WebLogic Server 11gR1 Clustering Labs

Introduction

The following hands-on labs are intended to provide an introduction to some of the main features of WebLogic Server clusters in WLS 11gR1. The labs are intended to give you practice in configuring and using WebLogic Server cluster technology, through the Admin Console and online WLST scripts, with a mixture of webapp and standalone client applications.

There are 9 labs in all. These cover:

1. Configuring WebLogic Server Domains and Clusters
2. Using WLS as a proxy server (
3. Using Apache HTTP Server 2.2 as a proxy server
4. Using in-memory HTTP session state replication
5. Using JDBC persistent HTTP session state replication (UOW)
6. Clustering stateful session EJBs
7. Load balancing with distributed JMS destinations
8. Automatic migration of JMS services
9. Automatic migration of JMS Store-and-Forward services

The lab materials are included in a zipped archive file. You can unzip this and store it anywhere on your local machine, this folder will be referred below as %LAB_HOME%. In it, you will find a number of folders:

Apache – plugins and sample Apache httpd.conf
Apps – deployable client/test applications
Build – source modules for client/test applications
Scripts – WLST scripts which replicate the lab configurations
SQL – SQL scripts for session replication and database leasing
Templates – Template jar files for creating WLS domains for the labs
Shortcuts – A number of useful MS-Windows shortcuts for starting servers etc.

Please note that the shortcuts provided assume a WLS installation directory of c:\wls103 – you may need to modify these for your local environment). You will also need to install Oracle DB 10g Express Edition (other Oracle database versions are fine) and the Apache HTTP Server:
http://httpd.apache.org/download.cgi

Lab Setup – Creating a WebLogic Server domain

In this part of the lab, you will use the WebLogic Configuration Wizard to build a domain containing an Admin Server and several Managed Servers, which we will use for the remaining lab activities.

1. Start the Configuration Wizard from the Windows Start Programs menu: Oracle WebLogic -> WebLogic Server 11gR1 -> Tools -> Configuration Wizard
2. Create a new WebLogic domain
3. Base this domain on an existing template: %LAB_HOME%\Templates\WLS103ClusterLab.jar
4. Domain name = ClusterLab, keep default domain location
5. Keep default administration credentials (weblogic/weblogic)
6. Check Development mode and JRockit JDK on the next page
7. To familiarize yourself with the servers/machines that are inherited from the domain template, check Administration Server and Managed Servers, Clusters and Machines checkboxes on the next step.
8. Click 'Next' until the Configuration Summary page, then click Create.
9. Click 'Done'

Lab Setup – Starting/Stopping Servers

For this lab, we will run the WebLogic Server instances inside command shell windows so that you can easily view the server log files and output. To make it easy to do this, a number of command shell windows have been configured in the Shortcuts folder – you may want to copy this to your desktop. These have been configured to use the domain name and location suggested above; you can change these if you prefer to use a different domain name. You will also need to reconfigure these shortcuts if your WLS installation directory is not "c:\wls103" – you can of course install WebLogic Server wherever you like.

To start the servers, double-click on the Admin Server command shell and wait until the server state changes to RUNNING. If you want to verify your domain creation, do the same for:

- Managed Server 1
- Managed Server 2
- Managed Server 3
- Proxy Server

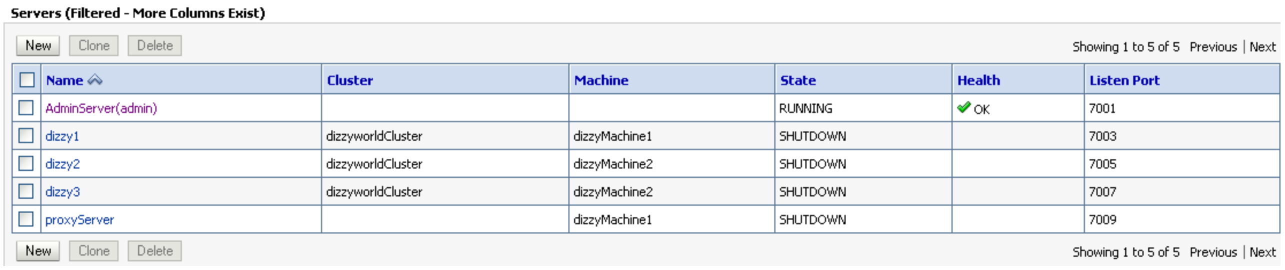
To stop a server, you use Ctrl-C in its command window. You can also start and stop servers from the WebLogic Admin Console, but you will need first to run the WebLogic Node Manager on your local machine by double-clicking on the Start Node Manager shortcut. If you wish, you can install Node Manager as a Windows service: use the installNodeMgrSvc.cmd command file in the %MIDDLEWARE_HOME%/wlserver_10.3/server/bin directory. You can use the stopManagedWeblogic.cmd command to stop managed servers, but we won't cover that in these labs.

Lab 1 – Creating a WebLogic Cluster

If you have already started Managed Server 1, 2 or 3 stop these servers now as a running server cannot be configured to join a newly-created cluster. However, you want the Admin Server for the domain to be running so that you can make configuration changes to the domain.

In this exercise, we will create a WebLogic cluster, using the three managed servers (called dizzy1, dizzy2 and dizzy3) that you created using the domain Configuration Wizard.

1. Open a browser window and login to the Admin Console (user: weblogic, password: weblogic): http://localhost:7001/console
2. navigate to Environment -> Clusters in the Domain Structure
3. Create a new cluster:
   Name: dizzyworldCluster
   Messaging Mode: Unicast
4. Click on the newly created dizzyworldCluster and navigate to Configuration -> Servers
5. Add dizzy1, dizzy2 and dizzy3 to dizzyworldCluster
6. Navigate to Environment -> Servers in the Domain Structure; you will now see that dizzy1, dizzy2 and dizzy3 are all members of the dizzyworldCluster, as shown in the screenshot below:


Servers (Filtered - More Columns Exist)

	Name	Cluster	Machine	State	Health	Listen Port
☐	AdminServer(admin)			RUNNING	✓ OK	7001
☐	dizzy1	dizzyworldCluster	dizzyMachine1	SHUTDOWN		7003
☐	dizzy2	dizzyworldCluster	dizzyMachine2	SHUTDOWN		7005
☐	dizzy3	dizzyworldCluster	dizzyMachine2	SHUTDOWN		7007
☐	proxyServer		dizzyMachine1	SHUTDOWN		7009

7. Start the dizzy1, dizzy2 and dizzy3 managed servers by double-clicking on the shortcuts provided in %LAB_HOME%\Shortcuts.. Examine the log file output for these servers and you should see the following:

```
####<Feb 8, 2011 9:04:04 PM MSK> <Notice> <Cluster> <BEA-000197> <Listening for announcements from cluster using unicast cluster messaging>
####<Feb 8, 2011 9:04:04 PM MSK> <Notice> <Cluster> <BEA-000133> <Waiting to synchronize with other running members of dizzyworldCluster.>
####<Feb 8, 2011 9:04:34 PM MSK> <Notice> <WebLogicServer> <BEA-000365> <Server state changed to ADMIN>
####<Feb 8, 2011 9:04:34 PM MSK> <Notice> <WebLogicServer> <Server state changed to RESUMING>
####<Feb 8, 2011 9:04:34 PM MSK> <Notice> <Cluster> <BEA-000162> <Starting "async" replication service with remote cluster address "null">
####<Feb 8, 2011 9:04:34 PM MSK> <Info> <Server> <BEA-002610> <Dynamic Listener Service initialized.>
```

```
####<Feb 8, 2011 9:04:34 PM MSK> <Notice> <Server> <Channel "Default" is now listening on
127.0.0.1:7003 for protocols iiop, t3, CLUSTER-BROADCAST, ldap, snmp, http.>
####<Feb 8, 2011 9:04:34 PM MSK> <Notice> <WebLogicServer> <BEA-000332> <Started WebLogic
Managed Server "dizzy1" for domain "ClusterLab" running in Development Mode>
####<Feb 8, 2011 9:04:34 PM MSK> <Info> <Server><BEA-002635> <The server "AdminServer"
connected to this server.>
...
####<Feb 8, 2011 9:04:39 PM MSK> <Notice> <WebLogicServer> <BEA-000365> <Server state
changed to RUNNING>
####<Feb 8, 2011 9:04:39 PM MSK> <Notice> <WebLogicServer> <BEA-000360> <Server started in
RUNNING mode>
```

NOTE: You may use the %LAB_HOME%/Scripts/CreateCluster.py WLST script to perform this configuration task if you prefer, or you are short of time.

Lab 2 – Configuring WLS as a Proxy Server

In this exercise, you will configure WLS to act as a proxy server for clustered web applications. In the next exercise, you will learn how to configure the Apache HTTP Server 2.2 to perform the same task.

1. Deploy the sample web application provided: %LAB_HOME%/Apps/browseStore.war. You can do this
   - either from the Admin Console, by navigating to the Deployments page, browsing to %LAB_HOME%/Apps, selecting browseStore.war and accepting defaults and target is to the dizzyworldCluster
   - or using the command-line weblogic.Deployer utility by opening a command shell, setting your domain-specific environment variables by running the script %DOMAIN_HOME%\bin\setDomainEnv.cmd (you can also run WebLogic Shell.lnk), cd to %LAB_HOME%/Apps and then launch the following command:
     ```
     java weblogic.Deployer -username weblogic -password weblogic -targets dizzyworldCluster -deploy browseStore.war
     ```
     You should see the following output, indicating that the application has been successfully deployed to the cluster:

   You can also test the application by pointing your browser directly at the individual cluster members (e.g. http://localhost:7003/browseStore/). However, this is not the behaviour we want, since we prefer the cluster operation to be transparent to users of the application.

2. Now we will configure the HttpClusterServlet proxy service to run on a separate WebLogic Server instance which will act a Proxy Server. You created this server as part of the initial domain configuration – the server is name proxyServer and it is configured to listen on port 7009 (in real-life, you would probably use the standard port 80, but we will be using that for the Apache HTTP Server, so we will keep 7009 for proxyServer).

3. We will deploy the Proxy Service as an application (proxyApp.war). You can examine the configuration by looking at the Build/proxyApp/WEB-INF folder and opening the web.xml and weblogic.xml deployment descriptors. The web.xml DD contains the following elements:

```
 <?xml version="1.0" encoding="UTF-8" ?>
- <web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
- <servlet>
  <servlet-name>HttpClusterServlet</servlet-name>
  <servlet-class>weblogic.servlet.proxy.HttpClusterServlet</servlet-class>
- <init-param>
  <param-name>WebLogicCluster</param-name>
```

```
<param-value>localhost:7003|localhost:7005|localhost:7007</param-
  value>
  </init-param>
  </servlet>
- <servlet-mapping>
<servlet-name>HttpClusterServlet</servlet-name>
<url-pattern>/</url-pattern>
  </servlet-mapping>
- <servlet-mapping>
<servlet-name>HttpClusterServlet</servlet-name>
<url-pattern>*.jsp</url-pattern>
  </servlet-mapping>
- <servlet-mapping>
<servlet-name>HttpClusterServlet</servlet-name>
<url-pattern>*.htm</url-pattern>
  </servlet-mapping>
- <servlet-mapping>
<servlet-name>HttpClusterServlet</servlet-name>
<url-pattern>*.html</url-pattern>
  </servlet-mapping>
  </web-app>
```

This configuration tells WebLogic Server to route any requests not explicitly matched by any other rule to the dizzyworldCluster servers. The weblogic.xml DD sets the context root for the HttpClusterServlet as follows:

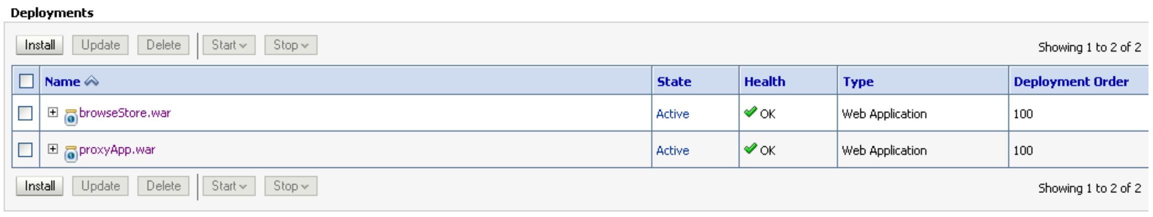
```
  <?xml version="1.0" encoding="UTF-8" ?>
- <weblogic-web-app xmlns="http://www.bea.com/ns/weblogic/90"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <context-root>/</context-root>
    </weblogic-web-app>
```

4. If you want to try out different configurations for HttpClusterServlet, you can do so by editing these DDs and repackaging the proxyApp.war webapp; then change to the Build/proxyApp folder and using the command:

   ```
   Jar –cf ..\proxyApp.war *
   ```

5. For now, you may prefer to use the pre-built proxyApp.war that you can find in the Apps folder. Start proxyServer ManagedServer and deploy proxyApp.war using the console or the following command (don't forget to set your environment variables with the setDomainEnv.cmd script) and cd to %LAB_HOME%\Apps folder:
   ```
   java weblogic.Deployer -username weblogic -password weblogic -targets
   proxyServer -deploy proxyApp.war
   ```

6. You can verify that the application has been deployed by looking at the Deployments page in the WebLogic Admin Console:



Deployments

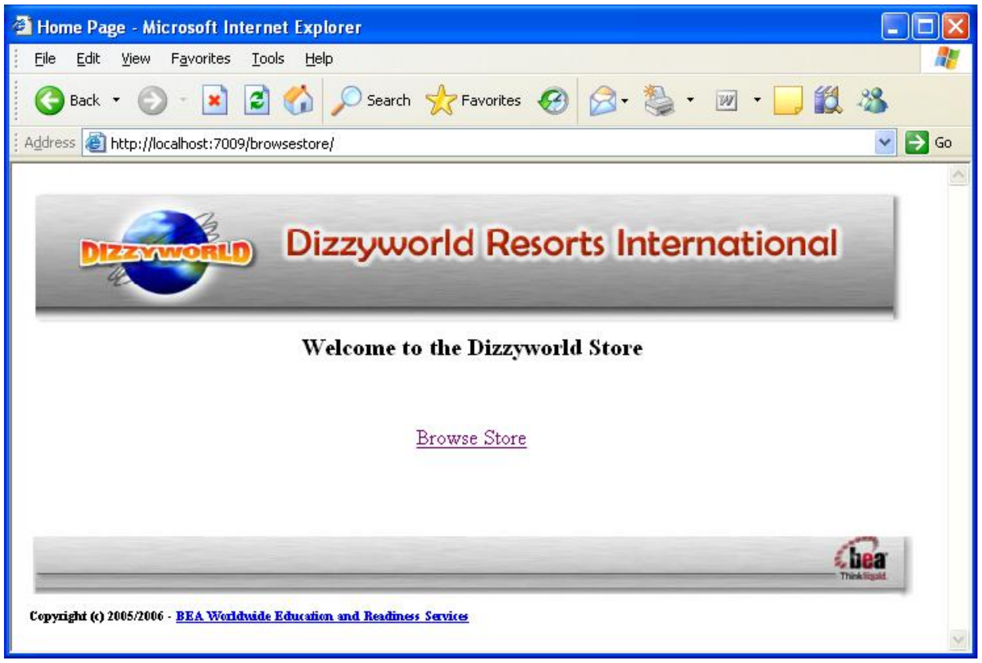
Install | Update | Delete | Start | Stop — Showing 1 to 2 of 2

Name	State	Health	Type	Deployment Order
browseStore.war	Active	✓ OK	Web Application	100
proxyApp.war	Active	✓ OK	Web Application	100

Install | Update | Delete | Start | Stop — Showing 1 to 2 of 2

7. Test the proxy service by entering the following URL: http://localhost:7009/browseStore/

   You should see the following application. Click on the link to browse the store and also try opening multiple browser windows to simulate multiple clients. If you look at the output from managed servers dizzy1, dizzy2 and dizzy3 you will see that the requests are being forwarded (using a round-robin algorithm) from the proxy server to the three members of the cluster.


Home Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
Go
DIZZYWORLD
bea
Think liquid

http://localhost:7009/browsestore/

Dizzyworld Resorts International

Welcome to the Dizzyworld Store

Browse Store

Copyright (c) 2005/2006 - BEA Worldwide Education and Readiness Services

Lab 3 – Configuring the Apache HTTP Proxy Server

In this exercise, you will learn how to configure the Apache HTTP Server to work as a proxy server with a WebLogic Server 10.3 cluster. Many customers do in fact use the Apache HTTP Server in this way and the Apache software can be downloaded from http://httpd.apache.org/. With version 2.2 of the Apache software, the commands to configure and control the HTTP Server are all available from the Windows Start Program menu and it is a simple job to configure the server to work with WebLogic Server.

1. Copy the file mod_wl_22.so from %LAB_HOME%\Apache to the %APACHE_INSTALL_DIR\modules..This is the WebLogic plug-in module for the Apache HTTP Server 2.2 and it can be downloaded as part of the package available from http://www.oracle.com/technology/products/weblogic/index.html.

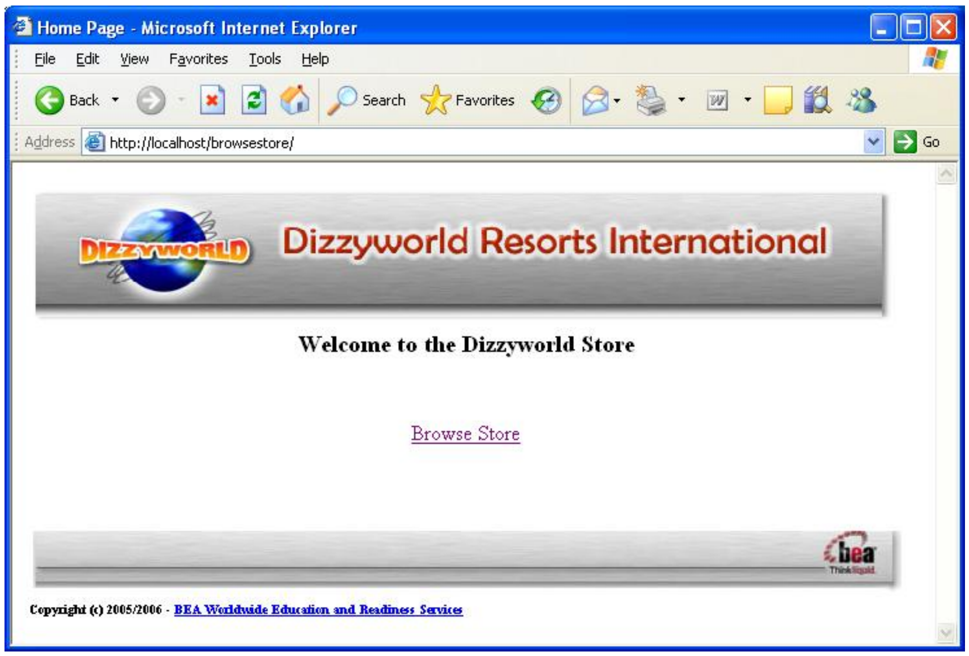
2. Go to %APACHE_INSTALL_DIR\conf to edit the Apache httpd.conf Configuration File. It is necessary to update the configuration file to include the WebLogic plug-in module; please update the file as follows or verify the contents if this has already been done (**bold type** indicates lines that need to be added to the default httpd.conf):

```
…
#LoadModule usertrack_module modules/mod_usertrack.so
#LoadModule version_module modules/mod_version.so
#LoadModule vhost_alias_module modules/mod_vhost_alias.so
LoadModule weblogic_module modules/mod_wl_22.so

<IfModule mod_weblogic.c>
    WebLogicCluster 127.0.0.1:7003,127.0.0.1:7005,127.0.0.1:7007
    MatchExpression /*
</IfModule>
<Location /weblogic>
    SetHandler weblogic-handler
    WebLogicCluster 127.0.0.1:7003,127.0.0.1:7005,127.0.0.1:7007
    DebugConfigInfo ON
    PathTrim /weblogic
</Location>

<IfModule !mpm_netware_module>
<IfModule !mpm_winnt_module>
…
```

3. This configuration will proxy any web URLs to the WebLogic Cluster running on the local machine at ports 7003 (dizzy1), 7005 (dizzy2) and 7007 (dizzy3). For more information on how to configure the Apache HTTP Server to work with WebLogic Server, or how to cover other versions of Apache or other commonly-used HTTP Servers, please consult the online documentation at: http://download.oracle.com/docs/cd/E17904_01/web.1111/e14395/toc.htm .

4. When you have verified that mod_wl_22.so has been copied to the Apache modules directory and the Apache httpd.conf file has been configured as per step 3. above, you should verify the configuration by going to the Windows Start Program menu and selecting Apache HTTP Server 2.2 -> Configure Apache Server -> Test configuration. If the test is successful (the window will appear briefly and then disappear if there are no errors), then you should restart the Apache Server using Apache HTTP Server 2.2 -> Control Apache Server -> Restart.

5. Test the Apache proxy server by entering the following URL into a web browser: http://localhost/browseStore/ . You should see the familiar Dizzyworld Store application:


Home Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/browsestore/ Go
DIZZYWORLD
bea

Dizzyworld Resorts International

Welcome to the Dizzyworld Store

Browse Store

Copyright (c) 2005/2006 - BEA Worldwide Education and Readiness Services

6. The Apache Server is configured by default to listen on port 80 and all requests are proxied through to your dizzyworldCluster WebLogic Server cluster. Note that you now have both proxyServer (listening on port 7009) and Apache httpd (listening on port 80) acting as proxy servers for the WebLogic Server cluster – you can use either in the following exercises.

Lab 4 – In-Memory Session State Replication

In this exercise, you will learn how to configure web applications to take advantage of in-memory session state replication.

1. Before packaging and deploying the web application, we will configure the servers in the WebLogic Server to use replication groups to control how session state is failed over form one server to another. Since all our servers are running on the same physical server, this is included only to illustrate how replications groups work. However, in real-life deployments, you would use replication groups to ensure that wherever possible server state is replicated on a separate physical server machine. When we set up out original domain configuration, we defined two machines (dizzyMachine1 and dizzyMachine2) both of which run on your laptop (localhost). We will configure two replication groups using these machine definitions.

2. Open the WebLogic Administration Console (http://localhost:7001/console) and login as weblogic/weblogic. Navigate to Environment -> Servers -> dizzy1 -> Configuration -> Cluster. Set the following properties:
   - Replication Group: dizzyRepGroup1
   - Preferred Secondary Group: dizzyRepGroup2

   Do the same for Servers dizzy2:
   - Replication Group: dizzyRepGroup1
   - Preferred Secondary Group: dizzyRepGroup2

   Do the same for Server dizzy3:
   - Replication Group: dizzyRepGroup2
   - Preferred Secondary Group: dizzyRepGroup1

This tells WebLogic Server to try to store the secondary state information for dizzy1 and dizzy2 on dizzy3 and vice-versa. You will be able to experiment later to view this behaviour in action.

NOTE: You may use the %LAB_HOME%Scripts/CreateRepGroups.py WLST script to perform this configuration task if you prefer, or you are short of time.

3. After configuring replication groups for the WebLogic Server cluster, you need to restart the servers dizzy1, dizzy2 and dizzy3 for this change to take effect. Please do this now.

4. We will use a simple shopping cart application to illustrate how to configure web applications to use session state replication with WebLogic clusters. You can find the JSP and other source files for this application in the %LAB_HOME%\Build\ In-Memory\ ShoppingCart. Open the WEB-INF directory and examine the weblogic.xml deployment descriptor file, which should look like this:

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <weblogic-web-app xmlns="http://www.bea.com/ns/weblogic/90"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
- <session-descriptor>
  <timeout-secs>300</timeout-secs>
  <invalidation-interval-secs>60</invalidation-interval-secs>
  <persistent-store-type>replicated_if_clustered</persistent-store-type>
    </session-descriptor>
    </weblogic-web-app>
```

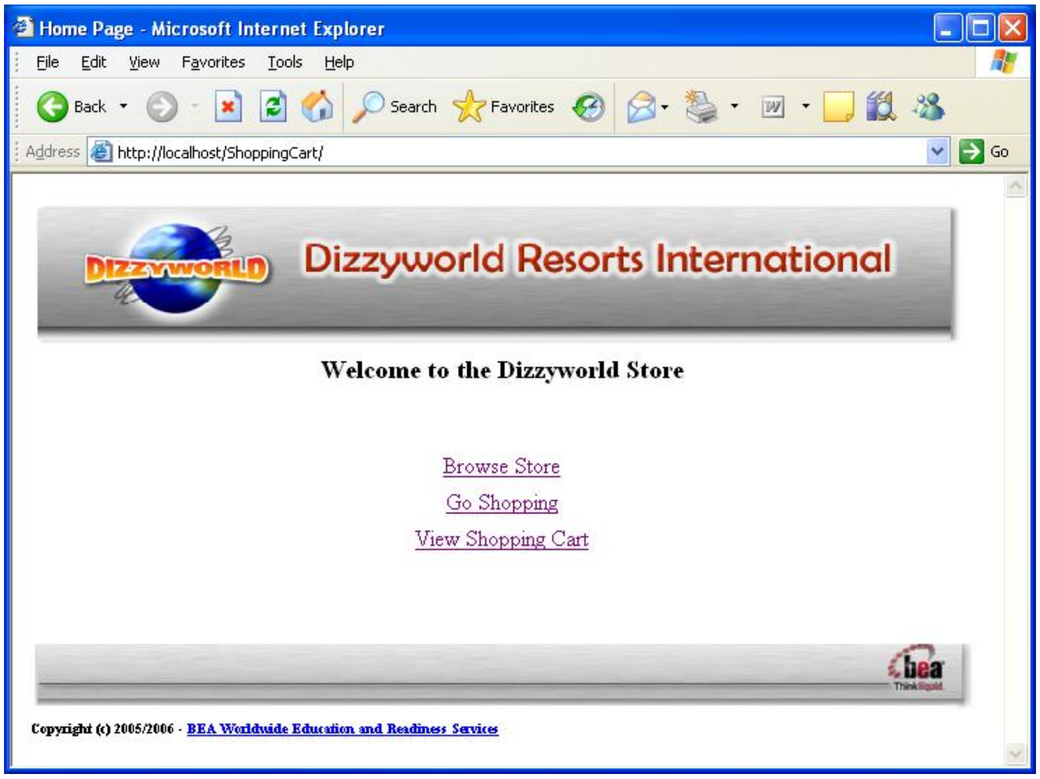
To package the application for deployment to the cluster, open a Windows command shell window and set your environment using %DOMAIN_HOME%\bin\SetDomainEnv.cmd, then change directory to %LAB_HOME%\Build\In-Memory\ShoppingCart. Package the web app for deployment with:

```
Jar -cf ../ShoppingCart.war *
```

You will find a ready-built shoppingCart.war in the ClusterLab/Apps/In-Memory folder and you can use this if you wish.

5. Deploy the ShoppingCart.war web application from the WebLogic Admin Console (target it to the whole DizzyWorldCluster) or If you prefer you can deploy the application from the command line using the following command (don’t forget to set your environment variables with the setDomainEnv.cmd script and cd to %LAB_HOME%\apps):
   ```
   java weblogic.Deployer -username weblogic -password weblogic -targets dizzyworldCluster -deploy In-Memory\ShoppingCart.war
   ```

6. Test the application by opening a browser window and entering the following URL: http://localhost/ShoppingCart (note that we are using the Apache Proxy Server to redirect the requests to the WebLogic cluster). You should see the shopping cart application:


Home Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/ShoppingCart/
Go
DIZZYWORLD
bea

Dizzyworld Resorts International

Welcome to the Dizzyworld Store

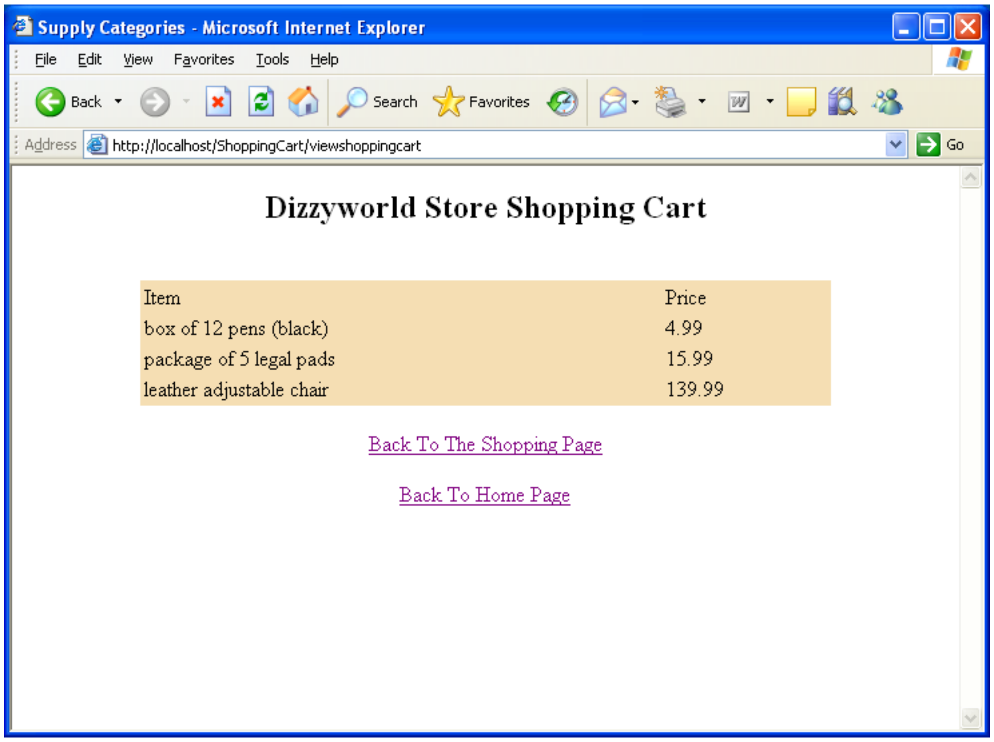
Browse Store

Go Shopping

View Shopping Cart

Copyright (c) 2005/2006 - BEA Worldwide Education and Readiness Services

7. Click on the ‘Go Shopping’ link to add items to your shopping cart: these items are stored in the session state and you can view the contents of the shopping cart (and the session state) by clicking on the ‘View Shopping Cart’ link:


Supply Categories - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/ShoppingCart/viewshoppingcart
Go

Dizzyworld Store Shopping Cart

Item	Price
box of 12 pens (black)	4.99
package of 5 legal pads	15.99
leather adjustable chair	139.99

Back To The Shopping Page

Back To Home Page

8. If you look at the log file outputs for the clustered servers dizzy1, dizzy2 and dizzy3 you will be able to see which server is the primary for a particular web session. Try opening a number of different browser windows to see how the sessions are handled by different servers within the cluster.

9. Now try simulating a machine failure by killing (Ctrc+C) the server that contained the session state. If you click on the 'View Shopping Cart' link for web sessions that were being served by that server instance, you will see how the session fails over to the secondary server in the cluster. Experiment with killing and re-starting servers and you should be able to see the effect of the replication groups that you configured earlier in this exercise. If you wish you can also experiment by changing the replication group configuration; however, you will need to restart the clustered servers after you do so.

Lab 5 – JDBC Session State Replication

In this exercise, you will learn how to configure WebLogic Server to use a JDBC persistent data store for session state replication, using an Oracle 10g database. You will need to install Oracle 10g Data Express Edition for this lab. You should create a user weblogic/weblogic, with permission to create database tables. We will use this account to configure a JDBC data source.

1. First we need to verify that the Oracle 10g XE database is correctly configured. Open a SQL Command window from the Windows Start Programs menu Oracle Database 10g Express Edition -> Run SQL Command Line. Login using the command 'conn weblogic'; you will be prompted for the password, which is 'weblogic'. We will use a database table (weblogic.WL_SERVLET_SESSIONS) to store the persistent session state information. There is a SQL script (SessionTable.sql) to create this table in %LAB_HOME%\SQL:

   ```
   CREATE TABLE "WEBLOGIC"."WL_SERVLET_SESSIONS"
   (WL_ID VARCHAR ( 100)  NOT NULL
   , WL_CONTEXT_PATH VARCHAR ( 100)  NOT NULL
   , WL_IS_NEW CHARACTER (1)
   , WL_CREATE_TIME DECIMAL ( 20)
   , WL_IS_VALID INTEGER
   , WL_SESSION_VALUES BLOB
   , WL_ACCESS_TIME DECIMAL ( 20)  NOT NULL
   , WL_MAX_INACTIVE_INTERVAL INTEGER,
   PRIMARY KEY (WL_ID, WL_CONTEXT_PATH)
    );
   ```

   You can cut and paste this script into the SQL command window to create the database table.

2. Now we will configure a JDBC data source to use when storing the session data. Open the WebLogic Administration Console in your browser and go to the Services -> Data Sources tab and create a new generic data source with the following properties:

   Name: SessionDS
   JNDI Name: SessionDS
   Database Type: Oracle
   Database Driver: Oracle's Driver (Thin) for Instance connections Versions 9.0.1 and later

   Transaction Options:
   - Supports Global Transactions
   - One-Phase Commit

Connection Properties
  Database Name: XE
  Host Name: localhost
  Port: 1521
  User Name: weblogic
  Password: weblogic

Test your connection settings on the next page by clicking Test Configuration button. If the test succeeded, press Next.
Target the data source to the whole DizzyWorldCluster and press Finish.

3. We now need to reconfigure the ShoppingCart application to use JDBC session state replication. The only change required is to the application's weblogic.xml deployment descriptor. You will find the necessary files in the %LAB_HOME%\Build\JDBC\ShoppingCart folder. Open WEB-INF/weblogic.xml and you should see the following:

```
 <?xml version="1.0" encoding="UTF-8" ?>
- <weblogic-web-app xmlns="http://www.bea.com/ns/weblogic/90"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
- <session-descriptor>
  <timeout-secs>300</timeout-secs>
  <invalidation-interval-secs>60</invalidation-interval-secs>
  <persistent-store-type>jdbc</persistent-store-type>
  <persistent-store-pool>SessionDS</persistent-store-pool>
  <persistent-store-table>WL_SERVLET_SESSIONS</persistent-store-table>
    </session-descriptor>
    </weblogic-web-app>
```

To package the application for deployment to the cluster, open a Windows command shell window and set your environment using SetDomainEnv.cmd from your domain bin directory, then change directory to %LAB_HOME%\Build\JDBC\Shopping Cart. Package the web app for deployment with:

Jar -cf ../ShoppingCart.war *

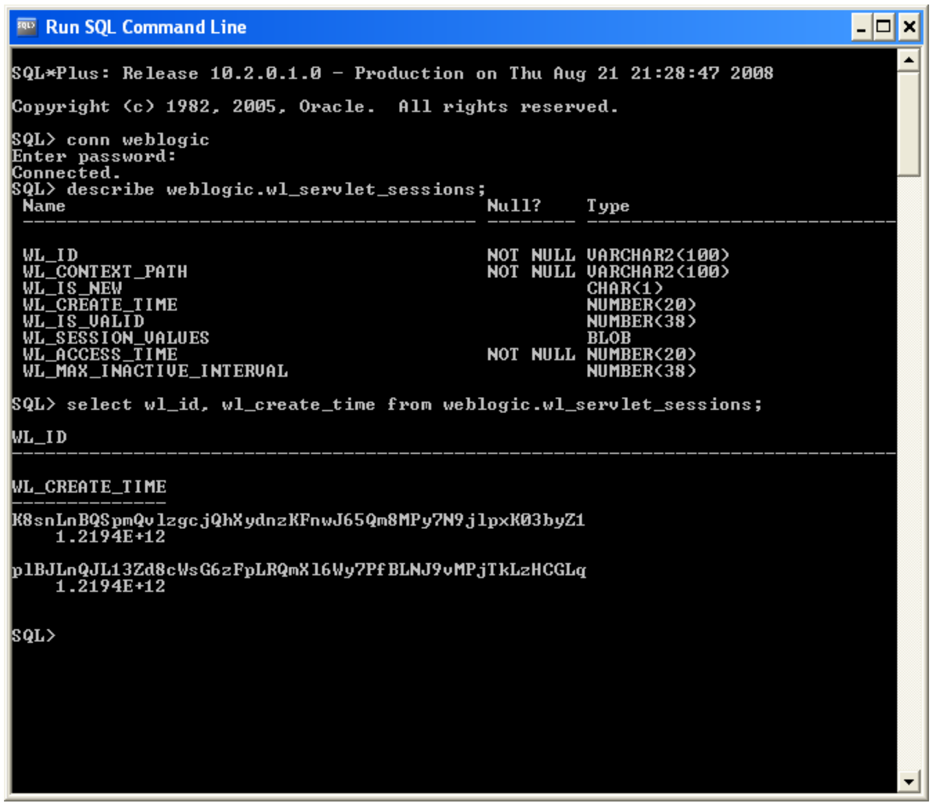
You will find a ready-built shoppingCart.war in the %LAB_HOME%/Apps/JDBC folder and you can use this if you wish.

4. Go into the WebLogic Admin Console, stop the ShoppingCart application and redeploy it from ClusterLab/AppsJDBC/ShoppingCart.war or your own version. If you prefer you can redeploy the application from the command line using the following commands (don't forget to set your environment variables with the setDomainEnv.cmd script and cd to %LAB_HOME%\Apps):

java weblogic.Deployer -username weblogic -password weblogic -targets dizzyworldCluster -undeploy -name ShoppingCart.war

java weblogic.Deployer -username weblogic -password weblogic -targets dizzyworldCluster -deploy JDBC\ShoppingCart.war

5. Test the new application by going to http://localhost/ShoppingCart and repeating some of the tests from the section above on In-Memory session replication. You can verify that session data is being stored in the JDBC data source using a SQL Command Line and the following SQL Statement:
   select WL_ID from weblogic.WL_SERVLET_SESSIONS;



```
SQL*Plus: Release 10.2.0.1.0 - Production on Thu Aug 21 21:28:47 2008

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

SQL> conn weblogic
Enter password:
Connected.
SQL> describe weblogic.wl_servlet_sessions;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 WL_ID                                     NOT NULL VARCHAR2(100)
 WL_CONTEXT_PATH                           NOT NULL VARCHAR2(100)
 WL_IS_NEW                                          CHAR(1)
 WL_CREATE_TIME                                     NUMBER(20)
 WL_IS_VALID                                        NUMBER(38)
 WL_SESSION_VALUES                                  BLOB
 WL_ACCESS_TIME                            NOT NULL NUMBER(20)
 WL_MAX_INACTIVE_INTERVAL                           NUMBER(38)

SQL> select wl_id, wl_create_time from weblogic.wl_servlet_sessions;

WL_ID
--------------------------------------------------------------------------------

WL_CREATE_TIME
--------------
K8snLnBQSpmQvlzgcjQhXydnzKFnwJ65Qm8MPy7N9jlpxK03byZ1
    1.2194E+12

plBJLnQJL13Zd8cWsG6zFpLRQmXl6Wy7PfBLNJ9vMPjTkLzHCGLq
    1.2194E+12


SQL>
```

Lab 7 – Clustering Stateful Session EJBs

In this exercise, you will use one of the sample applications distributed free with WebLogic Server 11gR1 to learn about clustering Stateful Session EJBs (SFSBs) with WLS 11gR1. In this example, a SFSB is used to simulate the behaviour of a stock trading application. Clients connect to the WebLogic cluster, perform trades and view their running account balance. State is automatically persisted (using in-memory replication) by the container between calls to the SFSB.

1. Open a command shell, and set your environment variables by running the %MIDDLEWARE_HOME%\wlserver_10.3\samples\domains\wl_server\bin\SetDomainEnv.cmd command script. Change directory to the Cluster/EJB/StatefulSession sample application directory: %MIDDLEWARE_HOME%\wlserver_10.3\samples\server\examples\src\examples\cluster\ejb\statefulSession

2. We will make some changes to the source files for the sample application to make it work with the dizzyworldCluster that we have used throughout this lab. You will find 2 files in the %LAB_HOME%/Build/SFSB directory: Client.java and TraderBean.java. Copy these files to the sample directory (you may want to rename the old files first, if you wish to keep them for later use).

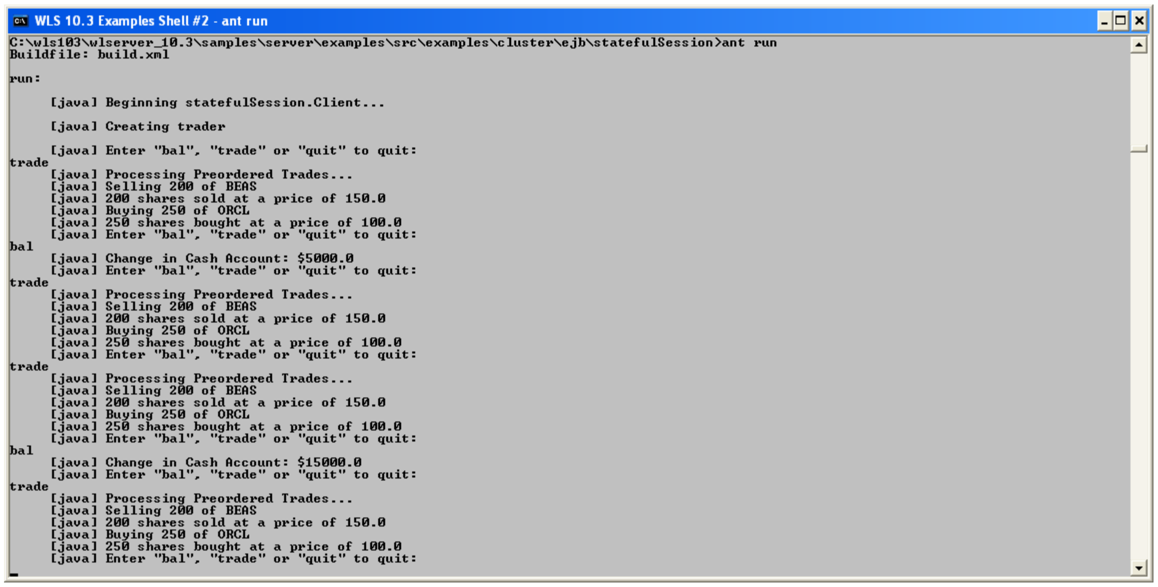
3. Build the application by typing: **ant build** and then package the client and server applications for deployment by typing: **ant deploy**

4. Deploy the SFSB application to the dizzyworldCluster (make sure all servers are running) using the following command:

   ```
   java weblogic.Deployer -username weblogic -password weblogic -targets
   dizzyworldCluster -deploy dist\cluster_ejb_stateful_session.ear
   ```

   You should see the following messages indicating that the application has been successfully deployed to the cluster:

5. Run the client application by typing: **ant run.** The client program has a simple user interface which allows you to choose whether to execute a trade, view your existing balance or quite the application as shown in the example below:


WLS 10.3 Examples Shell #2 - ant run

```
C:\wls103\wlserver_10.3\samples\server\examples\src\examples\cluster\ejb\statefulSession>ant run
Buildfile: build.xml

run:

     [java] Beginning statefulSession.Client...

     [java] Creating trader

     [java] Enter "bal", "trade" or "quit" to quit:
trade
     [java] Processing Preordered Trades...
     [java] Selling 200 of BEAS
     [java] 200 shares sold at a price of 150.0
     [java] Buying 250 of ORCL
     [java] 250 shares bought at a price of 100.0
     [java] Enter "bal", "trade" or "quit" to quit:
bal
     [java] Change in Cash Account: $5000.0
     [java] Enter "bal", "trade" or "quit" to quit:
trade
     [java] Processing Preordered Trades...
     [java] Selling 200 of BEAS
     [java] 200 shares sold at a price of 150.0
     [java] Buying 250 of ORCL
     [java] 250 shares bought at a price of 100.0
     [java] Enter "bal", "trade" or "quit" to quit:
trade
     [java] Processing Preordered Trades...
     [java] Selling 200 of BEAS
     [java] 200 shares sold at a price of 150.0
     [java] Buying 250 of ORCL
     [java] 250 shares bought at a price of 100.0
     [java] Enter "bal", "trade" or "quit" to quit:
bal
     [java] Change in Cash Account: $15000.0
     [java] Enter "bal", "trade" or "quit" to quit:
trade
     [java] Processing Preordered Trades...
     [java] Selling 200 of BEAS
     [java] 200 shares sold at a price of 150.0
     [java] Buying 250 of ORCL
     [java] 250 shares bought at a price of 100.0
     [java] Enter "bal", "trade" or "quit" to quit:
```

6. Look at the output from the clustered servers dizzy1, dizzy2 and dizzy3. Try killing a server that contains the state of the session bean and watch how the SFSB session state fails over to a secondary server – enter 'bal' on the command line to see that the account balance is maintained. Experiment with killing and restarting the servers in different orders, with multiple clients connected, to see how the replication group/machine configuration affects the failover behaviour.

Lab 8 – Load Balancing with Distributed JMS Destinations

In this exercise, you will learn how to load balance delivery of Java Message Service (JMS) messages across multiple WebLogic Server instances in a cluster using distributed destinations. JMS resources are pinned resources and so cannot be deployed to more than one server in a cluster (with the exceptions of JMS Connection Factories and Distributed Destinations). Each server handling JMS messages needs to have a JMS Server with one or more destinations (queues or topics) defined. Load balancing messages from clients can be accomplished using distributed destinations.

In the lab, we will configure a number of JMS resources on dizzy1, dizzy2 and dizzy3 and then define a virtual destination which will result in messages being load balanced across the three physical destinations. You will gain experience in configuring WebLogic JMS. You will also have an opportunity to examine some of the tools available in the WebLogic Server Admin Console for monitoring and managing JMS queues, topics and messages.

Completing this lab requires you to configure JMS Servers, JMS Modules, JMS Resources and JMS SubDeployments on the dizzyworldCluster servers. If you are short of time or do not want to perform these configuration tasks by hand, we suggest that you consider using the CreateDistributedQueue_step1.py (covers steps 1-5) and CreateDistributedQueue_step2.py (covers steps 9-14) WLST scripts in the %LAB_HOME%\Scripts sub-folder (remember to set your environment using your domain SetDomainEnv.cmd command file.

Note on JMS Terminology:

JMS Server: JMS servers act as management containers for the queues and topics in JMS modules that are targeted to them.

JMS Module: JMS system resources are configured and stored as modules similar to standard J2EE modules. Such resources include queues, topics, connection factories, templates, destination keys, quota, distributed queues, distributed topics, foreign servers, and JMS store-and-forward (SAF) parameters.

Subdeployment: A subdeployment is a mechanism by which JMS module resources (such as queues, topics, and connection factories) are grouped and targeted to a server resource (such as JMS servers, server instances, or cluster).

1. Configure a JMS Server on dizzy1. Open the WebLogic Admin Console and navigate to Services->Messaging->JMS Servers and create a new JMS Server with the following properties:

   Name: dizzyworldJMSServer
   Persistent Store: None

Target to: dizzy1

2. Create a JMS Module, Queue and Topic on dizzy1. Navigate to Services->Messaging->JMS Modules and create a new JMS module with the following properties:

   Name: dizzyworldModule
   Descriptor File Name: dizzyworldModule
   Target to: dizzy1

3. Add a Subdeployment to the dizzyworldModule. Navigate to Services->Messaging->JMS Modules and click on the dizzyworldModule you just created. Select the Subdeployments tab and add a Subdeployment with the following properties:

   Name: dizzy1SubDeployment
   Target to: dizzyworldJMSServer

4. Add a JMS Queue to the dizzyworldModule JMS Module: Navigate to Services->Messaging->JMS Modules-> dizzyworldModule and click ‘New’ to add a JMS Resource with the following properties:

   Type of resource: Queue
   Name: dizzyworldQueue
   JNDI Name: dizzyworldQueue
   Template: None
   Subdeployment: dizzy1SubDeployment

5. Add a JMS Topic to the dizzyworldModule JMS Module. Navigate to Services->Messaging->JMS Modules-> dizzyworldModule and click ‘New’ to add a JMS Resource with the following properties:

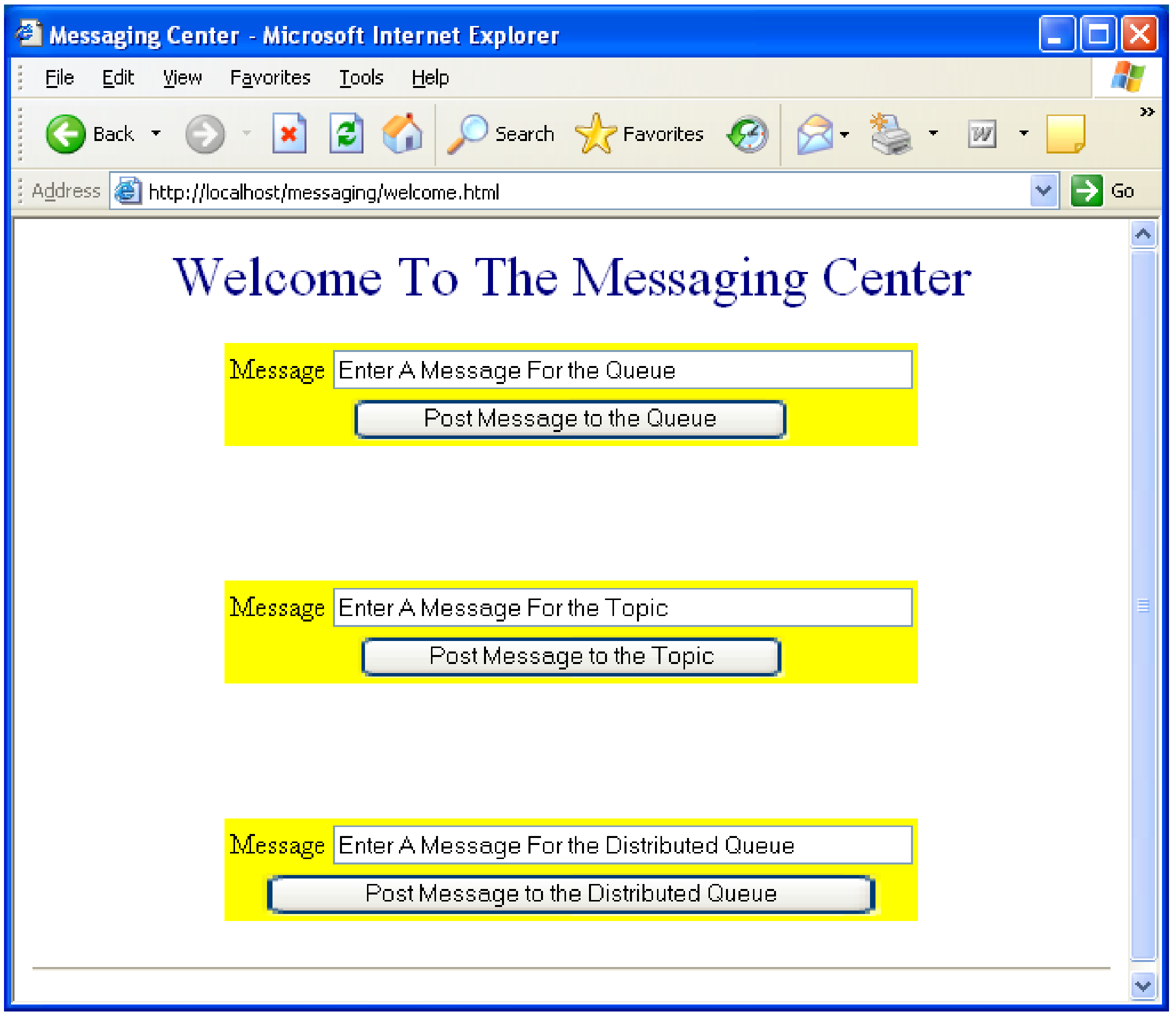
   Type of resource: Topic
   Name: dizzyworldTopic
   JNDI Name: dizzyworldTopic
   Template: None
   Subdeployment: dizzy1SubDeployment

6. If you wish, at this stage you can test the JMS configuration you just created on managed server dizzy1. You will find a JMS test client application in the %LAB_HOME%/Apps sub-folder, called messaging.war. Deploy the application using the Admin Console (deploy to all servers in the dizzyworldCluster as we will be using it later to test the distributed JMS destination) or by using the following command:

java weblogic.Deployer -username weblogic -password weblogic -targets dizzyworldCluster -deploy messaging.war

You should see the following message, indicating that the application has successfully been deployed:

7. Open a browser window and enter http://localhost/messaging to access the JMS test client application (using the Apache Proxy Server). You should see the following:



Messaging Center - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/messaging/welcome.html
Go

Welcome To The Messaging Center

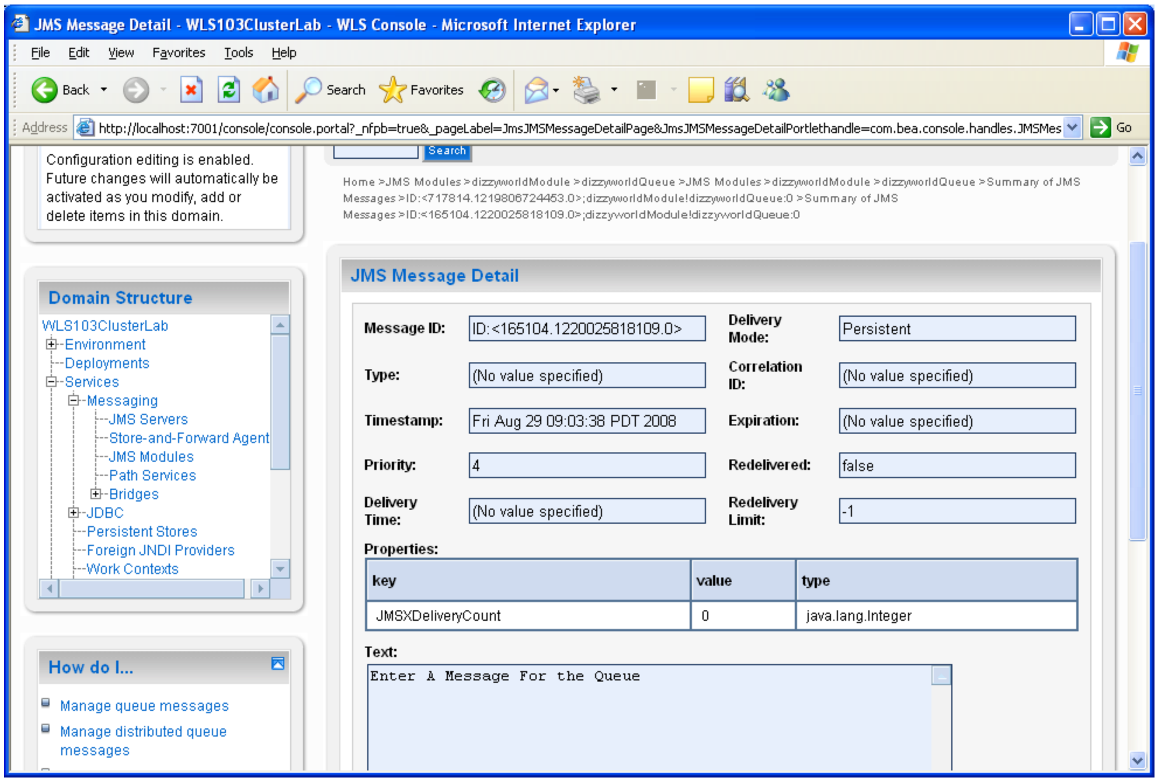
Message	Enter A Message For the Queue
	Post Message to the Queue

Message	Enter A Message For the Topic
	Post Message to the Topic

Message	Enter A Message For the Distributed Queue
	Post Message to the Distributed Queue

8. Experiment by posting a few messages to the dizzyworldQueue JMS Queue and dizzyworldTopic JMS Topic – do not try the Distributed Queue at this stage as we have not yet configured the necessary JMS Resources. If you want to see the messages in the Admin Console, navigate to Messaging->JMS Modules->dizzyworldModule, click on dizzyworldQueue, select the Monitoring tab, check the DizzyworldModule!dizzyworldQueue box and click

on ‘Show Messages’. You can now click on individual messages to see the message data as shown below:


JMS Message Detail - WLS103ClusterLab - WLS Console - Microsoft Internet Explorer

http://localhost:7001/console/console.portal?_nfpb=true&_pageLabel=JmsJMSMessageDetailPage&JmsJMSMessageDetailPortlethandle=com.bea.console.handles.JMSMes

Configuration editing is enabled. Future changes will automatically be activated as you modify, add or delete items in this domain.

Home >JMS Modules >dizzyworldModule >dizzyworldQueue >JMS Modules >dizzyworldModule >dizzyworldQueue >Summary of JMS Messages >ID:<717814.1219806724453.0>;dizzyworldModule!dizzyworldQueue:0 >Summary of JMS Messages >ID:<165104.1220025818109.0>;dizzyworldModule!dizzyworldQueue:0

Domain Structure

- WLS103ClusterLab
  - Environment
  - Deployments
  - Services
    - Messaging
      - JMS Servers
      - Store-and-Forward Agent
      - JMS Modules
      - Path Services
      - Bridges
    - JDBC
    - Persistent Stores
    - Foreign JNDI Providers
    - Work Contexts

How do I...

- Manage queue messages
- Manage distributed queue messages

JMS Message Detail

Message ID:	ID:<165104.1220025818109.0>	Delivery Mode:	Persistent
Type:	(No value specified)	Correlation ID:	(No value specified)
Timestamp:	Fri Aug 29 09:03:38 PDT 2008	Expiration:	(No value specified)
Priority:	4	Redelivered:	false
Delivery Time:	(No value specified)	Redelivery Limit:	-1

Properties:

key	value	type
JMSXDeliveryCount	0	java.lang.Integer

Text:

```
Enter A Message For the Queue
```

9. We will now extend our JMS configuration to include JMS Servers on dizzy2 and dizzy3, with associated JMS Modules and Resources.

10. Following the same procedures as before, create the following JMS Servers:

    Name: dizzyworldJMSServer2
    Target: dizzy2

    Name: dizzyworldJMSServer3
    Target: dizzy3

11. Create a new JMS Module with the following properties:

    Name: dizzyworldClusterJMSModule
    Target: dizzyworldCluster

12. Create a new JMS Subdeployment with the following properties:

    Name: dizzyworldClusterSubdeployment
    Target: dizzyworldCluster

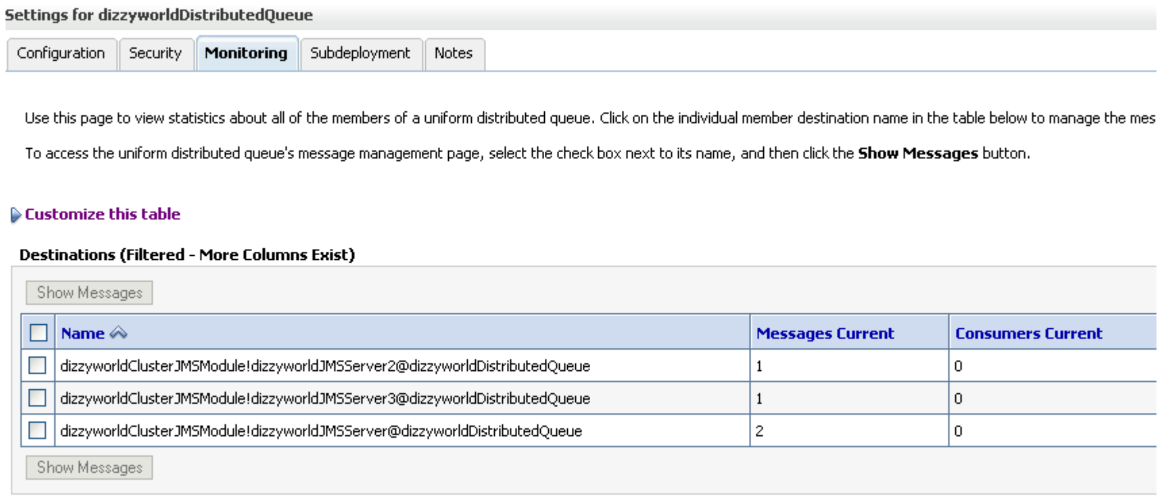
13. Create a new Distributed Queue with the following properties:

    Type: Distributed Queue
    Name: dizzyworldDistributedQueue
    JNDI Name: dizzyworldDistributedQueue
    Load-Balancing Policy: Round-Robin
    Allocate Members Uniformly: True
    [Click on Advanced Targeting]
    Target: dizzyworldClusterSubdeployment

14. Create a new Connection Factory for the JMS Distributed Queue with the following properties:

    Type: Connection Factory
    Name: dizzyworldConnectionFactory
    JNDI Name: dizzyworldConnectionFactory
    [Click on Advanced Targeting]
    Target: dizzyworldClusterSubdeployment

    When you have created the Connection Factory, disable Server Affinity by clicking on dizzyworldConnectionFactory, select the Load Balancing tab and uncheck the box for Server Affinity. This will enable you to watch the round-robin load-balancing policy in operation.
15. Now you can use the bottom button on the messaging test application to try out the dizzyworldDistributedQueue. Try sending several (at least 4) messages
16. go to the Messaging->JMS Modules -> dizzyworldClusterJMSModule-> dizzyworldDistributedQueue->Monitoring. Click Customize this table and shuttle Messages Current to the chosen columns list, press Apply.
    Now you can see your sent the messages distributed between 3 JMS Servers on dizzy1 – dizzy3.


Settings for dizzyworldDistributedQueue

Use this page to view statistics about all of the members of a uniform distributed queue. Click on the individual member destination name in the table below to manage the mes

To access the uniform distributed queue's message management page, select the check box next to its name, and then click the **Show Messages** button.

Destinations (Filtered - More Columns Exist)

Name	Messages Current	Consumers Current
dizzyworldClusterJMSModule!dizzyworldJMSServer2@dizzyworldDistributedQueue	1	0
dizzyworldClusterJMSModule!dizzyworldJMSServer3@dizzyworldDistributedQueue	1	0
dizzyworldClusterJMSModule!dizzyworldJMSServer@dizzyworldDistributedQueue	2	0

Lab 9 – Automatic Migration of JMS Services

In this section of the lab, you will see how a WebLogic Server 11gR1 Cluster can be configured to provide automatic service migration for JMS services. In this example, we will define a Migratable Target that is able to migrate across the servers that make up the dizzyworldCluster to ensure that the services it hosts (in this case, a JMS Server with its own persistent JDBC message store) are highly available. You will create a JMS queue (dizzyQueue) and connection factory (dizzyConnectionFactory), and target them at the dizzyMigratableTarget. Then you will be explore both manual and automatic service migration using a JMS client application that will allow you to send messages continuously to the queue as it migrates from server to server.
We start from creating new domain named ClusterMigrateLab based on %LAB_HOME%\Templates\ClusterLab.jar template. Start AdminServer for this new domain.
Make sure your local Oracle Database 10g XE is running. Log on as weblogic / weblogic. Run the SQL command given in %LAB_HOME%\SQL\Leasing.ddl (this is a copy of %MIDDLEWARE_HOME%\wlserver_10.3\server\db\oracle\920\leasing.ddl that is installed with WebLogic Server 11gR1). This will create the database table used by the WLS Cluster to determine which servers are active and available to host the migratable services:

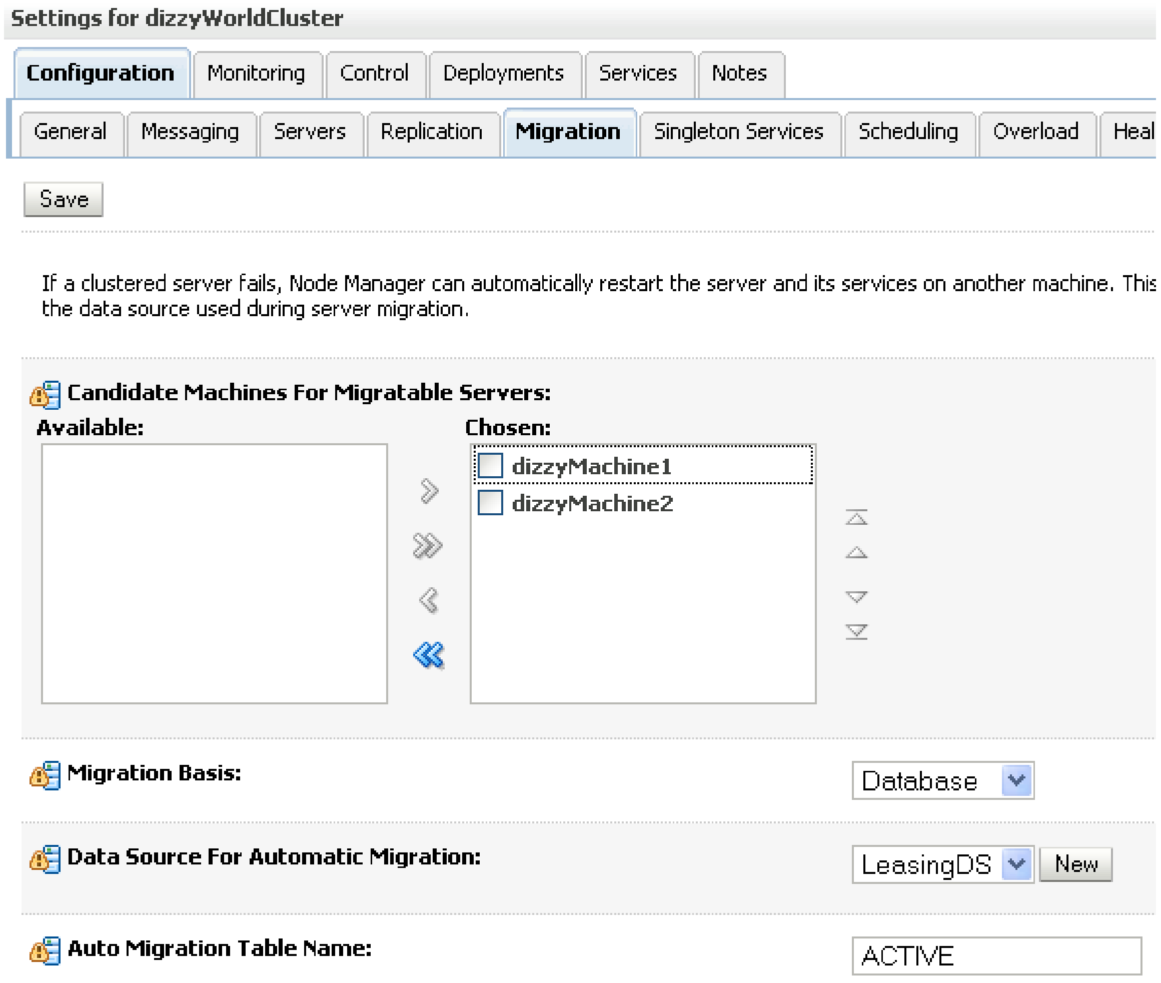
```
CREATE TABLE ACTIVE (
  SERVER VARCHAR2(150) NOT NULL,
  INSTANCE VARCHAR2(100) NOT NULL,
  DOMAINNAME VARCHAR2(50) NOT NULL,
  CLUSTERNAME VARCHAR2(50) NOT NULL,
  TIMEOUT DATE,
  PRIMARY KEY (SERVER, DOMAINNAME, CLUSTERNAME)
);
```

Create a cluster dizzyworldCluster (repeat the steps 1-6 from Lab 1).

Follow the instructions below to configure this lab manually:

1. Create a JDBC Data Source that WebLogic Server will use to access the database leasing information. Create a JDBC data source called LeasingDS,JNDI Name: LeasingDS, database type – Oracle. Select Oracle's Driver (Thin) for Instance Connections as a database driver, leave default values on the next page. Enter the following connection properties: database name - XE, host - localhost, port – 1521, user – weblogic, password – weblogic.. Test the connection and target it to all servers of the dizzyworldCluster.

2. Configure the Cluster Migration policy by selecting "Clusters" in the console navigation pane, choose dizzyworldCluster and click to the "Migration" tab. Select Migration Basis = Database, Data Source for Automatic Migration = LeasingDS, Auto Migration Table Name = ACTIVE. Specify dizzyMachine1 and dizzyMachine2 as candidate machines for migratable servers.



If a clustered server fails, Node Manager can automatically restart the server and its services on another machine. This the data source used during server migration.

Candidate Machines For Migratable Servers:

Available:

Chosen:
dizzyMachine1
dizzyMachine2

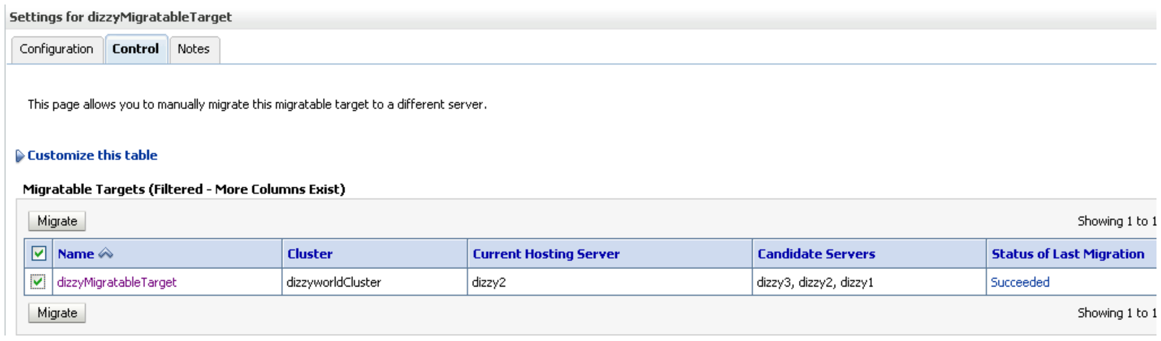
Migration Basis: Database

Data Source For Automatic Migration: LeasingDS New

Auto Migration Table Name: ACTIVE

3. Create the JDBC Data Source that will be used for the Migratable Target's persistent JMS message store. Create a JDBC data source called dizzyDS, using the Oracle Thin JDBC Driver (non-XA) with connection string localhost:1521:XE, one-phase commit and user/password = weblogic/weblogic. Test the connection and target it to all servers in the cluster.
4. Create a Migratable Target by selecting "Migratable Targets" in the console navigation pane, and click on 'New'. Create dizzyMigratableTarget targeted at the dizzyworldCluster, with a migration policy of "Auto Migrate Exactly-Once" and User-preferred server = dizzy3. Click on the newly created migratable target, go to "Migration" tab to add dizzy1, dizzy2 and dizzy3 as constrained candidate servers.

5. Create new JDBC Persistent Store: go to Services->Persistent Stores, and create new JDBC Store with the following properties: name - dizzyJDBCStore, target – dizzyMigratableTarget, data source – dizzyDS.
6. Stop all managed servers, stop and restart the Admin Server, and then start the managed servers dizzy1, dizzy2 and dizzy3. You will see the servers synchronizing with the dizzyworld cluster. When all the server are running, open the admin console again, select "Migratable Targets", choose dizzyMigratableTarget and click on the "Control" tab:



This page allows you to manually migrate this migratable target to a different server.

Migratable Targets (Filtered - More Columns Exist)

Name	Cluster	Current Hosting Server	Candidate Servers	Status of Last Migration
dizzyMigratableTarget	dizzyworldCluster	dizzy2	dizzy3, dizzy2, dizzy1	Succeeded

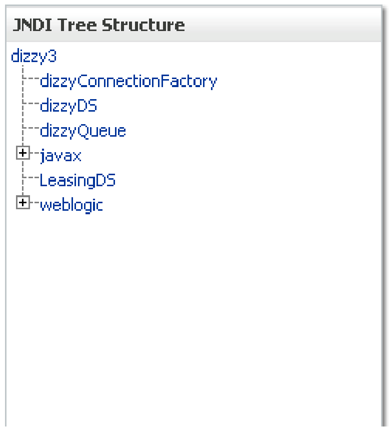
You can now see that dizzyMigratableTarget is deployed to dizzyworldCluster, is currently hosted on dizzy1 with dizzy1, dizzy2 and dizzy3 as candidate servers. Check the box and click on the "Migrate" button to perform a manual migration to another server, say dizzy2. After a few seconds, refresh the browser and you will see that dizzyMigratableTarget is now hosted on dizzy2. Next, try killing the server currently hosting the migratable target (Ctrl-C will force a shutdown) and watch dizzyMigratableTarget automatically migrate to another server. You can do this as often as you like.

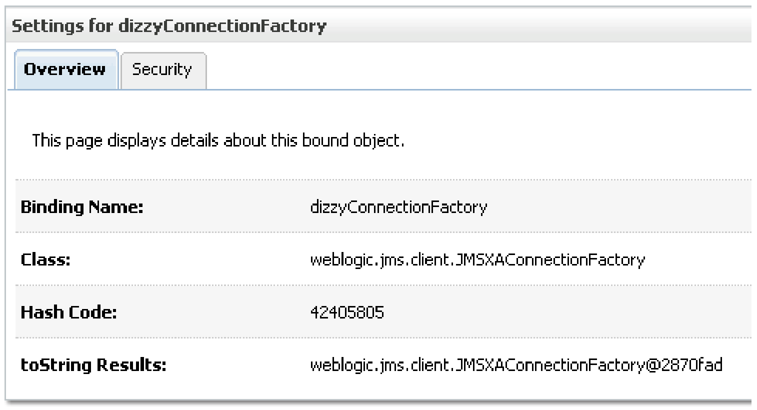
Next we will create a JMS Server and a number of JMS resources, which we will target at dizzyMigratableTarget. If you want to skip these steps, you can use the %LAB_HOME%\Scripts\CreateMigratableJMSQueue.py WLST script for this purpose.

- Create a new JMS Server called dizzyJMSServer, with dizzyJDBCStore as its persistent store, and target it to dizzyMigratableTarget. This allows the JMS Server to migrate with the migratable target. It is necessary for the persistent store to be defined and available on every server to which the JMS Server can migrate; in normal production scenarios a highly-available database solution such as Oracle RAC would be used for this purpose, but here we simply use the local XE database with a data source (dizzyDS) targeted at every server in the cluster.
- Create a new JMS Module called dizzyJMSModule, which will contain the JMS resources we wish to define. Target it to the dizzyworldCluster.

- Click on the newly created JMS Module and create a new JMS SubDeployment called dizzySubDeployment and target it to the dizzyJMSServer you just created.
- Click on dizzyJMSModule and create a new JMS Queue called dizzyQueue (JNDI: dizzyQueue), assign to the dizzySubDeployment.
- Click on dizzyJMSModule and create a new JMS Connection Factory called dizzyConnectionFactory (JNDI: dizzyConnectionFactory), assign it to the dizzySubDeployment.

Go to the “Servers” page in the admin console, select any server and inspect the JNDI tree by clicking on the “View JNDI tree” link. You will see that dizzyQueue and dizzyConnectionFactory are defined for all three servers in the cluster, no matter which server is actually hosting the migratable target. WebLogic Server maintains a cluster-wide JNDI tree, so that client objects anywhere in the cluster can access these resources, regardless of which server they are running on.





This page displays details about this bound object.

Binding Name:	dizzyConnectionFactory
Class:	weblogic.jms.client.JMSXAConnectionFactory
Hash Code:	42405805
toString Results:	weblogic.jms.client.JMSXAConnectionFactory@2870fad

To test the migratable JMS Server deployment, you can use the QueueSend client program that you will find in the %LAB_HOME%\Build\Migration folder. You will find two source files (QueueSend.java and QueueReceive.java) as well as the compiled Java class files. Take a look at QueueSend.java: it is written to allow you to send messages continuously to the dizzyQueue without having to know which server the migratable JMS Server is actually hosted on at any time. To try it out, open a shell window and run setDomainEnv.cmd from your domain’s bin directory and change directory to the %LAB_HOME%\Build\Migration folder. Run the JMS client program:
java QueueSend t3://localhost:7003,localhost:7005,localhost:7007

This sets the initial context to try to connect to the cluster using ports 7003,7005 or 7007. The program is coded to use dizzyConnectionFactory and dizzyQueue, and there is retry logic to reconnect to the cluster if the session is lost; note that the initial context will automatically try to connect to a different server if the one it is currently connected to is killed. The program will prompt you to enter a series of short text messages, which will be sent to the dizzyQueue running on the migratable target. Experiment with manual and automatic migration scenarios:

you should see the program continue without interruption as long as there is at least one server available in the cluster. To see the messages being stored on the JMS Server, open the console and go to JMS Modules->dizzyJMSModule->dizzyQueue and open the “Monitoring” tab to view or delete messages. To consume the messages, you can run the QueueReceive.java program:
Java QueueReceive t3://localhost:7003,localhost:7005,localhost:7007

Lab 10 – Automatic Migration of JMS Store-and-Forward (SAF) Services

In this lab, you will add a JMS Store-and-Forward (SAF) agent to the dizzyworld cluster, using the same Migratable Target we used in the previous lab. We will configure a SAF Agent targeted at the dizzyMigratableTarget, which will forward JMS messages to a queue hosted on a JMS server in a second WLS domain.

As the second domain we will use a domain called 'dizzyworld2' (the same domain that is used in the JMS Lab): if you have already built that domain, you can simply reuse it. Otherwise, do the following:

- there is Configuration Wizard template file you can use to create the domain in %LAB_HOME%\Templates. Start the Configuration Wizard and create a new domain based on the template file WLS103JMSLabDomain2.jar. You can choose a name for this domain - we will use 'dizzyworld2'. The domain consists of one AdminServer (listen port: 5001) and one managed server, remoteServer (listen port: 5003).
- When you have created the dizzyworld2 domain using the WebLogic Server Configuration Wizard, you should open a command shell and run the WLST script CreateSAFremote.py from the ClusterLab/Scripts folder: java weblogic.WLST CreateSAFremote.py

Run the AdminServer and the managed server named remoteServer for the dizzyworld2 domain.
Open the administration console for dizzyworld2 (http://localhost:5001/console) and take a look at the JMS services it hosts. There is one JMS Server (remoteJMSServer) which is targeted to remoteServer, and one JMS System Resources Module (remoteJMSModule), which contains a Connection Factory (remoteConnectionFactory) and a Queue (remoteQueue), both of which are part of remoteSubDeployment. We will use this queue as a SAF destination.

The steps for configuring a Store-and-Forward (SAF) Agent are covered in detail in the lab guide for the JMS Lab. Rather than repeat them, we will use a WLST script from the JMS Lab (adapted slightly to work with a cluster and migratable targets) to create the necessary resources. The script is called createMigratableSAFsource.py and you will find it in the %LAB_HOME%\Scripts folder. Note that this script references two further files (createSAFsourceSecret and createSAFsourceConfig) which must be in the same directory as the main script file; these are used to store an encrypted password for the remote WLS server. Open a shell window (set the environment with setDomainEnv.cmd), go to %LAB_HOME%\Scripts and run the following command:

```
>java weblogic.WLST createMigratableSAFsource.py
```

Open the administration console (http://localhost:7001/console) and have a look at the configuration that the WLST script has created:

JMS Server: sourceJMSServer, target: dizzyMigratableTarget
SAF Agent: sourceSAFAgent, target: dizzyMigratableTarget
JMS Module: sourceJMSModule, target: dizzyworldCluster

The JMS Resources are arranged into two subdeployments:
sourceSubDeployment: targeted to sourceJMSServer
sourceSAFSubDeployment: targeted to sourceSAFAgent

The sourceJMSModule contains the following resources:
sourceConnectionFactory (in sourceSubDeployment)
sourceSAFRemoteContext (in sourceSAFSubDeployment)
sourceSAFErrorHandling (in sourceSAFSubDeployment)
sourceSAFImportedDestinations (in sourceSAFSubDeployment)

sourceSAFImportedDestinations contains one remote JMS queue resource (remoteQueue).

Together, these JMS system resources define a new JMS server and a SAF Agent, both targeted at the dizzyMigratableTarget, that provide SAF message delivery to the remote queue. Client applications use sourceConnectionFactory to create a JMS connection and session, and remoteQueue to create the JMS destination, using the cluster-wide JNDI tree.

There are sample classes available for you to try out this configuration. Note that there is no need to restart the dizzyworld managed servers – the SAF Agent is automatically deployed and active in the cluster. The sample client classes are in the %LAB_HOME%\Build/Migration\SAF folder, with the Java source files. Open a couple of command shells, run setDomainEnv.cmd from your domain's bin directory and change directory to the %LAB_HOME%\Build\Migration\SAF folder. and try these out:

```
>java QueueSend t3://localhost:7003,localhost:7005,localhost:7007
```

```
>java QueueReceive t3://localhost:5003
```

You will be prompted to enter short text messages and you should see them appear in the QueueReceive window (which is listening to the remote queue in the other domain). As before, experiment by shutting down managed servers in the dizzyworld domain to see the migratable target move around the cluster to ensure that the SAF Agent is always available. You can use the admin console to view the location of the migratable target at any given time. If your system is short of memory, note that you do not need to have the admin servers for either domain running: the managed servers can start up and run quite independently.