab Guide Overview	3
Lab #1: Installing WebLogic.....	6
Lab #2: Creating a Domain	13
Lab #3: Admin Server	16
Lab #4: Managed Servers	19
Lab #5: Data Sources	22
Lab #6: JMS Resources	25
Lab #7: Application Deployment	28
Lab #8: Production Deployment.....	31
Lab #9: WebLogic Security	37
Lab #10: Clusters	45
Lab #11: Node Manager	51
Lab #12: Distributed Deployment	58
Lab #13: WLST	64
References.....	74

Lab Guide Overview

This lab guide will teach you how to perform common WebLogic administrative tasks using a variety of methods and tools. Each lab builds upon the previous, so it is highly recommended you perform all the labs in sequence.

The References section of this document contains links to the software needed for this lab, as well as links to online documentation for reference.

Organization

Each lab is organized in the following structure:

- **Skills Learned** describes what you will get out of the lab
- **Overview** describes the overall details of the lab
- **Configuration Parameters** defines parameters and values you will need to perform the lab
- **Instructions** provide the details steps needed to perform the lab

Formatting Convention

The lab uses various text formatting to convey meaning and action:

- **Bold** font will be used to emphasize actions to be taken:
 - **Click** the button
 - **Open** a terminal window
 - **Navigate to Environment > Servers**
- Courier font will be used to emphasize command syntax and command line execution
 - `cd <LAB_HOME>/domains/demo_domain`

File and Path Naming Convention

Oracle WebLogic is supported on Linux, Windows, and even OS X to a degree. This lab will work on all these systems. WebLogic for Windows includes .cmd or .bat versions of any scripts, whereas OSX and Linux include bash shell versions.

For the purposes of this lab, filenames and file paths will be referenced UNIX style in this document. If there are Windows-specific instructions for performing a particular task, they will be called out separately.

Lab Guide Directory Structure

This lab guide defines a standard set of directories for supporting the installation of WebLogic and the creation of WebLogic domains. It is highly recommended to use the directories outlined in Table 1 Lab Directories as the lab will reference these directories using its reference name and not the absolute path. You will see directory references throughout the lab in the following form:

*Navigate to **<DOMAIN_HOME>/bin** and execute **startWebLogic.sh***

Wherever you see this variable notation, either for navigating a file system or specifying a file or directory path as part of some WebLogic resource configuration, be sure to use the fully qualified path and not the variable notation.

Table 1 Lab Directories

Directory Reference	Path
LAB_HOME	/u01/udemy or c:\udemy
ORACLE_ROOT	<LAB_HOME>/Oracle
ORACLE_HOME	/<LAB_HOME>/Oracle/Middleware/Oracle_Home
JAVA_HOME	Where JDK7 is installed. Varies by system.
DOMAIN_ROOT	<LAB_HOME>/domains
DOMAIN_HOME	<DOMAIN_ROOT>/<domain_name>
APPLICATION_ROOT	<LAB_HOME>/applications
DERBY_HOME	<ORACLE_HOME>/wlserver/common/derby

Hardware Requirements

This lab requires a computer with sufficient memory and storage for installing WebLogic and running 2 or more WebLogic server instances. Suggested minimum requirements:

- 4GB ram, 6GB recommended
- 3GB disk space

Software Requirements

The following software is required:

- Oracle WebLogic Server 12.1.3. Generic Installer
- JDK 1.7.x
- Any text editor (notepad, vi, TextMate, etc)

Links to required software can be found in the References section of this lab guide.

Need Help?

If you need help with the labs or have questions please visit www.learnweblogiconline.com and contact me directly.

Lab #1: Installing WebLogic

Duration 30 minutes

Skills Learned

At the end of this exercise, you will be able to:

- Install Oracle WebLogic 12c
- Create 3 sample Domains
- Install and deinstall WebLogic silently

Overview

In this lab you will download and install both JDK 7 and Oracle WebLogic 12c Generic Installer. The lab will cover both interactive and silent installs of WebLogic. You will also prepare your environment for all the upcoming labs.

Configuration Parameters

Table 2 LAB_HOME Directory

Directory	OS
/u01/udemy	Linux, Unix, Mac
C:\udemy	Windows

Table 3 Generic Installer Parameters

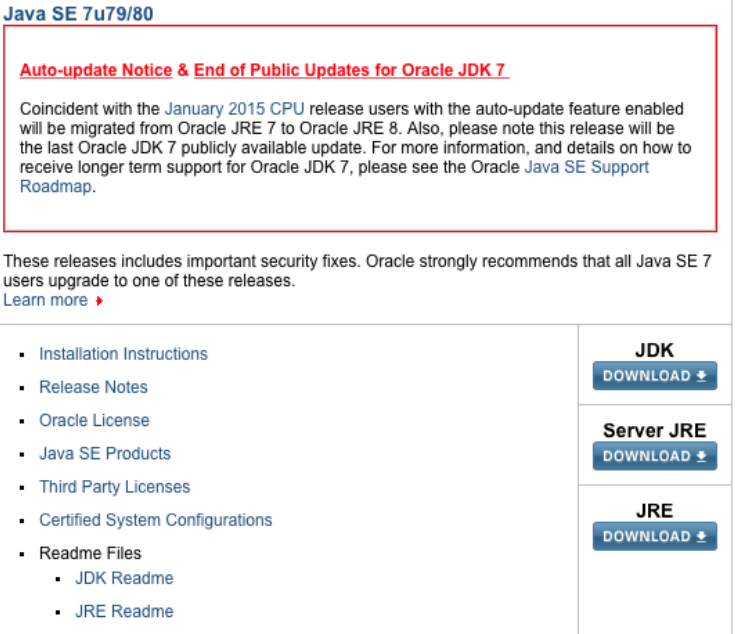
Parameter	Value
Inventory Directory	<LAB_HOME>/oraInventory
Operating System Group	Use default value
Oracle_Home	<ORACLE_ROOT>/Middleware/Oracle_Home

Table 4 Quickstart Configuration Wizard

Parameter	Value
Name	weblogic
Password	weblogic123
Domain Root Location	<LAB_HOME>/domains
Application Root Location	<LAB_HOME>/applications

Instructions

1	Preparing your Environment
1.1	Create your <LAB_HOME> directory for this Lab Guide according to Table 2
2	Installing JDK 7

2.1	<p>Download the latest JDK 7 from oracle.com. See the Appendix for a reference URL.</p>  <p>These releases includes important security fixes. Oracle strongly recommends that all Java SE 7 users upgrade to one of these releases. Learn more ▶</p>
2.2	<p>Follow these instructions for installing JDK7 on your system</p> <p>http://docs.oracle.com/javase/7/docs/webnotes/install/index.html</p>
2.3	<p>Once JDK7 is installed, make note of where JDK7 is installed. We will need to reference JAVA_HOME when we install Oracle WebLogic 12c.</p>
3	Installing Oracle WebLogic 12c using the Generic Installer
3.1	<p>Download Oracle WebLogic 12c (12.1.3) Generic Installer from oracle.com. See the Appendix for a URL.</p>
3.2	<p>After the generic installer has been downloaded, copy the installer to <LAB_HOME></p>
3.3	<p>Run the installer by executing the following command:</p> <pre><JAVA_HOME>/bin/java -jar <LAB_HOME>/fmw_12.1.3.0.0_wls.jar</pre> <p>The generic installer will now launch.</p>
3.4	<p>On the Installation Inventory Setup screen, specify oraInventory according to Table 3</p> <p>Click Ok</p>
3.5	<p>The Welcome screen for Oracle Fusion Middleware will appear. Click Next.</p>
3.6	<p>On the Installation Location screen specify Oracle Home according to Table 3</p>

3.7	On the Installation Type screen, select 'Complete with Examples' and click Next .
3.8	<p>The Installation Summary screen summarizes the installation location, required disk space and the software components that will be installed.</p> <p>In order to perform silent installs of WebLogic in the future, a response file is needed. The simplest way to create a response file is to have the WebLogic installer create one for you.</p> <p>Click the Save Response File button and save the response file to:</p> <pre><LAB_HOME>/wl_install_response.rsp</pre> <p>You will reference this response file later in this lab to perform a silent install.</p> <p>Back on the Installation Summary screen, click Install.</p>
3.9	On the Installation Progress screen, click Next .
3.10	On the Installation Complete screen, enable 'Automatically Launch the Quickstart Configuration Wizard' and click Finish .
4	Quickstart Configuration Wizard
4.1	Specify the values in Table 4 and accept all other defaults and click Create.
4.2	Quickstart will create 3 example WebLogic domains. When Progress is 100% click Next .
4.3	The Configuration Details page will provide details about each example domain, including its name, location, and admin console URL. Make note of this information and Click Finish .

Configuration Details

Domain Name : wl_server
 Domain Location : /u01/udemy/domains/wl_server
 Application Location : /u01/udemy/applications/wl_server
 Status : **Successful**
 Admin Server URL : <http://bugatti.local:7001/console>

Domain Name : medrec
 Domain Location : /u01/udemy/domains/medrec
 Application Location : /u01/udemy/applications/medrec
 Status : **Successful**
 Admin Server URL : <http://bugatti.local:7001/console>

Domain Name : medrec-spring
 Domain Location : /u01/udemy/domains/medrec-spring
 Application Location : /u01/udemy/applications/medrec-spring
 Status : **Successful**
 Admin Server URL : <http://bugatti.local:7001/console>

Next Step
☐ Start Domain wl_server

5	Performing a Silent WebLogic Install
5.1	<p>Oracle WebLogic 12c is the first release of WebLogic to be ported over to the Oracle Universal Installer, or OUI. OUI for WebLogic can be installed silently which eliminates the need to monitor the installation and no input from the user is required.</p> <p>Silent installs are a requirement when installation of a product is to be scripted or managed by a 3rd party product such as Puppet or Chef and user input is not possible.</p> <p>OUI performs silent installations through the use of response files. Response files define the various installation parameters for installing the product.</p> <p>The simplest way to create a response file if you do not have access to one already, is to generate one by running through the regular WebLogic graphical install as you did in the first part of this lab.</p> <p>During the graphical installer, you should have selected the option to save the response file to:</p> <pre><LAB_HOME>/wlinstall_response.rsp</pre> <p>Locate that response file now and open it up in a text editor.</p>
5.2	<p>Notice the response file is very short and only defines a few parameters. The two that you are interested in are ORACLE_HOME and INSTALL_TYPE.</p> <p>Modify ORACLE_HOME to point to a different location so we do not</p>

	<p>overwrite our previous work.</p> <p>Set ORACLE_HOME to:</p> <pre><LAB_HOME>/silent/Oracle_Home</pre> <p>We also do not want to deploy the examples either, so change the INSTALL_TYPE to:</p> <pre>WebLogic Server</pre> <p>Save the file and exit the editor.</p>
5.3	<p>Now run OUI in silent mode using the following syntax:</p> <pre>java -jar <LAB_HOME>/fmw_12.1.3.0.0_wls.jar -silent -responseFile <LAB_HOME>/wlinstall_response.rsp</pre>
5.4	<p>After you have executed the silent install command, OUI will exit with a message stating that <i>you are starting your first installation on this host or that you do not have sufficient permissions to access current inventory</i>.</p> <p>The Oracle Inventory directory on a host is used to store installation files for various Oracle products. Usually in production systems the oraInst.loc file is owned by root and located under /var/opt, however you can also create a private local oracle inventory, typically only used in development or QA environments.</p> <p>For the purposes of this lab we will create a private inventory location.</p>
5.5	<p>Create the following sub directory for the silent install files:</p> <pre><LAB_HOME>/silent</pre>
5.6	<p>Using a text editor create the following file:</p> <pre><LAB_HOME>/silent/createPrivateInventory.txt</pre> <p>and specify the following two lines in the file:</p> <pre>inventory_loc=<LAB_HOME>/silent/privOraInv.loc inst_group=<group_name></pre> <p>where group_name is the operating system group that the install user is a member of.</p> <p>If you are not sure what group to put and you are performing this task in Linux or Unix, type 'groups' in a terminal window and use the first group that is returned.</p>

	Save the text file and exit the editor.
5.7	<p>Now run the silent OUI again but this time specify the private inventory flag:</p> <pre>java -jar <LAB_HOME>/fmw_12.1.3.0.0_wls.jar -silent -responseFile <LAB_HOME>/wlinstall_response.rsp - invPtrLoc <LAB_HOME>/silent/createPrivateInventory.txt</pre>
5.8	<p>You will know the installation is complete when you see the following message:</p> <p>You can find the log of this install session at: /private/var/folders/ps/dwqs1btd2ls7hd0y0cnlfd9w0000gn/T/OraInstall2015-05-04_09-08-03PM/install2015-05-04_09-08-03PM.log ----- ---20%-----40%-----60%-----80%-----100%</p> <p>The installation of Oracle Fusion Middleware 12c WebLogic Server and Coherence 12.1.3.0.0 completed successfully. Logs successfully copied to /u01/udemy/silent/privOraInv.loc/logs.</p> <p>Please note the location of the log files will vary depending on your system.</p>
6	Performing a Silent Uninstall of WebLogic
6.1	<p>Now that you know how to perform a silent install of WebLogic, you will now uninstall WebLogic using the silent installer.</p>
6.2	<p>Run the following command to silently uninstall WebLogic:</p> <pre><LAB_HOME>/silent/Oracle_Home/oui/bin/deinstall.sh - silent -responseFile <LAB_HOME>/wlinstall_response.rsp</pre> <p>Note that there are other de-install options if you have multiple Oracle products installed.</p> <p>You should see similar output if the deinstallation is successful:</p> <pre>Copyright (c) 1996, 2014, Oracle and/or its affiliates. All rights reserved. Reading response file.. Starting silent deinstallation... -----20%-----40%-----60%-----80%----- ---100%</pre> <p>The uninstall of WebLogic Server 12.1.3.0.0 completed successfully. Logs successfully copied to /u01/udemy/silent/privOraInv.loc/logs.</p>

--	--

Lab #2: Creating a Domain

Duration 15 minutes

Skills Learned

At the end of this exercise, you will be able to:

- Create a WebLogic domain using Configuration Wizard
- Create a template using Template Builder
- Create a WebLogic domain using WLST

Overview

In this lab you will learn how to create a simple domain using the Configuration Wizard, a UI wizard-driven tool that allows you to specify a limited number of configuration items. You will then use the Template Builder to create a template of your domain. The template is an image of your configure domain that allows you to recreate a domain with preconfigured settings.

Lastly you will be introduced to WebLogic Scripting Tool where you will create a domain from the domain template using a series of WLST commands.

Configuration Parameters

Table 5 Configuration Wizard Parameters

Parameter	Value
Domain Location	<DOMAIN_ROOT>/lab_domain
Domain username	weblogic
Domain password	weblogic123
Domain Mode	Production
JDK	<JAVA_HOME>

Table 6 Create Domain Template Parameters

Parameter	Value
Create Domain Template	Checked
Use Domain Source	Checked
Source Location	<DOMAIN_ROOT>/demo_domain
Template Location	<LAB_HOME>/demo_template.jar

Table 7 createDomain Arguments

Parameter	Value
domainTemplate	<LAB_HOME/demo_template.jar
domainDir	<DOMAIN_ROOT>/template_domain
user	weblogic
password	weblogic123

Instructions

1	Create a WebLogic domain with Config Wizard
1.1	Start the WebLogic Configuration Wizard by running the following script: <pre><ORACLE_HOME>/wlserver/common/bin/config.sh</pre>
1.2	Select Create a new domain
1.3	Specify location of the domain then click Next . See Table 5.
1.4	Select the template Basic WebLogic Server Domain and click Next
1.5	Specify the domain username and password according to Table 5 and click Next
1.6	Specify domain mode and JDK according to Table 5
1.7	On the Advanced Configuration Screen, disable all checkboxes and click Next
1.8	On the Configuration Summary screen, click Create
1.9	On the Configuration Progress screen, click Next as you see the Domain Created Successfully! Message
1.10	On the Configuration Success page, make note of the Domain Location (known as DOMAIN_HOME) and the Admin Server URL. This will be used to log into the admin console.
2	Verify Domain Creation
2.1	Open a terminal window or command prompt and navigate to <DOMAIN_HOME>
2.2	List the files and directories underneath <DOMAIN_HOME> . Navigate around the directory structure, specifically config/ and servers/
3	Create a template using Domain Template Builder
3.1	In this section you will use the Domain Template Builder to create template of the domain you just built. Open a terminal window and launch the Domain Template Builder script: <pre><ORACLE_HOME>/oracle_common/bin/config_builder.sh</pre>
3.2	On the Create Domain Template screen, specify the values according to Table 6 Create Domain Template Parameters and click Next.
3.3	Use the default values on the Template Information screen and click Next.
3.4	The Domain Content screen allows you to add files to the template, such as

	SQL initialization scripts, custom shell scripts, property files, etc.
	Click Next on the Domain Content screen.
3.5	<p>The Scripts and Files screen allows you to parameterize the contents of files using pre-defined WebLogic variables.</p> <p>For instance, select the startWebLogic.sh script under Domain Contents > Domain Root Directory and click the Edit button.</p> <p>In the Edit File window, notice the value of DOMAIN_HOME is set to "@DOMAIN_HOME"</p> <p>@DOMAIN_HOME is a variable that will be defined during domain creation.</p> <p>WebLogic comes with many other pre-defined variables. See the online documentation for Domain Template Builder for more details.</p>
3.6	Click Next on the Scripts and Files screen.
3.7	On the Summary screen click Create .
3.8	Once the Configuration Process reaches 100% click Next .
3.9	On the Template Creation Success page, observe the template location then click Finish .
4	Create a Domain from a Template using WLST
4.1	In this section you will now create a domain using the template you just built. You could use the Configuration Wizard to create the domain, but this section will use WLST to script the creation.
4.2	<p>In a terminal window, launch WLST using the script:</p> <pre><ORACLE_HOME>/wlserver/common/bin/wlst.sh</pre>
4.3	<p>The createDomain WLST command will be used to create a domain using your template.</p> <p>Type help('createDomain') to get information about the command.</p>
4.4	Create a domain using the template according to Table 7 createDomain Arguments.
4.5	<p>When the command completes successfully you will not see any output.</p> <p>Exit WLST by typing exit()</p>
4.6	Verify the domain was creating by navigating to <DOMAIN_ROOT>/template_domain and take a look at the contents of the directory.

Lab #3: Admin Server

Duration 30 minutes

Skills Learned

At the end of this exercise, you will be able to:

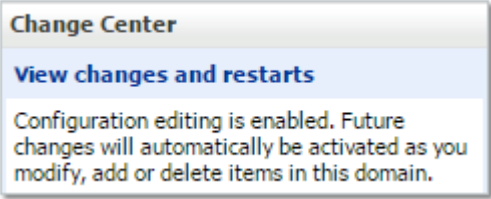

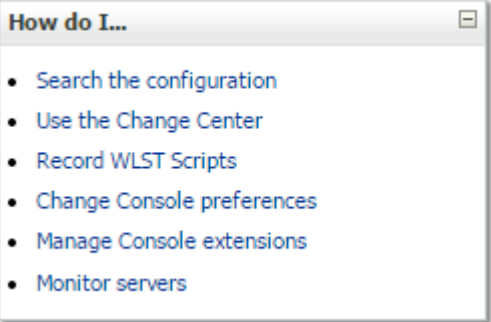
- Start the admin server
- Log into the admin console

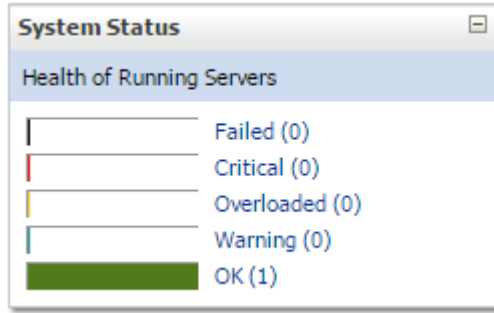
Overview

In this lab you will learn how to start the admin server and log into the Admin Console. This lab will walk through the user interface and explain its various functions.

Instructions

1	Start the admin server
1.1	Open a terminal window or command prompt
1.2	Navigate to <DOMAIN_ROOT>/demo_domain
1.3	Execute the following script <code>./startWebLogic.sh</code>
1.4	Once you execute the startWebLogic script you will begin to see output messages from the admin server. You will know when the admin server is up and running when you see the message "The server started in RUNNING mode." ####<Apr 28, 2015 12:21:56 PM MDT> <Notice> <WebLogicServer> <bugatti.local> <AdminServer> <[ACTIVE] ExecuteThread: '4' for queue: 'weblogic.kernel.Default (self-tuning)''> <<WLS Kernel>> <> <> <1430245316823> <BEA-000360> <The server started in RUNNING mode.>
2	Access the Admin Console
2.1	Log into the admin console at http://localhost:7001/console Use the domain username and password created during the previous lab. <ul style="list-style-type: none"> ▪ localhost assumes you have started the admin server on your local machine ▪ Port 7001 is the default port for the admin server. ▪ /console is the URN for the Admin Console application
2.2	The admin console home page contains several widgets or portlets. The following table describes these widgets.

Change Center	<p>The Change Center keeps track of all changes that have been made to a domain. It also serves as a check control system if production mode has been enabled. In production mode, use the Change Control center to obtain an exclusive lock in order to edit the domain. In production mode, changes to the domain are not committed until they are saved via the Change Center.</p>
 <p>Change Center</p> <p>View changes and restarts</p> <p>Configuration editing is enabled. Future changes will automatically be activated as you modify, add or delete items in this domain.</p>	<p>Every application and WebLogic service and resource is organized under the Domain Structure. It is here that you will access every configurable resource and setting, from data sources and JMS destinations to deployed applications and logging settings. Click the (+) to expand the Domain Structure.</p>
 <p>Domain Structure</p> <p>base_domain</p> <ul style="list-style-type: none">[-] Environment[-] Deployments[-] Services[-] Security Realms[-] Interoperability[-] Diagnostics	<p>The How do I... widget provides online help links to common tasks based on where you are within the admin console. If you are on the home page you will see the links included here, however if you are on the deployments page, you will see links specific to application deployment.</p>
 <p>How do I...</p> <ul style="list-style-type: none">• Search the configuration• Use the Change Center• Record WLST Scripts• Change Console preferences• Manage Console extensions• Monitor servers	



The System Status provides a snapshot of the health of all WebLogic servers including their states.

Lab #4: Managed Servers

Duration 30 minutes

Skills Learned

At the end of this exercise, you will be able to:

- Create and start a managed server
- Create a boot.properties file

Overview

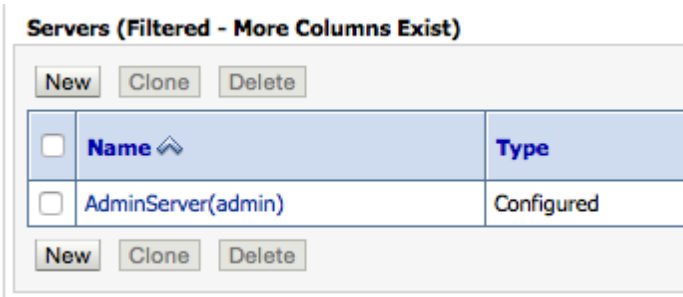
In this lab you will create a managed server using the admin console. You'll then start this managed server and monitor it via the admin console.

Configuration Parameters

Table 8 Managed server parameters

Parameter	Value
Server Name	ms1
Server Listen Address	<Leave blank>
Server Listen Port	8001
Should this server belong to a cluster	No, this is a stand-alone server

Instructions

1	Create a managed server
1.1	Start the admin server if not already running
1.2	Log into the admin console
1.3	Under Domain Structure widget, navigate to Environment > Servers
1.4	Click New to create a new managed server 
1.5	Specify the managed server parameters from Table 8 then click Finish
1.6	Verify your new managed server shows up in the Servers table.

Servers (Filtered - More Columns Exist)

New Clone Delete

<input type="checkbox"/>	Name	Type	Cluster	Ma
<input type="checkbox"/>	AdminServer(admin)	Configured		
<input type="checkbox"/>	ms1	Configured		

New Clone Delete

2 Start the Managed Server

2.1 Open a new terminal window or command prompt and navigate to **<DOMAIN_HOME>/bin**

2.2 Start the managed server using the startManagedWebLogic start script.

The syntax for starting a managed server is:

```
startManagedWebLogic.sh <managed_server_name>
t3://<admin_server_host>:<admin_port>
```

where:

- **<managed_server_name>** is the name of the managed server you want to start
- **<admin_server_host>** is the hostname of where the admin server is running
- **<admin_port>** is the port the admin server is listening on

If you have followed this lab exactly, the exact syntax should be:

```
./startManagedWebLogic.sh ms1 t3://localhost:7001
```

2.3 You will be prompted to enter a username and password to boot ms1. **Use** the credentials you defined when creating the domain.

2.4 Once the managed server is RUNNING, **log** back into the admin console and navigate to **Environment > Servers** and verify ms1 is **RUNNING**

Servers (Filtered - More Columns Exist)

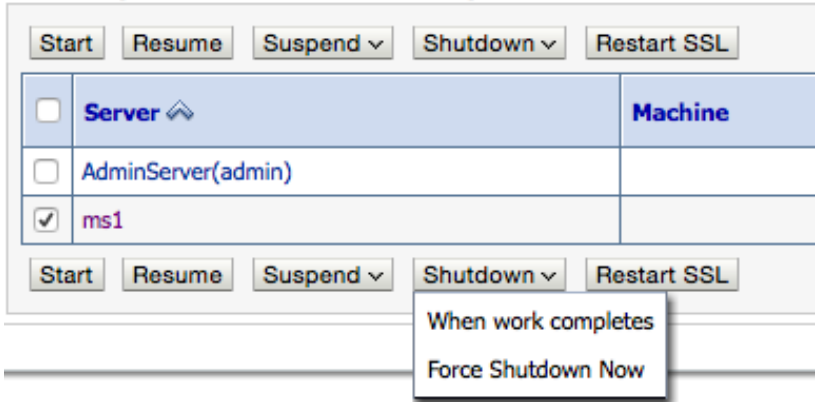
New Clone Delete Showing 1 to 2 of 2 Previous | Next

<input type="checkbox"/>	Name	Type	Cluster	Machine	State	Health	Listen Port
<input type="checkbox"/>	AdminServer(admin)	Configured			RUNNING	✓ OK	7001
<input type="checkbox"/>	ms1	Configured			RUNNING	✓ OK	8001

New Clone Delete Showing 1 to 2 of 2 Previous | Next

3 Configure Managed Server Boot Up

3.1 To prevent having to enter a username and password when starting a managed server we must create a property file with the username and

	<p>password in it.</p> <p>Each WebLogic server has its own boot.properties file which defines the username and password for booting that server. The username and password are entered in plaintext, however WebLogic will encrypt the values once the server has started.</p> <p>Create the following directory if it does not exist:</p> <p><DOMAIN_HOME>/servers/ms1/security</p>
3.2	<p>Create the boot.properties file in ms1/security with the following lines:</p> <pre>username=<admin username> password=<admin password></pre> <p>Save the file</p>
3.3	<p>Stop ms1 from the Admin Console. Navigate to Environment > Servers and click on the Control tab.</p> <p>Check the box next to ms1 and click Shutdown > Force Shutdown Now</p> <p>Servers (Filtered - More Columns Exist)</p> 
3.4	<p>Restart ms1 using startManagedWebLogic script. If you created the boot.properties correctly, you will not be prompted for the username and password.</p>
3.5	<p>Open another terminal window and take a look at the contents of the boot.properties now. You should see the values are encrypted.</p> <p>If you ever change the credentials for the admin user, you must update the boot.properties file using plaintext values.</p>

Lab #5: Data Sources

Duration 60 minutes

Skills Learned

At the end of this exercise, you will be able to:

- Start Derby DB
- Configure generic data source for Derby
- Test and monitor a data source

Overview

In this lab you will use Apache's open source Derby database to create a database instance. You will then use the Admin Console to create a generic data source for Derby.

Configuration Parameters

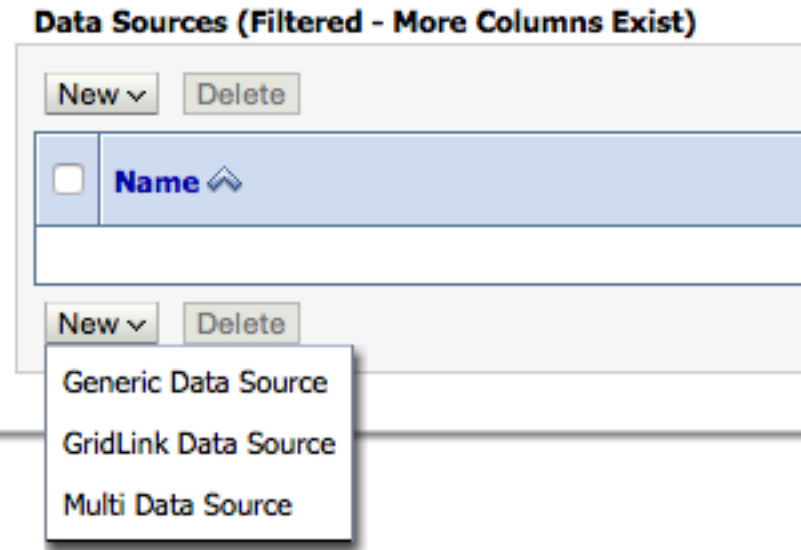
Table 9 Data source properties

Parameter	Value
Name	derbyDS
JNDI Name	demo.derbyDS
Database Type	Derby
Database Name	demodb
Host Name	localhost
Port	1527
Database User Name	<blank>
Password	<blank>

Instructions

1	Start Derby Database
1.1	Open a new terminal window
1.2	<p>Set environment variable JAVA_HOME to point to JDK 7. You will need to do this each time you open a new terminal window and want to run Derby. It is recommended to modify the Derby Network script to specify JAVA_HOME or specify this in your shell profile or Windows profile.</p> <p>On Linux systems:</p> <pre>export JAVA_HOME=<path_to_jdk7></pre>

	<p>On Windows:</p> <pre>set JAVA_HOME=<path_to_jdk7></pre>
1.3	<p>Start the Derby DB server by executing the script:</p> <pre><DERBY_HOME>/bin/startNetworkServer</pre> <p>Derby will start and listen on port 1527.</p>
2	Create a Derby Database Instance
2.1	<p>Open another terminal window and start the Derby interactive interface:</p> <pre>java -jar <DERBY_HOME>/lib/derbyrun.jar ij</pre>
2.2	<p>Create a new Derby database instance named demodb by issuing the following command:</p> <pre>CONNECT 'jdbc:derby://localhost:1527/demodb;create=true';</pre>
2.3	<p>Exit the interactive Derby shell by typing:</p> <pre>exit;</pre>
3	Load the JDBC Drivers
3.1	<p>Most database vendors include JDBC drivers in jar files. In order to configure a Derby data source in WebLogic, we need to tell WebLogic where to find these JDBC drivers. We do this by placing the jar file containing the JDBC drivers on WebLogic's classpath.</p> <p>There are several ways to do this. The simplest and most practical way is to place the JDBC jar file in your domain's lib directory. Jar files placed under <DOMAIN_HOME>/lib automatically get loaded onto the server's CLASSPATH.</p> <p>Copy <DERBY_HOME>/lib/derbyclient.jar to <DOMAIN_HOME>/lib</p>
3.2	<p>You must now restart the admin server in order for the classpath to be updated with the new JDBC driver.</p> <p>Stop the admin server using the stopWebLogic.sh script that can be found under <DOMAIN_HOME>/bin</p>
3.3	Restart the admin server
4	Create the Data Source
4.1	Log into the admin console
4.2	Under Domain Structure, navigate to Services > Data Sources
4.3	Click New > Generic Data Source

	
4.4	Configure the JDBC Data Source properties according to Table 9 then click Next
4.5	Select Derby's Driver (Type 4) Versions: Any from the driver drop down list and click Next
4.6	Accept all defaults under Transaction Options and click Next
4.7	Configure the connection properties according to Table 9 and click Next
4.8	<p>On the Test Database Connection screen, click the Test Configuration button near the bottom of the page to verify the data source can connect to the demo DB.</p> <p>You should receive a Connection Successful message.</p> <p>Click Next</p>
4.9	Target this data source to the AdminServer and click Finish .
5	Monitor Data Sources
5.1	Click on the data source in the data sources table then click on the Monitoring tab
5.2	Observe the various data source monitoring and diagnostic information

Lab #6: JMS Resources

Duration 45 minutes

Skills Learned

At the end of this exercise, you will be able to:

- Create a JMS server, Connection Factory, and Queue
- Create a file-based persistent storage
- Publish a JMS message
- Monitor JMS statistics

Overview

In this lab you will create resources necessary to support a JMS producer and consume. You will create a JMS server, connection factory, and queue. You will learn about JMS resource targeting through sub-deployments. You will also use the admin console to send a JMS message.

Configuration Parameters

Table 10 File Store Parameters

Parameter	Value
Name	demoFS
Target	AdminServer
Directory	<LAB_HOME>/filestore/AdminServer

Table 11 JMS Server Parameters

Parameter	Value
Name	demoJMSServer
Persistent Store	demoFS

Table 12 Queue Parameters

Parameter	Value
Name	demoQueue
JNDI Name	test.demoQueue
Subdeployment	demoQueue

Table 13 JMS Message Parameters

Parameter	Value
Type	text
Body	This can be anything. Enter any text here.

Table 14 Connection Factory Parameters

Parameter	Value
Name	demoConnFactory
JNDI Name	test.demoCF
Subscription Sharing Policy	Exclusive
Client ID Policy	Restricted
Maximum Messages per Session	10
XA Connection Factory Enabled	Enabled

Instructions

1	Create File Store
1.1	<p>Persistent storage is used to store JMS messages until they can be consumed. WebLogic supports both file-based and DB-based storage options. In this lab we will create a file-based persistent store.</p> <p>First, create a directory on the file system:</p> <pre><LAB_HOME>/filestore</pre>
1.2	Make sure the admin server is running and log into the admin console
1.3	Under Domain Structure, navigate to Services > Persistent Stores
1.4	Click New > Create FileStore
1.5	Configure the file store according to Table 10 then click Next
2	Create JMS Server
2.1	Navigate to Services > Messaging > JMS Servers
2.2	Click New
2.3	Configure the JMS Server according to Table 11 then click Next
2.4	Target the JMS Server to AdminServer, then click Finish
3	Create JMS Module
3.1	Navigate to Services > Messaging > JMS Modules
3.2	Click New to create a new Module
3.3	Specify demoJMSModule for the name and click Next
3.4	Target the module to AdminServer and click Next
3.5	Click Finish
3.6	On the Summary of JMS Modules page, click on demoJMSModule
4	Create JMS Connection Factory
4.1	Navigate to Services > Messaging > JMS Modules
4.2	Click on demoJMSModule
4.3	Click New
4.4	Select Connection Factory
4.5	Configure the connection factory according to Table 14 then click Next
4.6	Target to AdminServer then click Finish
4.7	In the Summary of Resources table click on demoConnFactory
4.8	Click on the Default Delivery tab and note the various configuration and tuning parameters

4.9	Click on the Client tab and note its configuration parameters												
4.10	Click on all the remaining tabs and view the various configuration, tuning, and security settings												
5	Create a JMS Queue												
5.1	Navigate back to Services > Messaging > JMS Modules > demoJMSModule and click New												
5.2	Select Queue from the list and click Next												
5.3	Configure and target the queue according to Table 12 You will have to create a subdeployment that is targeted to the JMS server.												
5.4	Click Create a new Subdeployment Use the default value for the Subdeployment name and click Ok Click Finish On the Settings for demoJMSModule page, click the Subdeployments tab Click on the demoQueue subdeployment Target the subdeployment to the demoJMSServer and click Save Navigate back to Services > JMS Modules Click on demoJMSModule												
5.5	Your complete JMS queue should appear as follows: <div><div>Summary of Resources</div><div><div>NewDelete</div><div>Showing 1 to 1 of 1PreviousNext</div><table><thead><tr><th><input type="checkbox"/></th><th>Name ↕</th><th>Type</th><th>JNDI Name</th><th>Subdeployment</th><th>Targets</th></tr></thead><tbody><tr><td><input type="checkbox"/></td><td>demoQueue</td><td>Queue</td><td>test.demoQueue</td><td>demoQueue</td><td>demoJMSServer</td></tr></tbody></table><div>NewDelete</div><div>Showing 1 to 1 of 1PreviousNext</div></div></div>	<input type="checkbox"/>	Name ↕	Type	JNDI Name	Subdeployment	Targets	<input type="checkbox"/>	demoQueue	Queue	test.demoQueue	demoQueue	demoJMSServer
<input type="checkbox"/>	Name ↕	Type	JNDI Name	Subdeployment	Targets								
<input type="checkbox"/>	demoQueue	Queue	test.demoQueue	demoQueue	demoJMSServer								
6	Publish a JMS Message												
6.1	Click on demoQueue												
6.2	Click on the Monitoring tab and observe the table which lists all configured JMS destinations												
6.3	Check the box for demoJMSModule!demoQueue and click Show Messages												
6.4	Click New to create a message												
6.5	Create a JMS message according to Table 13 then click Ok to publish												
6.6	Verify the message appears in the JMS Messages table. Click on the message to view the message details.												
6.7	Use the breadcrumbs to navigate back to Summary of JMS Messages > demoQueue												
6.8	Click on the Monitoring tab and observe the queue statics for Messages Current, Messages Total, and Messages High. This should be 1 and will increment for each JMS message published.												
6.9	Navigate to Services > Messaging > JMS Servers and click demoJMSServer												
6.10	Click on the Monitoring tab and observe the JMS server statistics for Destinations Current (1), Messages Current (1), Messages Received (1), and Bytes Current (will vary by system).												

Lab #7: Application Deployment

Duration 30 minutes

Skills Learned

At the end of this exercise, you will be able to:

- Deploy an application using the admin console
- Deploy an application using weblogic.Deployer
- Monitor an application using the admin console

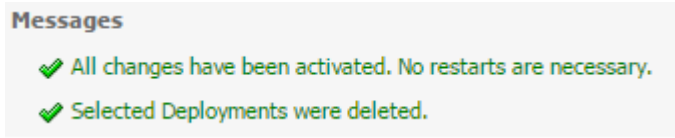
Overview

In this lab you will deploy an application to WebLogic using two common methods, the admin console and weblogic.Deployer. The admin console provides a wizard-driven interface for deploying applications. Or if you script operations and maintenance tasks, you can use a command line tool named weblogic.Deployer.

For this lab you will deploy a sample web application that's included when you install WebLogic with example code. The sample web application, mainWebApp, serves as the home page for all the other sample web applications and code examples from Oracle. The mainWebApp is packaged as an exploded archive directory and not as an archive file. The differences are covered in the lecture, but the deployment processes for both are the same.

Instructions

1	Deploy using Admin Console
1.1	Start the demo_domain admin server if not already running
1.2	Log into the admin console
1.3	Under Domain Structure, navigate to Deployments
1.4	Click Install
1.5	<p>Under current location, notice how each directory is hyperlinked. You can click on these links to navigate around the server's file system.</p> <p>Navigate to</p> <p><code><APPLICATIONS_HOME>/wl_server/examples/build/</code></p> <p>and select the radio button next to mainWebApp</p> <p>Click Next</p>

1.6	Select Install this deployment as an application and click Next
1.7	Target the deployment to the AdminServer then click Next
1.8	<p>Notice all the settings on the Optional Settings page. This page allows you to over security settings, how the application is staged and accessed, and how deployment plans are managed. Future labs will discuss these settings in further detail.</p> <p>For now accept the defaults and click Finish.</p>
1.9	The application should now appear in the Deployments table. Observe the State and health of the application.
2	Deploy using weblogic.Deployer
2.1	<p>You will now redeploy this application using weblogic.Deployer. First delete the deployment by selecting the checkbox for mainWebApp then clicking Delete.</p> <p>Once the deployment is deleted, you should see the following message in the console:</p> 
2.2	<p>Open a new terminal window and source the following WebLogic script to set your environment.</p> <p>In bash shell</p> <pre>source <ORACLE_HOME>/wlserver/server/bin/setWLSEnv.sh</pre> <p>This script will properly set CLASSPATH, JAVA_OPTIONS for running most WebLogic command line utilities.</p>
2.3	<p>Run weblogic.Deployer to deploy the same application:</p> <p>The syntax for executing weblogic.Deployer is:</p> <pre>java [SSL Arguments] weblogic.Deployer [Connection Arguments] [User Credentials Arguments] COMMAND-NAME command-options [Common Arguments]</pre> <p>For this lab execute weblogic.Deployer with the following parameters:</p> <pre>java weblogic.Deployer -adminurl t3://localhost:7001 - username weblogic -password weblogic123 -deploy <APPLICATION_ROOT>/wl_server/examples/build/mainWebApp</pre> <p>You should see the following output:</p> <pre><Apr 30, 2015 9:21:26 PM MDT> <Info> <J2EE Deployment SPI> <BEA-260121> <Initiating deploy operation for application, mainWebApp [archive:</pre>

	<pre> /u01/udemy/applications/wl_server/examples/build/mainWebApp :] , to configured targets.> Task 5 initiated: [Deployer:149026]deploy application stockBackEnd on AdminServer. Task 5 completed: [Deployer:149026]deploy application stockBackEnd on AdminServer. Target state: deploy completed on Server AdminServer </pre>
2.4	If you receive the error Error: Could not find or load main class weblogic.Deployer, then your CLASSPATH is not set properly. Make sure you source setWLSEnv.sh.
2.5	Back in the admin console, navigate to Deployments and verify the application appears in the Deployments table.
3	Application Monitoring
3.1	<p>Under Deployments, click on the Monitoring tab</p> <p>The Monitoring tab provides a table containing various statistics for each deployed application or library. Observe the series of sub-tabs under Monitoring for Web Applications, JMS, Resource Adapters, EJBs, Web Services, etc etc.</p> <p>Each one of these tabs provides runtime information for a particular deployed resource or application.</p> <p>Since the application you deployed is a web application, it appears under Web Applications tab. The tab includes a table the lists each web application, its context root and various statistics.</p>
3.2	Click on the Control tab at the very top of the page.
3.3	Click on the name of the application in the Deployments table.
3.4	<p>Click on the Monitoring tab. This tab shows more detailed monitoring information for this particular web application. Each sub-tab will show information for a given component or module within the application.</p> <p>Observe the various default attributes that are being monitored.</p>
3.5	<p>Click on the Servlets sub-tab.</p> <p>This page will show all the registered servlets within this web application along with numerous statistics.</p>

Lab #8: Production Deployment

Duration 60 minutes

Skills Learned

At the end of this exercise, you will be able to:

- Manage application deployments using application install directories and deployment plans
- Perform zero-downtime application upgrades
- Deploy applications in admin mode for testing

Overview

In this lab you will work with one of the example applications that ships with WebLogic. WebLogic ships with many sample applications with one particular application serving as the home page called mainWebApp. It is this application that we will be working with during this lab.




For this lab you will learn deployment tasks typically associated with production deployments. You will create an application install directory which will be used for staging and managing application files prior to deployment. You will also learn how to deploy an application using a deployment plan for your specific environment. Recall the deployment plan can be used to decouple environment configuration settings from an application.

Next you will learn how to perform a zero-downtime upgrade using the production redeployment strategy. This will involve versioning our sample application during deployment and then deploying a new version alongside the original. As part of the production redeployment strategy, you will deploy the new version of the application in an administrative state so that it can be tested prior to making it available to all users.


Instructions

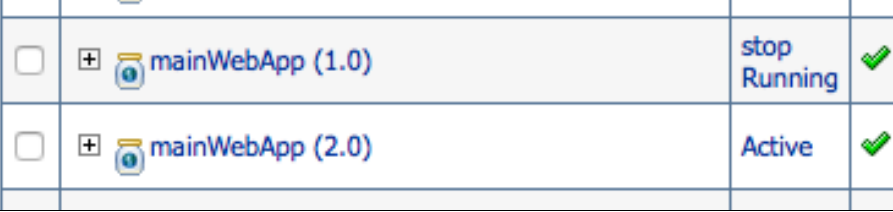

1	Create a Deployment Plan using weblogic.PlanGenerator
1.1	Open a terminal or command window
1.2	<p>Create the following directories</p> <pre> <APPLICATION_ROOT>/export <APPLICATION_ROOT>/export/mainWebApp_1.0 <APPLICATION_ROOT>/export/mainWebApp_1.0/app <APPLICATION_ROOT>/export/mainWebApp_1.0/plan </pre> <p>This directory structure starting with mainWebApp_1.0 constitutes an</p>

	<p>application installation directory. This directory contains both the application source files and a deployment plan. An application install directory allows us to organize specific version of an application and their deployment plan or plans.</p> <p>You can use the admin console or weblogic.Deployer to deploy an application with a deployment plan using this directory structure.</p>
1.3	<p>mainWebApp is distributed by WebLogic as an exploded archive directory instead of a .war file.</p> <p>Copy the entire exploded war directory from <code><APPLICATION_ROOT>/wl_server/examples/build/mainWebApp</code> to the <code>mainWebApp_1.0/app</code> directory you just created.</p> <p>Your app directory should look like the following:</p> <pre>mainWebApp_1.0/app/mainWebApp/</pre>
1.4	Source <code>setWLSEnv.sh (.cmd)</code> to set your environment's CLASSPATH
1.5	<p>Run Plan Generator to create a deployment plan for the application</p> <pre>java weblogic.PlanGenerator -root <APPLICATION_ROOT>/export/mainWebApp_1.0</pre> <p>PlanGenerator will create a deployment plan under <code>mainWebApp_1.0/plan</code></p>
1.6	<p>View the <code>plan.xml</code> under <code>mainWebApp_1.0/plan</code>. I recommend referring to https://docs.oracle.com/middleware/1213/wls/DEPGD/toc.htm for more information on the contents on a deployment plan and their meaning.</p> <p>The deployment plan allows an administrator to override certain configuration parameters, such as external resource references or tuning parameters such as session timeout, using variables and XPath.</p>
2	Deploy Application Install Directory
2.1	If there are any WebLogic servers running shut them down now.
2.2	<p>Start the admin server for the <code>wl_server</code> domain.</p> <p>The <code>wl_server</code> domain is located under <code><DOMAIN_ROOT>/wl_server</code></p>
2.3	Log into the admin console using the credentials you specified in the QuickStart Wizard.
2.4	Navigate to Deployments
2.5	Check the box next to mainWebApp in the Deployments table and click Delete
2.6	Click Install
2.7	Using either the Path text box or the Current Location links, locate <code><APPLICATION_ROOT>/export</code>
2.8	Check the radio button for mainWebApp_1.0 and click Next
2.9	Install the deployment as an application and click Next
2.10	Under Optional Settings accept all defaults and click Next
2.11	Click Finish

	mainWebApp is now deployed using the default deployment plan you generated.
3	Update the Deployment Plan
3.1	Go back to Deployments and click on mainWebApp
3.2	Click on the Deployment Plan tab. You should see two sub-tabs: Tuning Parameters and Resource Dependencies. Observe that nothing is defined for either of these tabs.
3.3	Click on the Configuration tab
3.4	<p>Change the Session Timeout value from 3600 seconds to 600 seconds. This is a great example of taking a value that may be sufficient for development, but for production requires something else.</p> <p>Click Save</p> <p>At the top of the page you will see several messages:</p> <div>  All changes have been activated. No restarts are necessary.  Deployment plan has been successfully updated.  Remember to update your deployment to reflect the new plan when you are finished with your changes. </div>
3.5	Click on the Deployment Plan tab and observe the new deployment plan variable
3.6	Now update the application so the new deployment plan can take effect.
	Navigate to Deployments
3.7	<p>Check the box next to mainWebApp and click Update</p> <p>WebLogic allows you to update an application one of two ways:</p> <ul style="list-style-type: none"> a) In-place upgrade which will use the application files that exist under the application install directory by default b) Re-deploy of the entire application allowing you to override the location of the source files and deployment plan.
3.8	Select the Re-deploy option and click Finish
3.9	<p>Back in your terminal window view the contents of plan.xml.</p> <p>Under <variable-definition> you should now see a SessionDescriptor_TimeoutSecs variable with a value of 600.</p> <pre><variable> <name>SessionDescriptor_TimeoutSecs_14305395201780</name> <value>600</value> </variable></pre> <p>Further down the file you should see this variable being applied using XPath:</p> <pre><module-descriptor external="false"> <root-element>weblogic-application</root-element> <uri>META-INF/weblogic-application.xml</uri> <variable-assignment> <name>SessionDescriptor_TimeoutSecs_14305395201780</name> <xpath>/weblogic-application/session-descriptor/timeout-</pre>

	<pre>secs</xpath> </variable-assignment> </module-descriptor></pre>
3.10	Deployment plans like this one can be used to decouple and resolve external dependencies and tuning parameters from an application for a given target environment.
4	Specify an Application Version during Deployment
4.1	Log back into the admin console and delete the mainWebApp deployment
4.2	<p>Open a terminal window and navigate back to <APPLICATION_ROOT>/export</p> <p>The mainWebApp application currently does not specify any version information. There are two ways to specify the version of an application:</p> <ol style="list-style-type: none"> 1. Generate a MANIFEST.mf with an application version identifier during the software build process 2. Specify an application version during deployment using weblogic.Deployer <p>You will use weblogic.Deployer to deploy the application install directory for mainWebApp and specify a version identifier.</p>
4.3	<p>The syntax for deploying an application along with a deployment plan is:</p> <pre>java weblogic.Deployer -adminurl http://localhost:7001 -username weblogic -password weblogic -deploy -name myTestDeployment -source /myDeployments/installedApps/myApplication/app/myApplication.ear -targets myCluster -stage -plan /myDeployments/installedApps/myApplication/plan/plan.xml</pre> <p>Our exact syntax is:</p> <pre>java weblogic.Deployer -adminurl http://localhost:7001 -username weblogic -deploy -name mainWebApp -source <APPLICATION_ROOT>/export/mainWebApp_1.0/app/mainWebApp -targets AdminServer -stage -plan <APPLICATION_ROOT>/export/medrec_v1.0/plan/Plan.xml - appversion 1.0</pre>
4.4	<p>If the syntax is correct and everything goes smoothly you should see the following output. Note the Version=1.0 in the log entry.</p> <pre>Task 1 completed: [Deployer:149026]deploy application mainWebApp [Version=1.0] on AdminServer. Target state: deploy completed on Server AdminServer</pre>
4.5	Log into the admin console and navigate to the Deployments summary table. You should now see the version identifier included in the name of the deployment.

	
5	Perform a zero-downtime upgrade using Production Redeployment
5.1	<p>For this part of the lab let's assume we have received a new version of the application from development that needs to be deployed to production.</p> <p>If the application meets the requirements for production redeployment (See Online Oracle Documentation) then we can perform this upgrade without impacting any currently connected users.</p>
5.2	Open a terminal window and navigate back to the application install directory we created for mainWebApp.
5.3	<p>Create a copy of the mainWebLogic_1.0 directory and name it mainWebLogic_2.0</p> <p>In <APPLICATION_ROOT>/export you should now have the following directories:</p> <pre><APPLICATION_ROOT>/export/mainWebApp_1.0 <APPLICATION_ROOT>/export/mainWebApp_2.0</pre>
5.4	Deploy version 2.0 of mainWebApp using the same weblogic.Deployer syntax used to deploy version 1.0. However this time, be sure to change the – appversion flag to 2.0 instead of 1.0.
5.5	<p>If your redeployment is successful you will see the following message:</p> <pre><May 4, 2015 8:16:21 AM MDT> <Info> <J2EE Deployment SPI> <BEA-260121> <Initiating deploy operation for application, mainWebApp [archive: /u01/udemy/applications/export/mainWebApp_1.0/app/mainWebAp p], to AdminServer .> Task 2 initiated: [Deployer:149026]deploy application mainWebApp [Version=2.0] on AdminServer.</pre>
5.6	<p>You may also see the following message if there are existing user sessions for mainWebApp. This message simply states that there are existing user sessions and that the application is in an administrative mode until all user sessions timeout.</p> <pre><May 4, 2015 8:16:21 AM MDT> <Notice> <HTTP> <BEA-101351> <Server has detected pending sessions while gracefully transitioning webapp module mainWebApp of application mainWebApp [Version=1.0] from running to administration mode. Server will wait for pending sessions to either become invalidated or timed out. The current timeout value is 3,600 seconds. To terminate sessions immediately, use the -force option.></pre>
5.7	If you refresh your Deployments table you may see the following status:

	
5.8	<p>mainWebApp (1.0) will be marked as retired once all user sessions have exited or timed out.</p> 
5.9	<p>When using the default deployment mode of staging, WebLogic copies the application files and deployment plans to each target. The application is then accessed from this staging location on the target.</p> <p>You can observe this by looking at the contents of the following directory:</p> <pre><DOMAIN_ROOT>/wl_server/servers/AdminServer/stage</pre> <p>The stage directory will contain subdirectories for each application deployed to the AdminServer using the stage mode. Inside these application subdirectories will be additional subdirectories for each version of the application that has been deployed.</p> <p>Verify this for yourself by listing the contents of:</p> <pre><DOMAIN_ROOT>/wl_server/servers/AdminServer/stage/mainWebApp</pre> <p>Inside each version directory (1.0 and 2.0) will be the application source files and any deployment plans.</p>

Lab #9: WebLogic Security

Duration 60 minutes

Skills Learned

At the end of this exercise, you will be able to:

- Enabling the Administration Port
- Create users and groups
- Configure an authentication provider
- Configure an auditing provider
- Secure an application using roles and policies
- Define password complexity and user logout

Overview

In this lab you will learn some basic yet important tasks related to securing WebLogic resources with roles, policies, and understanding WebLogic features such as security realms and security providers. You will also learn how to configure the domain-wide administration port for separating application traffic from administration traffic using SSL.

In this lab you will continue to work with sample application mainWebApp by protecting the application using a custom role and policy that you define.

In addition, you will learn how to configure a SQL-based authentication provider using the embedded Derby database, however the procedure is nearly the same for any type database.

This lab will also show you how to configure password complexity and user logout. While not typically done in a development environment, there are usually requirements either from your company's security organization or a customer that require passwords to be complex and how long and often to lock a user out.

Configuration Parameters

Table 15 User and Group Parameters

Parameter	Value
User name	sqluser
Provider	derbySQLAuthenticator
Password	weblogic123

Group name	AppAdmins
Group provider	derbySQLAuthenticator

Table 16 Admin Port Number

Server	Admin Port
Admin server	9001

Instructions

	Configure Domain-wide Admin Port
1.1	Start the admin server for the wl_server domain
1.2	<p>Navigate to the AdminServer Configuration Page (Environment > Servers > Admin Server) and set the Local Administration Port Override under the Advanced section to 9001.</p> <p>Click Save</p>
1.3	<p>Enable the Domain Wide Administration Port by clicking on the name of the domain under Domain Structure and checking Enable Administration Port.</p> <p>Click Save</p> <p>Once you click Save, you will no longer have access to the admin console. If you try to navigate anywhere within the application you will receive the following warning:</p> <p><i>Console/Management requests or requests with <require-admin-traffic> specified to 'true' can only be made through an administration channel</i></p>
1.4	<p>Log back into the admin console using https and the new admin port number:</p> <p>https://localhost:9001/console</p>
1.5	<p>You should receive a browser popup warning you about the certificate.</p> <p>Accept the warning to proceed to the admin console.</p>
1.6	<p>Look at the AdminServer output in the terminal window used to start the AdminServer. Notice there is now a separate network channel entry:</p> <pre><May 20, 2015 10:27:55 PM MDT> <Notice> <Server> <BEA-002613> <Channel "DefaultAdministration[3]" is now listening on 127.0.0.1:9001 for protocols admin, ldaps, https.></pre>
	Prepare Database for SQL Authentication
2.1	<p>WebLogic authentication providers are security modules that provide authentication services. WebLogic comes out of the box with several different types of authentication providers, such as LDAP, SQL, SAML, x.509 and many others.</p>

	<p>Any resource, service or application that requires authentication uses a WebLogic authentication provider.</p> <p>The default Authentication Provider uses the embedded LDAP server that comes with WebLogic for storing user and group information. However for this lab let us assume that we have our own enterprise database and that we want this database to serve as our identity store.</p> <p>You will create a SQL Authenticator that will point to an existing Derby database. First you need to prepare the Derby database by creating the schemas for users and groups.</p>
2.2	<p>Launch the interactive Derby client. You may need to set your JAVA_HOME environment variable first.</p> <p>Execute: <code><ORACLE_HOME>/wlserver/common/derby/bin/ij (ij.bat in Windows)</code></p>
2.3	<p>Connect to the examples database:</p> <pre>connect 'jdbc:derby://localhost:1527/examples' user 'examples' password 'examples';</pre>
2.4	<p>Copy and paste the entire SQL statement block into ij. The semi-colons are important and indicate the end of the command. These series of commands will create the necessary tables that the SQL Authenticator will look for.</p> <pre>CREATE TABLE USERS (U_NAME VARCHAR(200) NOT NULL, U_PASSWORD VARCHAR(50) NOT NULL, U_DESCRIPTION VARCHAR(1000)) ; ALTER TABLE USERS ADD CONSTRAINT PK_USERS PRIMARY KEY (U_NAME) ; CREATE TABLE GROUPS (G_NAME VARCHAR(200) NOT NULL, G_DESCRIPTION VARCHAR(1000) NOT NULL) ; ALTER TABLE GROUPS ADD CONSTRAINT PK_GROUPS PRIMARY KEY (G_NAME) ; CREATE TABLE GROUPMEMBERS (G_NAME VARCHAR(200) NOT NULL, G_MEMBER VARCHAR(200) NOT NULL) ; ALTER TABLE GROUPMEMBERS ADD CONSTRAINT PK_GROUPMEMS</pre>

	<pre> PRIMARY KEY (G_NAME, G_MEMBER) ; ALTER TABLE GROUPMEMBERS ADD CONSTRAINT FK1_GROUPMEMBERS FOREIGN KEY (G_NAME) REFERENCES GROUPS (G_NAME) ON DELETE CASCADE ; </pre>
2.5	<p>Exit ij by typing:</p> <pre>exit;</pre>
3	Configure SQL Authentication
3.1	Start the wl_server domain's admin server
3.2	Log into the admin console
3.3	<p>Under Domain Structure click on Security Realms then click on myrealm.</p> <p>myrealm represents the default security realm. A WebLogic security realm is a container for defining a variety of WebLogic security services, such as providers, users, groups, roles, and many other items.</p>
3.4	Click on the Providers tab
3.5	<p>The Authentication tab contains a table of configured Authentication providers. You should see the two default providers, DefaultAuthentication which uses the embedded LDAP as an identity store and the DefaultIdentityAsserter which handles authorization checks.</p>
3.6	To create a SQL Authentication provider, click New
3.7	Enter 'derbySQLAuthenticator' for the Name
3.8	<p>Select SQLAuthenticator from the Type dropdown list and click OK.</p> <p>Once you click OK you will see a message that says the server needs to be restarted. Before restarting the admin server you need to complete configuring the SQLAuthenticator.</p>
3.9	Click on the derbySQLAuthenticator in the Authentication Providers table you just created
3.10	<p>On the Provider Specific sub-tab, enter the name of the examples datasource in the Data Source Name field.</p> <p>SQL Authenticator providers use pre-configured WebLogic datasources. The datasource already defines how to connect to Derby in this case.</p> <p>The name of the examples data source is examples-demo</p>
3.11	Take a look at all the other configuration parameters on this page. Each

	<p>parameter represents an interface or contract that the authentication provider must perform. Next to each configuration item is the SQL that is needed to perform that function. The DDL you executed earlier to create the USERS and GROUPS table corresponds to SQL presented on this page.</p> <p>WebLogic allows you to modify this SQL if your schemas are laid out differently.</p> <p>Click Save when complete.</p>
3.12	Restart the admin server by shutting it down via the admin console and restarting using the startWebLogic script.
4	Create User and Group
4.1	Log back into the admin console and navigate to Security Realms > myrealm
4.2	Click on the Users and Groups tab
4.3	<p>To create a new user click on New in the Users table and specify the values in Table 15 then click Ok</p> <p>By specifying the Derby SQL Authenticator, WebLogic will store this new user in Derby.</p>
4.4	Click on the Groups sub tab
4.5	<p>Notice the various default WebLogic groups. All of these are stored in the embedded LDAP.</p> <p>Click on New to create a new group according to Table 15 then click OK</p>
4.6	Go back to the Users tab and click on the new user you created.
4.7	<p>Add this user to the new group by clicking the Groups sub tab and checking the box next to AppAdmins and clicking the blue arrow to assign the user to that group.</p> <p>The AppAdmins group should move from the Available groups list to the Chosen groups list.</p> <p>Click Save</p> <p>You have now created and assigned a user to a group that is being stored in a SQL database. This group and user can now be tied to a security policy for protecting WebLogic resources and applications. This will be demonstrated later on in this lab.</p>
5	Configure Password Complexity and User Lockout
5.1	<p>You can define password complexity by modifying the included Password Validation provider.</p> <p>Navigate back to Security Realms > myrealm > Providers and click on the Password Validation sub-tab</p>
5.2	<p>WebLogic comes with a default password validator named SystemPasswordValidator</p> <p>Click on the password validator</p>
5.3	The Provider Specific tab lists various policies and criteria that you can modify to

	<p>define a complex password validator.</p> <p>The default configuration defines a minimum length of 8 characters and at minimum 1 non-alphabetic character.</p> <p>For this lab exercise the default values are sufficient, however feel free to change them if you wish.</p>
5.4	To configure User Lockout, navigate back to Security Realms > myrealm and click on the User Lockout tab.
5.5	<p>The User Lockout page allows you to set various lockout settings for failed login attempts.</p> <p>The defaults are acceptable for this lab.</p>
6	Configure an Auditing Provider
6.1	You can configure an auditing provider to logging security events in WebLogic, such as failed application login access.
6.2	Navigate to Security Realms > myrealm > Providers and click on the Auditing tab.
6.3	<p>Click New to create a new auditing provider and name it auditor.</p> <p>Click Ok</p>
6.4	In the Auditing Providers table, click on the new auditor.
6.5	<p>On the Provider Specific tab, note the various configuration settings.</p> <p>We are interested in auditing failed authentication attempts, so under Severity select FAILURE.</p> <p>Click Save.</p>
6.6	You will see a message that a restart is required. Restart the admin server for the changes to take effect.
6.7	<p>Once the admin server has been restarted, open a terminal window and navigate to <code><DOMAIN_ROOT>/wl_server/servers/AdminServer/logs</code></p> <p>By default most log files will appear under <code><server_name>/logs</code></p> <p>If you list the contents of the directory you should see a log file named:</p> <p>DefaultAuditRecorder.log</p> <p>This audit log file will contain security events based on how the audit provider was configured.</p>
6.8	<p>To test the auditing provider, log into the admin console using an invalid username and or password such as testuser/testuser.</p> <p>After the failed login attempt, view the contents of the auditing provider.</p> <p>In my example I tried to login using a user that did not exist named testuser:</p> <pre>#### Audit Record Begin <May 4, 2015 10:35:37 AM></pre>

	<pre><Severity =FAILURE> <<<Event Type = Authentication Audit Event><testuser><AUTHENTICATE>>> <FailureException =javax.security.auth.login.FailedLoginException: [Security:090304]Authentication Failed: User testuser javax.security.auth.login.FailedLoginException: [Security:090302]Authentication Failed: User testuser denied> Audit Record End ###</pre> <p>The auditing log file can get quite large if the SEVERITY level is not set appropriately. If you are required to audit all security events, then the log file will grow enormously large, even in development or QA environments. While you can specify when the log file is rotated, you will need a process to clean up this particular file over time.</p>
7	Securing an Application
7.1	<p>Most Java EE-based applications use declarative security to protect the application or resources within the application. This is usually in the form of Java-based annotations or more commonly through the use of deployment descriptors.</p> <p>For instance, a developer can declare that a certain URL within an application needs to only be accessible by a certain group of people or a certain role. Developers can define what group or role has access, however you as an administrator can define who belongs to that role or group or even redefine the entire security model altogether.</p> <p>You do all this through the admin console. In order to do so, you must deploy an application using special options to override the application's security model.</p> <p>In this section of the lab you will define a custom role and policy that will only allow access to our sample application for the group you created earlier.</p>
7.2	Make sure the admin server for the wl_server domain is running.
7.3	Log into the admin console and navigate to Deployments.
7.4	Delete all versions of mainWebApp.
7.5	<p>Re-deploy mainWebApp using the admin console by clicking Install and locating the application install directory previously created for mainWebApp.</p> <pre><APPLICATION_ROOT>/export/mainWebApp_2.0</pre> <p>Check the radio button for this application and click Next</p>
7.6	Install as an application and click Next
7.7	On the Optional Settings page under the Security section, select Custom Roles and Policies. This option will allow us to completely override any security policies defined in the application's deployment descriptor.
7.8	<p>Click Next then Finish.</p> <p>You will be taken to the settings for mainWebApp.</p>
7.9	Click on the Security tab.

	<p>The Security tab allows you to define roles and policies for the application or for a specific URL pattern.</p> <p>For this lab we are going to secure the entire application by applying a stand-alone web application scoped role.</p> <p>Under Application Scope Roles sub tab, click New.</p>
7.10	Name the new role SampleAppUserRole and click OK
7.11	<p>Click on the new role to bring you to the Scoped Role Conditions page. You use the Conditions page to define membership for the role.</p> <p>Click Add Conditions and select Group from the Predicate List then click Next.</p>
7.12	Under Group Argument Name type in Administrators. Click Add to add the group to the list. Then click Finish.
7.13	Click Save on the Scoped Role Conditions page.
7.14	Using the bread crumbs, navigate to mainWebApp > Roles
7.15	<p>Click on the Policies tab under Application Scope.</p> <p>You will now create a policy that protects the application using the role you just defined.</p>
7.16	<p>Click Add Conditions and select Role from the Predicate List.</p> <p>Click Next.</p>
7.17	<p>Enter SampleAppUserRole and click Add.</p> <p>Then click Finish.</p>
7.18	<p>Click Save.</p> <p>The mainWebApp application should now be protected using the policy you just defined, which only allows access to the group Administrators.</p>
7.19	You will now test the new policy by access the application. However in order to ensure our admin console session does not interfere with this test, you must log out of the admin console. I recommend closing the browser, and in some cases you may have to flush your browser cache.
7.20	<p>After logging out and reopening your browser, navigate to http://localhost:7001</p> <p>The application's context root is actually /, so the above URL will take us directly to the application. By configuring the policy you are now challenged to log into the sample application.</p> <p>Log in using the admin username and password credential.</p>

Lab #10: Clusters

Duration 2 hours

Skills Learned

At the end of this exercise, you will be able to:

- Configure clusters
- Deploy servers and resources to a cluster
- Configure WebLogic's HttpClusterServlet

Overview

In this lab you will learn how to configure WebLogic for high availability by creating WebLogic clusters and configuring server and service level migration. Within these clusters you will deploy servers, resources, and applications. The process is not much different than deploying to a single server as you have already done in previous labs, however by targeting resources to clusters, WebLogic automatically provides high availability and load balancing of resources across multiple WebLogic managed servers.

Once the cluster is configured, you will configure a simple web proxy for load balancing web browser requests across the cluster. The proxy will also support failover by directing any request away from the failed server or service to a healthy server or service.

You will also learn how to set up a simple web proxy for load balancing web browser requests across a WebLogic cluster. In a production environment you would typically use Oracle HTTP Server or Apache with the mod_wls plugin, or a hardware load balancer such as BigIP's F5, however in this lab you will utilize WebLogic's HttpClusterServlet to perform load balancing and failover of HTTP requests.

The process for using clusters starts with first configuring a cluster, then managed servers are then created and assigned to the cluster.

For the purposes of this lab you will continue to use the wl_server example domain from Oracle. You will modify this domain to include a cluster and then deploy and access resources in the cluster.

Configuration Parameters

Table 17 Managed Server Parameters

Server	Listen Address	Listen Port	Local Admin Port
ms1	<Local IP of Machine>	8002	9002
ms2	<Local IP of Machine>	8003	9003
ms3	<Local IP of Machine>	8004	9004

Parameter	Value
Name	demoCluster
Messaging Mode	Unicast
Unicast Broadcast Channel	Blank

Table 18 Cluster Parameters

Instructions

1

Create Managed Servers

1.1

In this section you will create two managed servers that will be part of your WebLogic cluster. Recall that the admin server cannot be part of a WebLogic cluster.

First shutdown all running instances of WebLogic.

1.2

Start the wl_server domain's admin server.

1.3

Log into the admin console

1.4

Before creating the managed servers, it is imperative to define a listen address for the Admin Server. The managed servers in a cluster must have a listen address defined, otherwise cluster communication is unreliable since there are multiple network channels defined when no listen address is specified.

Change the listen address for the AdminServer to the IP address of the host.

1.5

Restart the Admin Server

1.6

Log back into the admin console

1.7

Under Domain Structure, navigate to Environment > Servers

1.8

Click New to create a server.

1.9

Configure the server's name and port according to Table 17 Managed Server Parameters then click Finish.

Repeat for the second and third managed server.

Your Servers table should now look like the following:

<input type="checkbox"/>	Name ↕	Type	Cluster	Machine	State	Health	Listen Port
<input type="checkbox"/>	AdminServer(admin)	Configured			RUNNING	✔ OK	7001
<input type="checkbox"/>	ms1	Configured			SHUTDOWN	Not reachable	8002
<input type="checkbox"/>	ms2	Configured			SHUTDOWN	Not reachable	8003
<input type="checkbox"/>	ms3	Configured			SHUTDOWN	Not reachable	8004

The Listen Address attribute for a WebLogic server specifies what listening

	<p>address WebLogic will respond to. In this lab you are specifying the IP address of the machine that is hosting the managed server, such as 192.168.1.150.</p> <p>When you specify a listen address, WebLogic will only respond on that listen address. It will no longer respond to any other value, such as localhost in this case.</p>
2	Create a Cluster
2.1	Under Domain Structure, navigate to Environment > Clusters.
2.2	Click New then Cluster to create a cluster.
2.3	Configure the cluster according to Table 18 Cluster Parameters and click Ok.
2.4	Click on the new cluster in the Clusters table.
2.5	<p>Take a look at the various configuration options under the Configuration tab. Most of these settings are out of scope for this lab.</p> <p>One of the most important cluster settings is the Default Load Algorithm. This drop down list defines the algorithm to be used for load-balancing between replicated services. Round-robin is the default algorithm and works by cycling through each managed server in order.</p>
2.6	Click on the Servers sub-tab and view the bottom of the page. There exists a Servers table which lists the servers that are members of this cluster. Currently no servers have been assigned.
2.7	Click Add
2.8	Select ms1 then click Next
2.9	Repeat for ms2
3	Start the Cluster
3.1	To start a cluster means to start all the members within a cluster. To do this you will execute the startManagedWebLogic script for each managed server.
3.2	Open a new terminal window for starting the managed server.
3.3	<p>Execute the startManagedWebLogic.sh (.cmd) script to start ms1</p> <p>Recall the syntax is:</p> <pre><DOMAIN_HOME>/bin/startManagedWebLogic.sh <name_of_managed_server> <t3_url_to_admin_server></pre> <p>For this lab you must connect securely using t3s to the admin server. The syntax is:</p> <pre><DOMAIN_ROOT>/wl_server/bin/startManagedWebLogic.sh ms1 t3s://<admin_listen_address>:9001</pre>
3.4	<p>You will be prompted to enter the username and password for the starting the managed server. If you recall in one of the earlier labs, you created a boot.properties file for the managed server.</p> <p>Terminate the process (ctrl-c on most systems) and create a boot.properties for each managed server.</p>

3.5	<p>Start ms1 again.</p> <p>Verify ms1 is now RUNNING and listening on Port 8001 by observing the terminal window output.</p>
3.6	Open another terminal window and perform the same steps to start ms2.
3.7	Once all managed servers are running, log into the admin console.
3.8	<p>You can use the admin console to monitor the state and health of a cluster.</p> <p>Navigate to Environment > Clusters and click on the name of the cluster.</p>
3.9	Click on the Monitoring tab and notice the three sub-tabs Summary, Health, and Failover.
3.10	<p>The Summary tab provides statistics related to the status of managed servers within the cluster.</p> <p>The Health tab provides overall health status for each managed server, much like the main Servers page.</p> <p>The Failover tab displays information related to failover in the even that a cluster member would fail.</p>
4	Deploy Resources to a Cluster
4.1	<p>Now that you have a cluster configured with managed servers, you can now deploy resources to this cluster that can be used by clustered applications.</p> <p>In this lab you will re-target one of the example data sources to the cluster. Doing so makes the data source highly available. If the data source on a cluster member would fail, such as ms1, the same data source would be available on the other cluster members, ms2.</p>
4.2	Navigate to Services > Data Sources
4.3	Click on the examples-demo data source
4.4	<p>On the Targets tab, check the box for demoCluster. Leave AdminServer checked as well.</p> <p>You can target a data source to an entire cluster and individual servers. We still want the examples-demo data source to be accessible from the AdminServer.</p> <p>Click Save.</p>
4.5	<p>You can verify that the data source is functioning properly on a given cluster member by testing it through the admin console.</p> <p>Click on the Monitoring tab for the examples-demo data source.</p>
4.6	<p>Click on the Testing sub-tab and notice that each target that the data source is deployed to is listed.</p> <p>Select the radio button for a particular managed server and click Test Data Source.</p> <p>At this point the test should return successful.</p>

5	Deploy an Application to a Cluster
5.1	In this section you will now target an application deployment to a cluster much like you did for the data source. By clustering an application, WebLogic can provide high availability for that application. If the application fails on a ms1 and it still available on ms2.
5.2	Navigate to Deployments.
5.3	Click on the mainWebApp application.
5.4	<p>Before you target the application to the cluster, remove the security policy and role you defined earlier for this application.</p> <p>Click on the Security tab then Policies sub-tab under Application Scope. Check the box next to the Security Policy and click Remove then Save.</p>
5.5	<p>Remove the security role by click on the Roles sub-tab under Application Scope.</p> <p>Check the box next to the role and click Delete.</p>
5.6	<p>Click on the Targets tab and target the application to the cluster and the AdminServer and click Save.</p> <p>After you click Save, the mainWebApp may attempt to launch a browser window for each managed server. This behavior is programmed into the mainWebApp application and is not the norm for most other web applications.</p> <p>You can now access this web application on each cluster member, ms1 and ms2.</p> <p>Try it now using the following URLs in your browser:</p> <p>http://<listen_address_for_ms1>:8002</p> <p>http://<listen_address_for_ms2>:8003</p> <p>Now that the application has been clustered across to managed servers, you now need a web proxy to serve as a single endpoint to your clients and to provide load balancing across the cluster members.</p> <p>Oracle provides libraries for Apache, Oracle HTTP Server, IIS and iPlanet for providing load balancing for web applications across a cluster.</p>
6	Configure Load Balancer
6.1	<p>You will configure the WebLogic HttpClusterServlet to serve as a proxy and load balancer for HTTP requests.</p> <p>First download the proxy.war application from the course and place it under cd <DOMAIN_HOME>/proxy</p> <p>You will have to create the proxy directory.</p>

	Detailed instructions on configuring the HttpClusterServlet can be found in the References section of this Lab Guide.
6.2	<p>The proxy.war application contains 2 deployment descriptors that define the HttpClusterServlet and associated configuration parameters.</p> <p>The proxy.war that you downloaded from the Oracle WebLogic 12c for Administrators course contains a sample configuration. You must explode the war file and update the configuration to match your environment.</p> <p>Explode the archive using jar:</p> <pre>jar -xvf proxy.war</pre> <p>You should see WEB-INF/web.xml and WEB-INF/weblogic.xml extracted from the archive.</p>
6.3	<p>Edit the WEB-INF/web.xml file and locate the <code><init-param>WebLogicCluster</init-param></code> XML element.</p> <p>This parameter defines all the cluster members in a cluster in the form of:</p> <pre>listen_address1:listen_port1 listen_address2:listen_port2</pre> <p>Update the listen addresses and port numbers to match ms1 and ms2 in your environment.</p>
6.4	Save the file and exit the editor
6.5	Deploy the exploded archive directory (<code><DOMAIN_HOME>/proxy</code>) to ms3
6.6	<p>Test the proxy by accessing the mainWebApp that is deployed to the cluster. To do this use your browser to visit:</p> <pre>http://<m3_listen_address>:<ms3_listen>port>/</pre> <p>The proxy will round robin requests to the sample application between ms1 and ms2.</p>

Lab #11: Node Manager

Duration 30 minutes

Skills Learned

At the end of this exercise, you will be able to:

- Control servers using the Admin Console and Node Manager
- Control servers using WLST and Node Manager
- Restart failed servers

Overview

In this lab you will configure Node Manager to control the startup and shutdown of WebLogic managed servers rather than executing shell scripts from the command line. The lab will show you how to use both the admin console for starting servers and using WebLogic Scripting Tool or WLST for starting servers.

WLST is WebLogic's scripting language for managing WebLogic domains. It is essentially a series of functions and addon modules for jython, which WLST is based on. If you are familiar with Python, then learning WLST/jython will be fairly straightforward. If you do not have any python scripting experience, do not worry as this lab will provide the commands necessary to execute.

Node Manager comes in 2 versions: shell script and Java-based. For this lab you will work with the Java version, which is the preferred version to use in that it supports a wider array of options for configuration and for security.

The Java version of Node Manager runs as a JVM process that can manage servers for one or more domains. You interact with Node Manager either through WLST or through the admin console. In this lab you will use the admin console start, restart, and stop members of the cluster created previously.

There are essentially two steps for configuring Node Manager. The first step involves configuring the `nodemanager.properties` file for your environment. This file defines various configuration parameters and behaviors for how Node Manager operates.

The second step is to configure WebLogic machines in your domain. A machine is a configuration item that represents a physical or virtual host where WebLogic runs. Managed servers are then assigned or associated with a particular machine.

This assignment is merely logical however it is intended to represent a physical mapping of WebLogic server to application host. WebLogic uses machines to identify where a WebLogic server is located in relation to other WebLogic servers. Machines play an important part in whole server and service migration as well.

Configuration Parameters

Table 19 `nodemanager.properties`

Parameter	Value
ListenAddress	<IP address of the machine>
NativeVersionEnabled	false

Table 20 Machine Parameters

Parameter	Value
Name	machine
Machine OS	Other
Type	SSL
Listen Address	Specify the IP address from <code>nodemanager.properties</code>
Listen Port	5556

Instructions

1	Configure Node Manager Properties
1.1	<p>Open a terminal window and locate the node manager properties file here:</p> <p><DOMAIN_HOME>/nodemanager/nodemanager.properties</p> <p>The location of <code>nodemanager.properties</code> has changed from Oracle WebLogic 11gR3 when <code>nodemanager</code> was located under <ORACLE_HOME>.</p> <p>In 12c node manager is now associated with the domain itself by default. You can either have a single Node Manager per host, managing all domains on that host, or you can have a node manager per domain.</p> <p>For this lab exercise you will work with a domain-specific Node Manager.</p>
1.2	Edit <code>nodemanager.properties</code> for the <code>wl_server</code> domain using a text editor
1.3	<p>Take a look at the various configuration parameters. Note the location of the log files, the default listen port for Node Manager, and the start script parameters.</p> <p>The default configuration will cause Node Manager to listen securely for incoming requests to <code>localhost:5556</code>.</p>

	<p>Update this file according to the parameters and values defined in Table 19 nodemanager.properties</p> <p>Save the file and exit the editor.</p>																								
1.4	Start the admin server for the wl_server domain if not already running.																								
1.5	Log into the admin console and navigate to Environment > Machines																								
1.6	Configure the machine according to Table 20 Machine Parameters then click Finish when complete.																								
1.7	Assign managed servers to the machine by clicking on machine in the Machines table.																								
1.8	<p>Click on the Servers tab and add ms1 and ms2 to the machine.</p> <p>Your machineA > Configuration > Servers table should look like the following:</p> <div><p>Servers (Filtered - More Columns Exist)</p><div><div>AddRemove</div><div>Showing 1 to 2 of 2PreviousNext</div><table><thead><tr><th><input type="checkbox"/></th><th>Name </th><th>Type</th><th>Cluster</th><th>Machine</th><th>State</th><th>Health</th><th>Listen Port</th></tr></thead><tbody><tr><td><input type="checkbox"/></td><td>ms1</td><td>Configured</td><td>demoCluster</td><td>machineA</td><td>SHUTDOWN</td><td>Not reachable</td><td>8001</td></tr><tr><td><input type="checkbox"/></td><td>ms2</td><td>Configured</td><td>demoCluster</td><td>machineA</td><td>SHUTDOWN</td><td>Not reachable</td><td>8005</td></tr></tbody></table><div>AddRemove</div><div>Showing 1 to 2 of 2PreviousNext</div></div></div>	<input type="checkbox"/>	Name	Type	Cluster	Machine	State	Health	Listen Port	<input type="checkbox"/>	ms1	Configured	demoCluster	machineA	SHUTDOWN	Not reachable	8001	<input type="checkbox"/>	ms2	Configured	demoCluster	machineA	SHUTDOWN	Not reachable	8005
<input type="checkbox"/>	Name	Type	Cluster	Machine	State	Health	Listen Port																		
<input type="checkbox"/>	ms1	Configured	demoCluster	machineA	SHUTDOWN	Not reachable	8001																		
<input type="checkbox"/>	ms2	Configured	demoCluster	machineA	SHUTDOWN	Not reachable	8005																		
1.9	<p>Open a terminal window and start nodemanager using the following command:</p> <pre><DOMAIN_HOME>/bin/startNodeManager.sh (.cmd)</pre>																								
2	One-time Setup for Starting Admin Server using Node Manager																								
2.1	<p>In order to start the AdminServer using Node Manager there are a few one-time setup steps that need to be done.</p> <p>You must generate startup information, specifically a boot.properties and startup.properties for the AdminServer, specific to Node Manager.</p> <p>Make sure the admin server is running using the startWebLogic script.</p>																								
2.2	<p>Since the admin server is now running securely on the domain-wide admin port, you need to connect WLST to the admin server securely as well.</p> <p>To establish an SSL connection to the admin server’s admin port, you must configure Trust for WLST.</p> <p>If you attempt to connect without configuring a trust, you will get an error message.</p> <p>To configure Trust for WLST, edit the wlst.sh script using a text editor.</p> <p>wlst.sh is located here:</p> <pre><ORACLE_HOME>/oracle_common/common/bin/wlst.sh</pre>																								

2.3	<p>Near the bottom of the script, add the following JVM parameter to the JVM_ARGS variable:</p> <pre>-Dweblogic.security.TrustKeyStore=DemoTrust</pre> <p>Save the file and close the editor.</p> <p>This JVM flag allows WLST to locate the trust keystore. The trust keystore contains the public key that can be used to trust the Admin Server's certificate.</p> <hr/> <p>Next you need to generate startup information for the AdminServer by invoking WLST and executing the nmGenBootStartupProp command:</p> <p>In a terminal window invoke the interactive WLST shell:</p> <pre><ORACLE_HOME>/oracle_common/common/bin/wlst.sh</pre> <p>Your screen should appear as:</p> <pre>Initializing WebLogic Scripting Tool (WLST) ... Jython scans all the jar files it can find at first startup. Depending on the system, this process may take a few minutes to complete, and WLST may not return a prompt right away. Welcome to WebLogic Server Administration Scripting Shell Type help() for help on available commands wls:/offline></pre>
2.4	<p>Use WLST to connect to the running AdminServer.</p> <p>The basic syntax for connecting is:</p> <pre>connect([username, password], [url], [adminServerName], [timeout])</pre> <p>where username and password are the admin user's credentials and URL is the admin server URL including port number. You can disregard the other optional parameters.</p> <p>Example syntax:</p> <pre>wls:/offline> connect('weblogic','weblogic123','t3://192.168.1.150:7001') Connecting to t3://192.168.1.150:7001 with userid weblogic ... Successfully connected to Admin Server "AdminServer" that belongs to domain "wl_server".</pre>

	<p>Warning: An insecure protocol was used to connect to the server. To ensure on-the-wire security, the SSL port or Admin port should be used instead.</p> <pre>wls:/wl_server/serverConfig></pre>
2.5	<p>Run nmGenBootStartupProps('AdminServer') to generate the startup.properties and boot.properties for the AdminServer.</p> <pre>wls:/wl_server/serverConfig> nmGenBootStartupProps('AdminServer')</pre> <p>Successfully generated boot.properties at /u01/udemy/domains/wl_server/servers/AdminServer/data/nodemanager/boot.properties. Successfully generated startup.properties at /u01/udemy/domains/wl_server/servers/AdminServer/data/nodemanager/startup.properties.</p>
2.6	Exit WLST by typing exit()
2.7	Shutdown the AdminServer gracefully using the Admin Console (Environment > Servers > Control tab) or using <DOMAIN_HOME>/bin/stopWebLogic.sh
3	Start Admin Server using WLST and Node Manager
3.1	Invoke WLST
3.2	<p>Connect to NodeManager using nmConnect. Be sure to update the IP address with the IP address or hostname of your server. This command will connect to NodeManger at the stated address on port 5556 using plain (unencrypted) communication.</p> <pre>wls:/offline> nmConnect('weblogic','weblogic123','192.168.1.150','5556','wl_server',nmType='plain')</pre>
3.3	<p>Start the AdminServer using the nmStart command:</p> <pre>nmStart('AdminServer')</pre> <pre>wls:/nm/wl_server> nmStart('AdminServer') Starting server AdminServer ... Successfully started server AdminServer ... wls:/nm/wl_server></pre> <p>You can also use the same nmStart command to start the other servers in the domain.</p>
4	Starting Managed Servers using Node Manager
4.1	Go back to the admin console and navigate to Environment > Servers .
4.2	On the Control tab, check the box next to ms1 and click Start .
	<p>You should see a message stating that a request has been sent to Node Manager to start the selected servers.</p>

4.3	Click the Refresh icon (two arrows in a circle) just above the Servers table. This will cause the Servers table to periodically update so that you can monitor the state and status of any commands that have been issued.
4.4	The State and Status of Last Action should change from STARTING and TASK IN PROGRESS to RUNNING and TASK COMPLETED once ms1 is running.
4.5	Repeat the process for starting ms2 .
5	Test Automatic Restart of Failed Servers
5.1	<p>In this section of the lab you will demonstrate how Node Manager restarts a failed server.</p> <p>To do this, make sure nodemanager is currently running in a terminal window. You will need to observe the output from nodemanager.</p>
5.2	<p>Open another terminal window and kill the JVM process for ms2. This will simulate a server failure.</p> <p>To do this, obtain the PID for the ms2 JVM process as follows:</p> <p>In Linux/OSX:</p> <pre>> ps grep Name=ms2</pre> <p>This command will return the PID of the ms2 JVM process. The PID is the first number returned. In the example below it is 7866.</p> <pre>/u01/udemy/domains/wl_server/bin > ps grep Name=ms2</pre> <pre>7866 ttys001 0:17.76 /Library/Java/JavaVirtualMachines/jdk1.7.0_75.jdk/Contents/ Home/bin/java -server -Xms256m -Xmx512m - XX:CompileThreshold=8000 -XX:PermSize=128m - XX:MaxPermSize=256m -Dweblogic.Name=ms2 - Djava.security.policy=/u01/udemy/Oracle/Middleware/Oracle_H ome/wlserver/server/lib/weblogic.policy - Dweblogic.system.BootIdentityFile=/u01/udemy/domains/wl_ser ver/servers/ms2/data/nodemanager/boot.properties - Dweblogic.nodemanager.ServiceEnabled=true - Dweblogic.security.SSL.ignoreHostnameVerification=false - Dweblogic.ReverseDNSAllowed=false -Xverify:none - Djava.endorsed.dirs=/Library/Java/JavaVirtualMachines/jdk1. 7.0_75.jdk/Contents/Home/jre/lib/endorsed:/u01/udemy/Oracle /Middleware/Oracle_Home/wlserver/../../oracle_common/modules/e ndorsed -da - Dwls.home=/u01/udemy/Oracle/Middleware/Oracle_Home/wlserver /server - Dweblogic.home=/u01/udemy/Oracle/Middleware/Oracle_Home/wls erver/server - Dweblogic.management.server=http://192.168.1.150:7001 - Dweblogic.utils.cmm.lowertier.ServiceDisabled=true</pre>

	<code>weblogic.Server</code> In Windows you will have to use Task Manager to kill the JVM process for ms2.
5.3	To kill the process in Linux/OSX, type: <code>kill -9 <PID></code>
5.4	Observe the terminal window output for the nodemanager process. You should see a statement that says: <code><INFO> <wl_server> <ms2> <The server 'ms2' is running now.></code> This statement indicates that nodemanager has restarted ms2. To confirm this, run the ps command again in another terminal window and confirm the returned PID is different than the original PID.

Lab #12: Distributed Deployment

Duration 60 minutes

Skills Learned

At the end of this exercise, you will be able to:

- Distribute a domain across remote servers using templates
- Understand how to use pack and unpack

Overview

Through the entire lab guide you have worked with domains and WebLogic servers on a single host. You have installed WebLogic and created domains on a single host. However in the real world, WebLogic installs and domains are distributed across multiple servers for a variety of reasons such as load balancing and high availability.

In previous labs you have scaled *up* your WebLogic deployment by creating additional managed servers on a single machine. Scaling up involves adding additional resources to a server, such as disk space or ram or in this case, WebLogic managed server instances.

In this lab you will learn how to scale *out* your WebLogic deployment using domain templates, specifically managed server templates. A WebLogic template is used to create or extend a WebLogic domain. You can create a template based on an existing domain and then use that template to redeploy the domain. Domain templates ease the domain creation process by allowing you to generate a template of an already configured domain and then distributing that domain to other environments, developers, or even customers for their use.

Aside from domain templates, there are managed server templates which are used to extend or scale out an existing WebLogic domain. In this lab you will learn how to create a template using the pack command. You will then learn how to create a domain from that template using the unpack command.

In this lab you will scale out an existing WebLogic domain to a second server that we will call Node B. Let us assume that our primary server, the server hosting our AdminServer is Node A. Our deployment architecture then will consist of 2 machines, Node A and Node B, hosting our WebLogic domain.

To truly test the scaling out of WebLogic and remote deployment aspect of this lab, it would be ideal to have two distinct servers to work with. This could be two

physical machines that you may have, or your machine with a VM running on it to server as that second remote host.

However if you do not have access to a second machine, you can simulate one on your existing machine using different directory structures to physically separate the Node A domain files and Node B domain files. We can further simulate a distributed deployment by using a different set of ports and host names for Node A and Node B.

This lab will work with the assumption that you will be working with a single machine.

Configuration Parameters

Table 21 Node B Node Manager

Parameter	Value
ListenAddress	demo-b
ListenPort	5557

Table 22 Machine Parameters

Parameter	Value
Name	machineB
Machine OS	Other
Type	SSL
Listen Address	demo-b
Listen Port	5557

Instructions

1	Prepare Node A and Node B environments
1.1	Open a terminal window on Node A (your machine) and shutdown all running WebLogic processes, including Node Manager.
1.2	<p>Simulate node A and node B hostnames by editing your system's host file:</p> <p>On Linux/Unix: /etc/hosts</p> <p>On Windows: C:\Windows\System32\Drivers\etc\hosts</p> <p>The hosts file allows you to define host names and map them to IP addresses.</p> <p>In this lab you will define a host name for node A (demo-a) and a host name for node B (demo-b). If you are working on a single machine, simply associate the node B hostname with the IP address of your machine.</p> <p>For example in Linux:</p>

	<pre>sudo vi /etc/hosts</pre> <p>Add the following 2 lines to the hosts file to create 2 hostnames demo-a and demo-b</p> <pre><IP_address_of_your_machine> demo-a <IP_address_of_your_machine> demo-b</pre> <p>Save the hosts file and close the editor.</p>
1.3	Verify the newly created hostnames resolve by pinging demo-a and demo-b . Unless your system is blocking ICMP, ping should return a response.
1.4	<p>Next you will create a directory structure for holding node B domain files.</p> <p>Create a directory under <LAB_HOME> for demo-b:</p> <pre>mkdir <LAB_HOME>/demo-b/</pre> <p>Create a <code>domains</code> directory under <code>demo-b/</code></p>
2	Update Domain Configuration
2.1	<p>You now need to update your domain and node manager configuration to incorporate the new host names defined in the host file.</p> <p>You will do this by using a text editor to edit the domain and node manager configuration files.</p>
2.2	<p>Using a text editor, open <DOMAIN_ROOT>/wl_server/config/config.xml</p> <p>Change the value for all occurrences of <listen-address></listen-address> to point to demo-a:</p> <pre><listen-address>demo-a</listen-address></pre> <p>There will be a listen address defined for each WebLogic server: AdminServer, ms1, ms2, ms3, plus the node manager configuration for machineA.</p> <p>Save the file and exit the editor.</p>
2.3	<p>Edit the <DOMAIN_ROOT>/wl_server/nodemanager/nodemanager.properties file and update the ListenAddress to point to demo-a.</p> <p>Save the file and exit the editor.</p>
3	Create a Managed Server Template
3.1	<p>You will create a managed server template using the pack command that comes with WebLogic.</p> <p>The pack command is located under <ORACLE_HOME>/oracle_common/common/bin</p>

	<p>Run the pack script with no parameters to see which options are available.</p> <p>The syntax needed to create a managed server template is:</p> <pre>./pack.sh -domain=<DOMAIN_ROOT>/wl_server -template=<LAB_HOME>/wl_server_template.jar -managed=true -template_name="wl_server managed template"</pre> <p>-<i>domain</i> specifies the domain to create the template from. -<i>template</i> specifies the location of the template you are about to create -<i>managed=true</i> specifies creating a managed server template -<i>template_name</i> specifies a descriptive template name</p> <p>Successful executing of pack should look similar to:</p> <pre>< read domain from "/u01/udemy/domains/wl_server" >> succeed: read domain from "/u01/udemy/domains/wl_server" << set config option Managed to "true" >> succeed: set config option Managed to "true" << write template to "/u01/udemy/wl_server_template.jar" >> succeed: write template to "/u01/udemy/wl_server_template.jar" << close template >> succeed: close template</pre>
3.2	<p>In a real scenario you then take the managed server template and distribute it to each server or host in your environment using a variety of means such as scp, ftp, shared network drive, etc.</p> <p>For this lab copy the managed server template to <LAB_HOME>/demo-b</p>
4	Create a Domain using the Managed Server Template
4.1	<p>Once the template has been distributed to Node B, you can now use the unpack command to create the domain structure.</p> <p>The unpack command is in the same location as the pack command.</p> <p>Run unpack with no parameters to see which options are available.</p> <p>The syntax needed to unpack a managed server template is:</p> <pre>./unpack.sh -template=<LAB_HOME>/demo- b/wl_server_template.jar -domain=<LAB_HOME>/demo- b/domains/wl_server</pre> <p>-<i>template</i> specifies the location of the managed server template on the node B -<i>domain</i> specifies the location to <DOMAIN_HOME> on node B.</p>

	<p>Your output should look similar to:</p> <pre><< read template from "/u01/udemy/demo- b/wl_server_template.jar" >> succeed: read template from "/u01/udemy/demo- b/wl_server_template.jar" << set config option DomainName to "wl_server" >> succeed: set config option DomainName to "wl_server" << write Domain to "/u01/udemy/demo-b/domains/wl_server" >> succeed: write Domain to "/u01/udemy/demo- b/domains/wl_server" << close template >> succeed: close template</pre>
4.2	<p>Verify the wl_server domain was successfully created on demo-b by navigating to <LAB_HOME>/demo-b/domains/</p> <p>The wl_server domain should exist under domains/. The contents of wl_server will look very similar to that of <DOMAIN_HOME> on Node A, however the managed servers are missing from the /servers subdirectory.</p>
5	Configure Node Manager for Node B
5.1	<p>Edit the nodemanager.properties for wl_server domain on Node B:</p> <pre><LAB_HOME>/demo- b/wl_server/nodemanager/nodemanager.properties</pre> <p>Update the property file according to Table 21 Node B Node Manager</p>
5.2	<p>Start the node manager on node b:</p> <pre><LAB_HOME>/demo-b/domains/wl_server/bin/startNodeManager.sh</pre>
6	Configure WebLogic Server for Node B
6.1	Open a new terminal window and start Node Manager and the Admin Server for the wl_server domain on Node A .
6.2	<p>Log into the admin console using the demo-a hostname created earlier.</p> <p>The admin console should be accessible at https://demo-a:9001/console</p>
6.3	Create a machine for Node B according to Table 22 Machine Parameters.
6.4	Assign ms3 to machineB from machineB's Configuration page.
7	Start Managed Server on Node B
7.1	<p>Start ms3 using the admin console. The admin console will issue a command to node manager running on Node B, however you may receive an error stating that demo-b is unreachable.</p> <p>Look at the AdminServer output in the terminal window and you may see Certificate chain validation errors. For the purposes of this lab we must disable</p>

	hostname verification.
7.2	Disable hostname verification on all servers by navigating to Environment > Servers > server_name > SSL
	Under the SSL tab, click the Advanced link
7.3	Select None for Hostname Verification and click Save
7.4	Repeat disabling hostname verification for all servers
7.5	Verify ms3 is up and running via the admin console.
7.6	Restart the AdminServer for the verification changes to take effect
7.7	Start ms3 using the admin console.
	Refresh the control page to see any status updates.
7.8	After ms3 is running, Open a terminal window and navigate to wl_server on node B.
7.9	View the contents of wl_server/servers and observe the existence of the ms3 server directory.
	This server directory is created automatically when a managed server is started for the first time.
7.10	Also note the contents of ms3/security is empty. When using Node Manager to start servers, Node Manager automatically creates a boot.properties file under: <DOMAIN_HOME>/servers/<server_name>/data/nodemanager

Lab #13: WLST

Duration 60 minutes

Skills Learned

At the end of this exercise, you will be able to:

- Understand Offline versus Online modes
- Create and configure a domain
- Navigate and manipulate MBeans
- Generate and execute WLST scripts

Overview

WebLogic includes a very powerful scripting language called WebLogic Scripting Tool or WLST. WLST is built on top of jython, who's syntax is similar to python. WLST is essentially a collection of jython modules for performing various administrative tasks against a WebLogic domain. WLST is primarily used to manipulate Mbeans.

The entire configuration of a domain is represented as a hierarchy of MBeans. There are several hundred types of MBeans, including server and application mbeans that represent WebLogic servers and deployed applications. Each type of Mbean defines a set of operations and attributes that can be invoked and introspected using WLST.

In this lab you will use WLST to create a very simple WebLogic domain. This will expose you to using WLST in both an offline and online mode. You will configure this domain by manipulating a domain template jar and creating resources by instantiating mbean instances.

This lab will expose you to these MBean hierarchies and various MBean types.

Configuration Parameters

Table 23 Domain Creation Parameters

Parameter	Value
domainTemplate	<ORACLE_HOME> /wlserver/common/templates/wls/wls.jar
domainDir	<DOMAIN_ROOT>/wlst_domain
user	weblogic

password	weblogic123
----------	-------------

Table 24 JMS Server

Parameter	Value
Name	scriptedJMSServer
PersistentStore	WseeFileStore
Target	AdminServer

Instructions

1	Launch WLST
1.1	Stop all running WebLogic processes, including Node Manager
1.2	<p>You can use WLST both interactively using a shell or using scripts that you develop.</p> <p>First let's open an interactive WLST session by launching the wlst script.</p> <p>In a terminal window execute wlst.sh:</p> <pre><ORACLE_HOME>/oracle_common/common/bin/wlst.sh</pre> <p>You will be presented with the default WLST prompt: Initializing WebLogic Scripting Tool (WLST) ...</p> <p>Welcome to WebLogic Server Administration Scripting Shell</p> <p>Type help() for help on available commands</p> <pre>wls:/offline></pre> <p>As mentioned earlier, WLST operates in 2 modes, offline and online. In offline mode, WLST is disconnected from a running WebLogic server. In offline mode, you can perform some administrative tasks, such as creating a resource or managed server. However certain administrative tasks require to be done online.</p> <p>In online mode, you use WLST to connect to a running admin server.</p>
1.3	Launch WLST using the wlst.sh script
2	Create a Domain
2.1	<p>In this section you will create a WebLogic domain using the default WLS template jar file.</p> <p>Create a domain using the WLS template. The command to do this is:</p> <pre>createDomain(domainTemplate, domainDir, user, password)</pre>

	<p>See Table 23 Domain Creation Parameters for details.</p> <p>Once the domain is created, the WLST prompt will change:</p> <pre>wls:/offline></pre>
2.2	<p>Read the new domain using the readDomain()</p> <p>The WLST prompt should change to:</p> <pre>wls:/offline/wlst_domain></pre>
2.3	<p>List the contents of the domain using ls()</p> <p>Notice there are directories and files. Each directory represents an MBean hierarchy or type.</p>
2.4	<p>Use the cd() command to navigate around the Mbean tree.</p> <pre>cd('Server') ls()</pre> <p>These commands will list the servers configured in this domain.</p> <p>The cd command is very similar to what you would use in most operating systems. You can specify absolute and relative mbean paths.</p> <p>As you may have noticed, just about every parameter in WLST is enclosed in single quotes.</p>
2.5	<p>You can change the properties of an Mbean using WLST. Let's change the listen address for the AdminServer.</p> <pre>cd('AdminServer') ls()</pre> <p>Performing an ls() will list all the configuration parameters for the Admin Server and their values. Locate the attribute for the listen address.</p> <p>Change the listen address to a valid hostname. If you have followed all steps in this lab, you should have created an /etc/hosts entry for demo-a.</p> <p>To change the value of an mbean attribute, you manipulate an object called cmo which stands for "Current managed object." It represents the mbean instance.</p> <p>The cmo automatically provides setter and getter operations for each attribute. To set a value, you simply type:</p>

	<pre>cmo.set<Attribute>('<value>')</pre> <p>where <Attribute> is the parameter you want to change</p> <p>Change the listen address to 'demo-a'</p> <pre>cmo.setListenAddress('demo-a')</pre> <p>Verify the change by typing:</p> <pre>cmo.getListenAddress()</pre>
2.6	<p>You can also use the WLST command <code>set()</code> to set values.</p> <p>Use <code>help('set')</code> to learn the syntax then set the listen port for the admin server to 7011.</p>
2.7	<p>Save the change you have made using the <code>updateDomain</code> command.</p> <pre>updateDomain() exit()</pre> <p>This command will persist any changes you have made in offline mode back to the domain configuration.</p> <p>To verify the changes were made, take a look at the admin server's listen address in <DOMAIN_HOME>/config/config.xml</p>
3	Connect WLST in Online Mode
3.1	Start the admin server for the <code>wlst_domain</code> domain.
3.2	Open a new terminal window
3.3	To use WLST in online mode you must connect to a running admin server.
	Launch WLST
3.4	<p>Type <code>help('connect')</code> to get the syntax for connecting to a WebLogic server.</p> <p>Use connect() to connect to the admin server now listening on address <code>demo-a</code> and port 7011.</p> <p>You should see a similar message when connected:</p> <pre>Connecting to t3://demo-a:7011 with userid weblogic ... Successfully connected to Admin Server "AdminServer" that belongs to domain "wlst_domain".</pre> <p>Warning: An insecure protocol was used to connect to the server. To ensure on-the-wire security, the SSL port or Admin port should be used instead.</p> <pre>wls:/wlst_domain/serverConfig></pre>

	<p>The prompt indicates the name of the domain you are connected to and which mbean hierarchy you are presently in.</p> <p>The types of actions you can perform will vary depending on which mbean hierarchy you are in.</p> <p>There are 4 primary MBean hierarchies within WebLogic:</p> <p>serverConfig is a read-only configuration MBean hierarchy for the server you are currently connected to. Configuration MBeans capture the configuration for a service, feature, resource, or application in WebLogic.</p> <p>serverRuntime is a read-only runtime MBean hierarchy for the server that you are connected to. Runtime MBeans provide runtime statistics and information for services, features, resources, or applications in WebLogic.</p> <p>domainConfig represents the configuration MBean hierarchy for the entire domain. This is where most configuration and admin tasks will take place.</p> <p>domainRuntime represents all runtime MBeans for the entire domain.</p>
3.5	<p>You switch MBean hierarchies by typing the name of the hierarchy. To switch to the read-only domain Mbean hierarchy, type:</p> <pre>domainConfig()</pre> <p>Perform an ls() and note the differences.</p>
4	Edit the Domain in Online Mode
4.1	<p>When you want to edit anything in a domain using WLST, you must first switch to the 'edit' version of the domain MBean hierarchy then start an edit session.</p> <p>To do this you type:</p> <pre>edit() startEdit()</pre> <p>Notice how the command prompt changes when you enter the edit MBean hierarchy and start an edit session:</p> <pre>wls:/wl_server/domainConfig> edit() wls:/wl_server/edit> startEdit() Starting an edit session ... Started edit session, please be sure to save and activate your changes once you are done. wls:/wl_server/edit !></pre>
4.2	<p>Create a new managed server using the create() WLST command.</p> <p>Type help('create') to get detailed syntax and example information on the create</p>

	<p>command. In WebLogic whenever you need to create anything, you create an MBean of a certain type. If you want to create a JDBC resource, you create a JDBC System Resource MBean.</p> <p>To create a managed server, you then must create an instance of a Server MBean.</p> <p>To do so the syntax is:</p> <pre>create('ms1','Server')</pre> <p>This command will create a Server MBean with the name ms1.</p> <p>So to create a managed server, type the following command:</p> <pre>create('ms1','Server')</pre>
4.3	<p>Set the listen address and port for ms4 by navigating to the Servers MBean hierarchy:</p> <pre>cd('/Servers/ms1')</pre> <p>Then set the values of ListenAddress and ListenPort to demo-a and 8011 respectively.</p> <p><i>Hint: Do not surround integer values with single quotes</i></p> <p>Once you have set the values save your work by issue the save() command.</p>
4.4	<p>Verify the changes you made using the get() command to return the values for ListenAddress and ListenPort.</p>
4.5	<p>Run the validate() command to validate your changes</p>
4.6	<p>Active all changes by using the activate() command</p>
4.7	<p>Open a new terminal window and Start ms1 using the startManagedWebLogic.sh script</p>
5	<p>Deploy an Application</p>
5.1	<p>Switch back to WLST and switch to the domain config hierarchy:</p> <pre>domainConfig()</pre>
5.2	<p>Use the deploy() command to deploy mainWebApp exploded archive directory to ms1.</p> <p>The parameters for deploy() are:</p> <p>appName = The name of the deployment path = Location of deployment file or directory targets = Comma delimited list of servers or clusters</p>

	<p>block = Whether WLST should block or return during deployment</p> <pre>wls:/wlst_domain/domainConfig> deploy (appName='mainWebApp',path='/u01/udemy/applications/wl_server/examples/build/mainWebApp',targets='ms1') Deploying application from /u01/udemy/applications/wl_server/examples/build/mainWebApp to targets ms1 (upload=false) ... <May 26, 2015 9:47:04 PM MDT> <Info> <J2EE Deployment SPI> <BEA-260121> <Initiating deploy operation for application, mainWebApp [archive: /u01/udemy/applications/wl_server/examples/build/mainWebApp], to ms1 .> .Completed the deployment of Application with status completed Current Status of your Deployment: Deployment command type: deploy Deployment State : completed Deployment Message : no message</pre>
5.3	Verify the application is deployed by using your browser to visit http://demo-a:8011
6	Manage an Application
6.1	<p>In this section you will manage an application using WLST.</p> <p>Make sure the Admin Server for wlst_domain is running and WLST is connected to the admin server.</p>
6.2	Run <code>listApplications()</code> to get a list of all deployed applications.
6.3	<p>Let's get the state of a deployed application using the <code>AppRuntimeStateRuntime</code> MBean.</p> <p>This bean is available in the <code>domainRuntime</code> hierarchy. Switch to this hierarchy by typing:</p> <pre>domainRuntime()</pre>
6.4	<p>Navigate to the <code>AppRuntimeStateRuntime</code> MBean by changing directories:</p> <pre>cd('AppRuntimeStateRuntime/AppRuntimeStateRuntime')</pre> <p>Your command prompt should look like the following:</p> <pre>wls:/wlst_domain/domainRuntime/AppRuntimeStateRuntime/AppRuntimeStateRuntime></pre>
6.5	<p>Perform a listing in this directory and notice the various attributes and operations for getting status about an application.</p> <p>Use <code>getCurrentState</code> to get the current deployment state for the <code>mainWebApp</code> application deployed on the AdminServer.</p> <pre>cmo.getCurrentState('mainWebApp','ms1')</pre>

6.6	<p>Since mainWebApp is deployed to the demoCluster in this domain, we do not need the application deployed on the AdminServer.</p> <p>Let's undeploy the application using the undeploy command. You do this from the domainConfig MBean hierarchy.</p> <p>Switch to the domainConfig hierarchy.</p>
6.7	<p>To undeploy an application, you specify the name of the application and the target to remove it from.</p> <p>Run help('undeploy') to get the syntax for undeploying an application.</p> <p>Run undeploy() to remove mainWebApp from the AdminServer.</p> <p>The output should appear as follows:</p> <pre>Undeploying application mainWebApp ... <May 6, 2015 8:57:54 PM MDT> <Info> <J2EE Deployment SPI> <BEA-260121> <Initiating undeploy operation for application, mainWebApp [archive: null], to AdminServer .> .Completed the undeployment of Application with status completed Current Status of your Deployment: Deployment command type: undeploy Deployment State : completed Deployment Message : no message wls:/wl_server/domainConfig></pre>
7	Using Record to Capture Tasks to WLST
7.1	<p>WebLogic comes with a WLST record feature that allows you to record the actions you take in the admin console and writes them to a WLST script.</p> <p>The record feature is an excellent way to quickly generate scripts for performing administrative tasks. Keep in mind however that the WLST generated by WLST can be verbose at times. The record feature is also an excellent method for learning.</p> <p>In this section you will record creating a new JMS server.</p>
7.2	Log into the admin console
7.3	<p>Click the Record button located in the top menu bar, near the very top of the page.</p> <p>WebLogic will then display a message tell you the location of the script it will create. Be sure to write this down.</p> <p>The recording session has started. Recording to /u01/udemy/domains/wlst_domain/Script1430969334443.py.</p>
7.4	Create a new JMS server according to Table 24 JMS Server.
7.5	Once the JMS server has been created, cancel the recoding by clicking the Record button again.

	<p>You will see a confirmation message stating the recording session has ended:</p> <p>The recording session has ended. Script recorded to /u01/udemy/domains/wlst_domain/Script1430969334443.py.</p>
7.6	Stop the recording session by clicking the Record button again.
7.7	<p>Open a terminal window and view the contents of the generated script.</p> <p>By default all recorded scripts end up in <DOMAIN_HOME>.</p>
7.8	Back in the console, delete this JMS server. You will recreate it using the generated WLST script.
7.9	<p>Before we can execute the generated script we edit the file to include some additional tasks. The generated script assumes you have an 'edit' session started and that you are already connected to the admin server.</p> <p>Edit the generated script to including connecting to the admin server and starting an edit session and activating changes.</p> <p>Your script should look like the following (beware of text wrapping in the lab guide).</p> <pre>connect('weblogic','weblogic123','t3://demo-a:7011') edit() startEdit() cd('/') cmo.createJMSServer('scriptedJMSServer') cd('/JMSservers/scriptedJMSServer') cmo.setPersistentStore(getMBean('/FileStores/WseeFileStore')) set('Targets',jarray.array([ObjectName('com.bea:Name=AdminServer,Type=Server')], ObjectName)) save() activate()</pre> <p>Save the file.</p>
7.10	<p>Invoke WLST passing in the name of the script</p> <pre>./wlst.sh <path_to_script></pre> <p>You should see the following output:</p> <pre>Starting an edit session ... Started edit session, please be sure to save and activate your changes once you are done.</pre>

	<p>Saving all your changes ...</p> <p>Saved all your changes successfully.</p> <p>Activating all your changes, this may take a while ...</p> <p>The edit lock associated with this edit session is released once the activation is completed.</p> <p>Activation completed</p>
7.11	Verify the JMS server was actually created by looking in the admin console.
8	Create a User Config File to Connect WLST
8.1	<p>Instead of embedded username and passwords in your scripts or when manually connecting to WebLogic, you can create a user config file that contains encrypted username and password.</p> <p>Navigate to <DOMAIN_ROOT>/wlst_domain</p>
8.2	Start the WLST interactive shell.
8.3	Connect to the admin server using connect()
8.4	<p>Run storeUserConfig() to generate the user config store and key. The key is used to encrypt/decrypt the username and password stored in the config store.</p> <p>It is imperative that the key is kept in a secure location.</p> <p>Specify the location and name of the config file and key file:</p> <pre>storeUserConfig('/<DOMAIN_HOME> /admin.config','<DOMAIN_HOME>/admin.key')</pre>
8.5	Disconnect from the AdminServer using the disconnect() command
8.6	<p>Re-connect to the admin server using the user config file and key:</p> <pre>connect(userConfigFile='<DOMAIN_HOME>/admin.config',userKey File='<DOMAIN_HOME>/admin.key',url='t3://demo-a:7011')</pre>

References

Create an Oracle Account

<https://login.oracle.com/mysso/signon.jsp>

JDK7 Download Page

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

JDK7 Installation Guide

<http://docs.oracle.com/javase/7/docs/webnotes/install/index.html>

Oracle WebLogic Server 12c Downloads Page

<http://www.oracle.com/technetwork/middleware/weblogic/downloads/wls-main-097127.html>

Oracle WebLogic 12.1.3 Online Documentation

<http://docs.oracle.com/middleware/1213/wls/index.html>

Derby 10.10.x Admin Guide

<https://db.apache.org/derby/docs/10.10/adminguide/index.html>

Oracle WebLogic Server Plug-in Downloads Page

<http://www.oracle.com/technetwork/middleware/webtier/overview/index.html>

Configure HttpClusterServlet

<https://docs.oracle.com/middleware/1213/wls/CLUST/setup.htm#i760227>

WebLogic MBean Reference

<https://docs.oracle.com/middleware/1213/wls/WLMBR/core/index.html>

WLST Command Reference

<https://docs.oracle.com/middleware/1213/wls/WLSTC/intro.htm#WLSTC107>