**Weblogic Material**

 WebLogic Server Administration  
  
WebLogic Server: Oracle WebLogic is a server software application that runs on a middle tier, between back-end databases and related applications an browser-based thin clients. WebLogic is a leading e-commerce online transaction processing (OLTP) platform, developed to connect users in a distributed computing environment and to facilitate the integration of mainframe applications with distributed corporate data and applications.  
WebLogic server is based on Java2 Platform, Enterprise Edition (J2EE), the standard platform used to create Java-based multi-tier enterprise applications.  
  
Oracle WebLogic Server 12c is the industry's best application server for building and deploying enterprise Java EE applications with support for new features for lowering cost of operations, improving performance, enhancing scalability and supporting the Oracle Applications portfolio.  
WebLogic Server 12c R2 (12.2.1) - October 26, 2015  
WebLogic Server 12c Release 3 (12.1.3) - June 26, 2014  
WebLogic Server 12c Release 2 (12.1.2) - July 11, 2013  
WebLogic Server 12c Release 1 (12.1.1) - Dec 1, 2012  
WebLogic Server 12c (12.0) - December 1, 2011  
WebLogic Server 11gR1 PS5 (10.3.6) - February 2012  
WebLogic Server 11gR1 PS4 (10.3.5) - May 16, 2011  
WebLogic Server 11gR1 PS3 (10.3.4) - January 15, 2011  
WebLogic Server 11gR1 PS2 (10.3.3) - April 2010  
WebLogic Server 11gR1 PS1 (10.3.2) - November 2009  
WebLogic Server 11g (10.3.1) - July 2009  
WebLogic Server 10.3 - August 2008  
WebLogic Server 10.0 - March 2007  
WebLogic Server 9.2  
WebLogic Server 9.1  
WebLogic Server 9.0 - November 2006  
WebLogic Server 8.1 - July 2003  
WebLogic Server 7.0 - June 2002  
WebLogic Server 6.1  
WebLogic Server 6.0 - file date March 2001 on an old CD  
WebLogic Server 5.1 (code name: Denali) First version supporting hot deployment for applications (via command line)  
WebLogic Server 4.0  
WebLogic Tengah 3.1 - June 1998  
WebLogic Tengah 3.0.1 - March 1998  
WebLogic Tengah 3.0 - January 1998  
WebLogic Tengah - November 1997  
  
The table below lists major standards supported by WebLogic Server product version.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Standard | [WLS 7.0](http://download.oracle.com/docs/cd/E13222_01/wls/docs70/notes/new.html#1134707) | [WLS 8.1](http://download.oracle.com/docs/cd/E13222_01/wls/docs81/notes/new.html#1184292) | [WLS 9.0](http://download.oracle.com/docs/cd/E13222_01/wls/docs90/notes/new.html#1199788) | [WLS 10.0](http://download.oracle.com/docs/cd/E13222_01/wls/docs100/notes/new.html#wp1065515) | [WLS 10.3](http://download.oracle.com/docs/cd/E12840_01/wls/docs103/notes/new.html#wp1093291) | [WLS 12c](http://docs.oracle.com/cd/E24329_01/web.1211/e24494/toc.htm#BGGBIJBI) |
| Java | 1.3 | 1.4 | 5 | 5 | 6 (7 in 10.3.6+) | 7 |
| Java EE | 1.3 | 1.3 | 1.4 | 5 | 5 | 6 |
| Servlet | 1.2 | 2.3 | 2.4 | 2.5 | 2.5 | 3.0 |
| JSP | 1.2 | 1.2 | 2.0 | 2.1 | 2.1 | 2.2 |
| EJB | 2.0 | 2.0 | 2.1 | 3.0 | 3.0 | 3.1 |
| JDBC | 2.0 | 2.0 | 3.0 | 3.0 | 3.0 | 4.0 |

Difference between web server and application server:

|  |  |
| --- | --- |
| Webserver 1) Webserver handles the HTTP protocol. When    the Web server receives an HTTP request,   it responds with an HTTP response, such as    sending back an HTML page. 2) Web Server serves static HTML pages or    gifs, jpegs, etc., and can also run code written    in CGI, JSP etc. 3) Web Server only supports Servlets and JSP. | Application Server  1) It serves business logic to application     programs through any number of protocols  2) Application server is more capable of    dynamic behavior than webserver. We can   also configure application server to work   as a webserver. Simply application server is a    superset of webserver.  3)Application Server supports distributed     transaction and EJB.  4) An Application Server is used to run business logic     or dynamically generated presentation code.    It can either be .NET based or J2EE based |

Domain: An Oracle WebLogic Server administration domain is a logically related group of Oracle WebLogic Server resources. Domains include a special Oracle WebLogic Server instance called the Administration Server, which is the central point from which you configure and manage all resources in the domain. Usually, you configure a domain to include additional Oracle WebLogic Server instances called Managed Servers. You deploy Web applications, EJBs, Web Services, and other resources onto the Managed Servers and use the Administration Server for configuration and management purposes only.  
  
Node Manager: Node Manager is a WebLogic Server utility that enables you to start, shut down, and restart Administration Server and Managed Server instances from a remote location. Although Node Manager is optional, it is recommended if your WebLogic Server environment hosts applications with high availability requirements.

A Node Manager process is not associated with a specific WebLogic domain but with a machine. You can use the same Node Manager process to control server instances in any WebLogic Server domain, as long as the server instances reside on the same machine as the Node Manager process. Node Manager must run on each computer that hosts WebLogic Server instances—whether Administration Server or Managed Server—that you want to control with Node Manager.

WebLogic Server provides two versions of Node Manager, Java-based and script-based, with similar functionality. However, each version has different configuration and security considerations.

Java-based Node Manager: Java-based Node Manager runs within a Java Virtual Machine (JVM) process. It is recommended that you run it as a Windows service on Windows platforms and as an operating system service on UNIX platforms, allowing it to restart automatically when the system is rebooted.

Oracle provides native Node Manager Libraries for Windows, Solaris, HP UX, Linux on Intel, Linux on Z-Series, and AIX operating systems.

Note: Node Manager is not supported on Open VMS,OS/390,AS400,UnixWare/Tru64 UNIX.

Script-based Node Manager: For UNIX and Linux systems, WebLogic Server provides a script-based version of Node Manager. This script is based on UNIX shell scripts, but uses SSH for increased security. SSH uses user-id based security.

Note: It is recommended that you run script-based Node Manager as Operating System services, which allows to restart automatically when the system is rebooted.This version does not provide as much security as the Java-based version. However, the advantage of the script-based Node Manager is that it can remotely manage servers over a network that has been configured to use SSH. No additional server installation is required. The scripts merely have to be copied to the remote machine.

|  |  |
| --- | --- |
|  |  |

Administration Server: Admin Server is an instance of Weblogic server. The Administration Server operates as the central control entity for the configuration of the entire domain. It maintains the domain's configuration documents and distributes changes in the configuration documents to Managed Servers. You can also use the Administration Server as a central location from which to monitor all resources in a domain.

Managed server: Apart from Admin Server any weblogic server instance is called Managed server. To prevent the Administration Server from becoming a single point of failure, Managed Servers can always function without the presence a running Administration Server. When a Managed Server starts, it contacts the Administration Server to retrieve its configuration information. If a Managed Server is unable to connect to the specified Administration Server during startup, it can retrieve its configuration directly by reading a copy of the config.xml file and other files located on the Managed Server's own file system.

Cluster: A WebLogic Server cluster consists of multiple WebLogic Server instances running simultaneously and working together to provide increased scalability and reliability. A cluster appears to clients to be a single WebLogic Server instance. The server instances that constitute a cluster can run on the same machine, or be located on different machines. You can increase a cluster’s capacity by adding additional server instances to the cluster on an existing machine, or you can add machines to the cluster to host the incremental server instances. Each server instance in a cluster must run the same version of WebLogic Server.

A cluster is defined as a group of application servers that transparently run a J2EE application as if it were a single entity.  There are two methods of clustering:  vertical scaling and horizontal scaling

Horizontal clustering: It involves running multiple Java application servers that are run on two or more separate physical machines. Horizontal scaling is more reliable than vertical scaling, since there are multiple machines involved in the cluster environment, as compared to only one machine.  
  
Vertical clustering: However, consists of multiple Java application servers on a single physical machine. With vertical scaling, the machine's processing power, CPU usage, and JVM heap memory configurations are the main factors in deciding how many server instances should be run on one machine  
  
Proxy Server: Proxy Server is an intermediary server between your web browser (client) which requests for some information/data and your server (web server/Application server) that process the data.

Types of Proxy Server: They are three different types of proxy servers. They are as follows

1) Forward Proxy Servers: Forward Proxy Server is a server which forwards the request from the intranet clients (web browser) to the internet servers. These proxy servers are present in the same network of your client.

2) Open Proxy Server: An open proxy is a proxy server which is accessible by any Internet user. Any proxy server that doesn’t restrict its client base to its own set of clients and allows any other client to connect to it is known as an “Open Proxy”. An anonymous open proxy allows users to conceal their IP address while browsing the Web or using other Internet services. They are in numerous open proxy servers present in Internet. For converting any flavor of proxy servers to Open Proxy servers we just have to enable the flag “ProxyRequests On” in the configuration file.

3) Reverse Proxy Server: A Proxy Server which takes requests from external clients (web browsers) or Internet and forwards them to servers in an internal network is called as Reverse Proxy Server. Generally, the reverse proxy servers are present in the same network where we have our App/Web servers.

Advantages of using Reverse Proxy Servers: The various advantages of using the proxy servers are as follows

1)      Filtering  
2)      Caching  
3)      Bypassing filters and censorship  
4)      Logging and eavesdropping  
5)      Gateways to private networks  
6)      Accessing services anonymously

Fire wall: It provide filtering, autherization& Authentication services.

                  It can act as proxy servers.

      Mapping port requests.

Machine: A machine is a computer that host WebLogic server.

                  A machine runs a supported o.s platform.

Can host multiple WebLogic server instances.

WebLogic server Life Cycle:

Starting state: During the starting state  instances read the domain configuration data from its configuration directory. Whereas the Manager server will get their configuration data from Admin server. It is in this state that the instance the basic services such as the kernel and execute queues, the container service for logging and Node manager service. The server also deploy during this phase.

Stand by: In this state the server Instance will allow you to issue just to administrative requests. You can me the server state either running or shutdown state. Normally the server instance will automatically transition through the stand by state to next stage unless you start the instance with the start in stand by command.  
Note: All ports are closed in this stat. But you can quickly transition to a running state.

Admin mode: The admin mode permits only Administrative task, deploying applications with those applications being able to only request from users with the admin and App tester roles. Running a server in admin mode is also useful when trying to diagnose problems with application gone badly.

Note: Servers will run in admin mode when there is problem with deployed application or JDBC connection pool.we can resume the server from Admin state to resume state.

Resuming state: This is purely transitional state the server instance goes through after it transitions automatically through Admin state or you issue the resume command after first placing the instance in the stand by or Admin state. You can do this state change from command line or through the Admin console.

Running state: This is off course final state the server instance reaches after you either issue a start up command or resume command to move the server out of the Admin or stand by state. It is in the running state that the server can accept the service client request for it services.

WebLogic Installation:   
  
Installation Methods:  
  
1) Binary Installation  
This type of installer is a standalone version of the installation that includes JDKs for the selected platform. The installer is either and .exe file (for windows) or .bin file (for UNIX)  
2) Generic Jar installation  
This type of installer is a .jar file. It does not include JDK. You can use this type of installer to install the product on any supported platform where java is already installed.  
  
There are 3 types of installations

1. GUI mode.

2. Console mode.

3. Silent mode.

1) GUI Mode:

Step1: welcome screen

Step2: Accept license agreement.

Step3: Create new BEA home directory

C:\bea10.3

Step4: Choose installation- Complete (or) Custom

Step5.Select complete

Step6.Select product installation directory

          C:\bea10.3\wlserver\_10.3

2) Console Mode:

       server10.3\_win32.exe  -mode = console

3) Silent Mode:   It is a way of setting installation configuration only once and then using those configurations to duplicate the installation on many machines.

The installation programs read the settings for your configuration from an xml file.(silent.xml)

server10.3\_win32.exe -mode=silent -silent.xml=c:\bea10.3\silent.xml  -log=c:\10.3\silent.log

Step1: Create silent.xml file

<?xml version="1.0" encoding="ISO-8859-1" ?>

<bea-installer>

<input-fields>

<data-value name="BEAHOME" value="C:\bea10.3" />

<data-value name="USER\_INSTALL\_DIR" value="C:\bea10.3\wlserver\_10.3" />

</input-fields>

</bea-installer>

Step2: Save this file as silent.xml file in C:\bea10.3 folder

Step3: Copy the WebLogic software under C:\bea10.3 folder

Step4: Create a sub folder wlserver\_10.3 under c:\bea10.3

Step5: Execute the below command

server103\_win32.exe -mode=silent -silent\_xml=C:\bea10.3\ silent.xml  -log=C:\10.3\silent.log

Domain creation using configuration wizard:           

Step1: Goto the below path and execute config.cmd script

 C:\bea10.3\wlserver\_10.3\common\bin>config.cmd

Step2: Select - Create new WebLogic domain

Step3: Select - Base this domain on an existing template

Step4: Configure Administrator username and password.

Step5: Configure server start mode and JDK

Development mode   ---- SunJdk

Production mode   ----- Jrocket

Select development mode and SunJdk

Step6: Customize environment & services settings

Select yes if you would like to configure resources

Step7: Configure Admin server

Name : AdminServer

Listen address: localhost

Listen port : 7001

SSL listen:  None

\*\* Default port number for AdminServer-7001

            \*\* Default ssl listen port : 7002

Step8: Configure managed servers

name    listen address     listen port   ssl listen port

ms1      localhost              7003            None

ms2      localhost               7004           None

Step9: Configure cluster

Name Multicast address  multicast port  cluster address

c1          239.192.0.0        7010            localhost:7003, localhost:7004

Step 10: Assign servers to clusters

Step 11: Configure machines

Name   NodeManagerListenAddress   NodeManagerListenPort

m1                 localhost                                5556

Step 12: Assign servers to machines

Step 13: Create WebLogic domain

              Enter the name and location for the domain

Domain name : dev\_domain

Domain location : c:\bea10.3\user\_projects\domains

Step 14:  Click on Create

Q) How to start Admin server?

 C:\bea10.3\user\_projects\domains\dev\_domain\bin>startWebLogic.cmd

Q) How to Access Admin console?

http://localhost:7001/console

Q) Starting managed servers without node manager?

If Adminserver is in another machine: c:\bea\user\_projects\domains\webdomain\bin>startmanagedWebLogic.cmd ms1   
http://localhost:7001 (Admin server port number)

If Adminserver is in same machine: c:\bea\user\_projects\domains\maindomain\bin>startmanagedWebLogic.cmd ms1

Q) How to stop Admin server?

StopWebLogic.cmd   -----Windows

StopWebLogic.sh   -------Unix

Q) How To stop managed server?

StopManagedWebLogic.cmd ms1

Q) How to start Nodemanager?

C:\bea10\wlserver\_10.3\server\bin>startNodemanager.cmd

Creating managed servers from console:

Step1: Click on Lock& Edit

Step2: Navigate to servers and then click on new

Step3: Create a new server

Step4: Provide server properties

Server name : ms3

Server listen address: localhost

Server listen port: 7005

Step5: Select - this is a stand alone server

Step6: Review choices

Step7: Activate changes

Assigning server to cluster:

Step1: Navigate to cluster-cluster name-servers

Step2:  Click on Lock & Edit

Step3:  Add

Add server to cluster

Select existing server and add it as a cluster

Step4: Activate changes

Adding servers to machines:

Step1: Navigate to machines-m1-servers

Step2: Lock& Edit

Step3: Add

Step4: Activate changes

How to check Node manager status from console:

We can’t Start Node manager from console

Step1: Click on machines

Step2: Select m1

Step3: Click on monitoring

Clone: It creates other server with same properties.

we can clone servers machines and clusters.

we should not cluster managed server while the server is running.

-->boot.properties-it contains username and password

-->Development mode-automatically creates boot.properties file

--> Production mode-we have to create boot.properties file.

--> setDomainEnv -we can change the mode from development to production or production to development modedoing changes in this file.

 Different between development mode and production mode

|  |  |
| --- | --- |
| Development mode  1) The default JDK for development domain      is SunHotSopt.(SunJdk) 2) It uses demo certificate for SSL  3) In this mode Auto deployment is enabled.  4) Server instances rotate their log files on startup.  5) Admin server uses an automatically      created boot.properties during startup.  6) The default maximum capacity for JDBC      Data source is 15.  7) The debug flag which is used to start       Weblogic workshop debugger is enabled. | Production mode  1) The default JDK for production domain      is JRocket.  2) It is used demo certificate for SSL a warning      is displayed.  3) In this mode Auto deployment is disabled.  4) Server instances states their log files when      it reaches 5Mb  5) Admin server prompt for username      and password during startup.  6) The default maximum capacity for JDBC                 Data source is 25.  7) The debug flag which used to start      WebLogic workshop debugger is disabled. |

                                                Deployments

Deployment an application involves the following tasks:

Preparing:  Choosing weather to package the application as an archived file or keep in an exploded directory.

Configuring: Creating a deployment plan to maintain the configuration changes without changes the deployment descriptors.

Deploying: Targeting and distributing the application to servers in an OracleWebLogicServer domain.

Deploment Methods: WLS (WebLogic scripting) supports three types of deployment methods.

1.   Auto-deployment

2.  Console deployment

3.  Command-line Deployment

You can deploy:

1.   Enterprise, Web, and EJB applications

2.  Web services

3.   J2EE libraries

4.   JDBC, JMS and Diagnostic Framework modules

5.   Resource adapters

6.  Optional packages

7.  Client application archives

Applications and EJBs can be deployed:

1.   In an archived file(.ear, .war, .jar)

2.   In an exploded(open)directory format

Archive Files: There are three archive files to deployed in Oracle WebLogic Server.

1) JAR(Java Archive): Jar is java archive file. It contains all class file, image, sound and other files which will needed in whole application. Computer users can create or extract JAR files using the jar command that comes with the JDK. They can also use zip tools. The JavaTM Archive (JAR) file format enables you to bundle multiple files into a single archive file. jar was designed mainly to facilitate the packaging of java applets or applications into a single archive.

Creating Jar File:

            jar  -cvf   filename.jar “.”

Extracting Jar file:

            Jar   -xvf  filename.jar

2) WAR(web Archive): Web Archive (WAR) file is a Java archive file used to store jsp, servlets, classes, meta data information, images and Sound and tag libraries etc. It is standard file extension is .war. WAR files are used to package Web modules. A WAR file is for a Web application deployed to a servlet/jsp engine.

Creating war File:

            jar  -cvf   filename.war “.”

Extracting Jar file:

            jar   -xvf  filename.war

3)  Ear(Enterprise Archive): An Enterprise Archive file represents a J2EE application that can be deployed in a Web Sphere application server. EAR files are standard Java archive files and have the file extension .ear. EAR file contain ejb, web or application client module. ear file is complete j2ee application file that contain all(jar +war)

Creating Ear File:

            jar  -cvf   filename.ear “.”

Extracting Ear file:

            Jar   -xvf  filename.ear

In shot we can say  
EAR = WAR(Web module) + JAR(can be EJB module or application client module)  
  
Deployment Tools: Several methods are available to deploy the Oracle WebLogic Server applications and shared libraries, including:

1.   Administration console

2. WebLogic Scripting Tool(WLST)

3. WebLogic.Deployer java class

4. wldeploy Ant task

5.  Auto-deployment folder

1) Auto-deployment: Auto-deployment is a method for quickly deploying an application to a stand-alone server (Administration Server) for evaluation or testing. It is recommended that this method be used only in a single-server development environment.

            If auto-deployment is enabled, when an application is copied into the \auto-deploy directory of the Administration Server, the Administration Server detects the presence of the new application and deploys it automatically (if the Administration Server is running). If WebLogic Server is not running when you copy the application to the \auto-deploy directory, the application is deployed the next time the WebLogic Server Administration Server is started. Auto-deployment deploys only to the Administration Server.

            You can run a WebLogic Server domain in two different modes: development and production. Only development mode allows you use the auto-deployment feature

Development mode enables a WebLogic Server instance to automatically deploy and update applications that are in the domain\_name/auto-deploy directory (where domain\_name is the name of a WebLogic Server domain).

Production mode disables the auto-deployment feature and prevents any applications you place in the auto-deploy directory after you switch to production mode from being deployed. When you switch from development mode to production mode, any applications that were previously deployed via the auto-deploy directory remain deployed; if you wish to undeploy or redeploy such applications after you have switched to production mode, you must undeploy or redeploy them manually (for instance, with the WebLogic. Deployer command and the -undeploy or -redeploy options, as described in [WebLogic. Deployer Command-Line Reference](http://docs.oracle.com/cd/E11035_01/wls100/deployment/wldeployer.html)).

            To auto-deploy an archived application, copy its archive file to the /autodeploy directory. WebLogic Server automatically sets the application’s deployment mode to stage mode.

A deployment unit that was auto-deployed can be dynamically redeployed while the server is running. To dynamically redeploy, copy the new version of the archive file over the existing file in the /auto-deploy directory.

To undeploy an archived deployment unit that was auto-deployed, delete the application from the /autodeploy directory. WebLogic Server stops the application and removes it from the configuration.

2) Console Deployment: If we deploy an application in console deployment first we create domain, and start the Admin server.

Console deployment steps:

            Step1: Click on Deployments

                    2: Click on Lock And Edit

                    3: Click on Install

                    4: Select Location Deployed Application

                    5: Click on Next

                    6: Choose Targeting Style

                        Select  Install this deployment as an application

                    7: Click on Next

                    8: Select Deployment Targets

                        Admin server   or   Cluster

                    9: Click on Next

                    10: Select security rules and policies

                         Select DD only

                         Select Stage or No-Stage Mode

                     11: Click on Next

                     12: Click on Finish

                     13: Click on Activate Changes

You can test your application from within the Administrative Console by following the steps below:

1. In the Domain Structure section of the console, click 'Deployments'.

2. In the Summary of Deployments page, click on the name of the deployed Web application.

3. Select the 'Test' tab. Here, you'll find the URL to the deployed Web application. Click the link to launch it in a separate browser window.

Table of Deployable Modules:

|  |  |  |
| --- | --- | --- |
| Application or Module | Archive Extension | Key J2EE Deployment Descriptor |
| Enterprise Application | .ear | META-INF/application.xml |
| Enterprise JavaBean Module | .jar | META-INF/ejb-jar.xml |
| Web Application | .war | WEB-INF/web.xml |
| Web Service | .ear or .war | WEB\_INF/web-services.xml |
| Connector Module | .rar | META-INF/ra.xml |
| Startup or Shutdown Class | n/a | No deployment descriptor (Class file only) |

WebLogic server provides three different modes  for staging archive files.  
     1) Stage mode       2) No-stage mode         3) External stage mode

1) Stage mode: The administrator server copies the deployment unit files to the staging directories of target servers and they are deployed using local copy.

1.This mode is useful when deploying small or moderate size applications and prevents having a single point of failure if the original copy is not accessible.

2.  This is the default staging mode for managed server.

2) No-stage mode: The deployment units are deployed using the same physical copy, which must be accessible by the Administrator  server and target servers.

1. The administrator server does not copy the deployment unit files to the target server.

2. This mode is useful when deploying very large deployments to multiple targets and for deployment that require dynamic updates.

3. This is the default staging mode for the Administrator server.

3) External stage mode: In the External stage mode you must copy the deployment units manually to the correct staging directories before deployments.

1.Use this staging mode for deployments where you want to manually control the distribution of deployment files to target servers.

2.This mode prevents deployment information beginning dynamically updated . In this case the administration server  access the original deployment unit for validation.

3) Command Line Deployment:

i) Java WebLogic.Deployer: WebLogic.Deployer is a Java-based deployment tool that provides a command-line interface to the WebLogic Server deployment API.  WebLogic.Deployer is intended for administrators and developers who want to perform interactive, command-line based deployment operations.

To set up your environment to use the WebLogic.Deployer utility:

 Add the WebLogic Server classes to the CLASSPATH environment variable, and ensure that the   
correct JDK binaries are available in your PATH. You can use the setDomainEnv.cmd script,  
located in the server/bin subdirectory of the WebLogic Server installation directory, to set the environment.

Deploy:

Syntax: java WebLogic.Deployer [-adminurl] [specifiedurl(t3://localhost:7001)] [-username] [username]   
[-password] [password] [-name] [appname] [-source] [app source path] [-targets] [targets servers] -deploy

Ex:java WebLogic.Deployer -adminurl t3://localhost:7001 –username WebLogic -password  WebLogic  
 -name benefits -source C:\course\labs\Lab08\exercise\applications\benefits.war -targets  ms1, ms2  -deploy

Redeploy:

Syntax:  java WebLogic.Deployer [-adminurl] [specifiedurl(t3://localhost:7001)] [-username] [username]   
[-password] [password] [-name] [appname] [-targets] [targets servers] –redeploy

Ex:java WebLogic.Deployer -adminurl t3://localhost:7001 –username  WebLogic   -password  WebLogic   -name  benefits -targets  ms1, ms2  -redeploy

Undeploy:

Syntax:  java WebLogic.Deployer [-adminurl] [specifiedurl(t3://localhost:7001)] [-username] [username] [-password] [password] [-name] [appname] [-targets] [targets servers] –undeploy

Ex: java WebLogic.Deployer -adminurl t3://localhost:7001 –username WebLogic -password  WebLogic    -name  benefits -targets  ms1, ms2  -undeploy

To display list of applications:

Syntax: java WebLogic.Deployer [-adminurl] [specifiedurl(t3://localhost:7001)] [-username] [username]   
[-password] [password] -listapps

Ex: java WebLogic.Deployer -adminurl t3://localhost:7001 –username WebLogic  -password  WebLogic   -listapps

To display list tasks:

Syntax:java WebLogic.Deployer [-adminurl] [specifiedurl(t3://localhost:7001)] [-username] [username]  
 [-password] [password] [-listtasks]

Ex:java WebLogic.Deployer -adminurl t3://localhost:7001 –username  WebLogic -password  WebLogic   
  -listtasks

To check the server status:

Syntax: java WebLogic.Deployer [-adminurl] [specifiedurl(t3://localhost:7001)] [-username] [username]   
[-password] [password] [GETSTATE] [server name]

Ex: java WebLogic.Deployer -adminurl t3://localhost:7001 –username WebLogic -password  WebLogic   GETSTATE   ms1

WebLogic Scripting Tool(WLST): Wlst is a kind of scripting tool which is a combination of jython and python language.

The WebLogic Scripting Tool (WLST) is a command-line scripting environment that you can use to create, manage, and monitor WebLogic Server domains. It is based on the Java scripting interpreter, Jython. In addition to supporting standard Jython features such as local variables, conditional variables, and flow control statements, WLST provides a set of scripting functions (commands) that are specific to WebLogic Server.

You can use WLST as the command-line equivalent to the WebLogic Server. Administration Console (WLST online) or as the command-line equivalent to the Configuration Wizard (WLST offline).

Three types of modes : 1) Interactive mode. 2) Scripting 3) Embedded.

Interactive mode, in which you enter a command and view the response at a command-line prompt, is useful for learning the tool, prototyping command syntax, and verifying configuration options before building a script. Using WLST interactively is particularly useful for getting immediate feedback after making a critical configuration change. The WLST scripting shell maintains a persistent connection with an instance of WebLogic Server.

Script Mode: Scripts invoke a sequence of WLST commands without requiring your input, much like a shell script. Scripts contain WLST commands in a text file with a .py file extension, for example, filename.py. You use script files with the Jython commands for running script.

Embedded mode, you instantiate the WLST interpreter in your Java code and use it to run WLST commands and scripts. All WLST commands and variables that you use in interactive and script mode can be run in embedded mode.  We don’t use this mode generally.

To invoke the wlst go to C:/bea9/WebLogic91/common/bin/wlst.cmd  and execute wlst.cmd

You can use WLST as the command-line equivalent to the WebLogic Server Administration Console (WLST online) or as the command-line equivalent to the Configuration Wizard (WLST offline).

Offline: Without connecting to a running WebLogic Server instance, you can use WLST to create domain templates, create a new domain based on existing templates, or extend an existing, inactive domain. You cannot use WLST offline to view performance data about resources in a domain or modify security data (such as adding or removing users).

Online: You can use WLST to connect to a running Administration Server and manage the configuration of an active domain, view performance data about resources in the domain, or manage security data (such as adding or removing users). You can also use WLST to connect to Managed Servers, but you cannot modify configuration data from Managed Servers.

Display help information for WLST commands by entering the help command:

help(‘online’)

help(‘offline’)

wls:/offline> help()

WLST is a command line scripting tool to configure and administer WebLogic Server.

To invoke the wlst go to /bea/WebLogic91/common/bin /wlst.cmd  and execute wlst.cmd

Connect to WLST:

Step 1: Set class path first (C:\bea9\user\_projects\domains\ram\_domain\bin\SetDomainEnv.cmd)

Step 2:  enter WLST.cmd (C:\bea9\WebLogic91\common\bin\WLST.cmd)

            Installing WLST and goto offline mode.

Step 3: connect(‘username’,’password’,’url’)    enter

            Ex: connect(‘WebLogic’,’WebLogic’,’t3://localhost:9001’)

To connect to the domain specified port number. And goto online mode.

Wls:/ram\_domain/serverCofig>

Step 4: edit()

Step 5: startEdit()

            You goto edit mode and deploy an application after this.

Step 6: deploy an application

Syntax: deploy(‘appname’,’app path’,targets=’servers’)

Step 7: activate()

Step 8: disconnect()   disconnect and come to offline state.

Step 9: exit()  come out to the WLST.

Deploying a file using WLST in different ways: In this ways to deploy an application by using script based.

a) java WebLogic.WLST

Syntax: java WebLogic.WLST  path of script

Ex: java WebLogic.WLST  C:\scripts\deploy.py

Example script:

print ‘\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'

connect('WebLogic','WebLogic','t3://localhost:9001')

print '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'

edit()

print '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'

startEdit()

print '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'

print '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'

deploy('ShoppingCart','C:/course/labs/Lab25/exercise/applications/ShoppingCart.war',targets="ms1,ms2")

print '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'

save()

print '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'

activate()

print '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'

disconnect()

b)  WLST.cmd  script path

Syntax: WLST.cmd  script path

Ex: WLST.cmd   C:\scripts\deploy.py

3) WLST.cmd

Syntax: WLST.cmd

Wls:/offline> execfile(‘C:\scripts\deploy.py’)

iii) Side by Side Deployment:  Using side by side deployment strategy, the user can experience how to use the WebLogic server to re-deploy a new version of production application with out interrupting the availability of the application to new client request. The way the new client gets connected to the new version of the application and the previous version of the application is still in use by the older clients and gets retrived after the client disconnects.

Steps :

Step1. Copybenefits.war in new folder and deploy

Step2. setDomainEnv.cmd

Step3: java WebLogic.Deployer –adminurl  t3://localhost:7001 –username WebLogic  –password  WebLogic –name benefits  –source <app\_location>\benefits.war –nostage  –targets ms1,ms2 –deploy  
-appversion version1

Step 4.Copy benefits.war into another folder(new\_war) and extract

           jar –xvf benefits.war

Step5: Edit welcome file color:navy replace navy with Color:green

Step:6 Save

Step7: jar  –cvf benefits.war  \*(or) jar –cvf benefits.war “.”

Step8: Delete all files except  benefits.war

Step9: Now deploy benefits.war

: java WebLogic.Deployer –adminurl  t3://localhost:7001 –username WebLogic  –password  WebLogic –name benefits  –source <New\_app\_location>\benefits.war –nostage  –targets ms1,ms2 –deploy

-appversion version1

Step10: Test the application version1 before version2 deploy. Afetr version2 deploy test the application to see the difference. version1 application is in retried state.

IV) Deployment using plan:

 Steps:

|  |  |
| --- | --- |
| 1. | Start your administration server and managed servers, if not already started. If prompted, enter your domain's administrative username and password. |
| 2. | Download the [deploy\_plan.zip](http://www.oracle.com/webfolder/technetwork/tutorials/obe/fmw/wls/10g/r3/appdeploy/deploy/deploy_plan/files/deploy_plan.zip) file that contains the sample Web application and WLST script listed below:  HRApp.war deploy\_HRApp.py  Extract and place both files within the same directory on your local file system. This location will be referred to as <APP\_HOME> in later steps. |
| 3. | Open a new command shell, Navigate to the directory <INSTALL\_HOME>/wlserver\_10.3/server/bin where <INSTALL\_HOME> is the location of your Oracle WebLogic Server installation. |
|  | 4.Execute the setWLSEnv script. For example, on Linux, type the following:  source setWLSEnv.sh |
| 5. | Change directories to your <APP\_HOME> folder (the location of the downloaded WLST script and sample application). |
| 6. | Execute the deploy\_HRApp.py script using WLST:  java WebLogic.WLST deploy\_HRApp.py  Tip: If your domain's administrative credentials are not admin/welcome1, you will need to first edit this script file and change these values. Tip: Make sure you have not locked the administration console prior to running this script. |
| 7. | Confirm that the application has been deployed to the ms1 server. Direct a Web browser to the following URL:  <http://localhost:7003/HRApp> |

Generating a Deployment Plan for an Application:

Perform the following steps:

|  |  |
| --- | --- |
| 1. | Return to the same command shell used to run the WLST script. Confirm that the current directory is still <APP\_HOME>. |
| 2. | Execute the WebLogic.PlanGenerator tool on the HRApp.war application:  java WebLogic.PlanGenerator -all HRApp.war |
| 3. | You should receive a message similar to the following:  <Saved configuration for application, HRApp.war> |

Editing a Deployment Plan:

Perform the following steps:

|  |  |
| --- | --- |
| 1. | Locate the <APP\_HOME>/plan.xml file, and open it in a text editor. |
| 2. | Locate the following <variable> element:  <variable> <name>WebLogicWebApp\_ContextRoots\_*xxxxxxxxxxxxxx*</name> <value xsi:nil="true"></value>  </variable> |
| 3. | Remove the following text from the <value> child element:  xsi:nil="true" |
| 4. | Set the value of the <value> child element to /HR:  <value>/HR</value> |
| 5. | Futher down in the file, locate the following <variable-assignment> element:  <variable-assignment> <name>WebLogicWebApp\_ContextRoots\_*xxxxxxxxxxxxxx*</name> <xpath>/WebLogic-web-app/context-root</xpath> </variable-assignment> |
| 6. | Add a new <operation> child element to this <variable-assignment>:  <variable-assignment> <name>WebLogicWebApp\_ContextRoots\_*xxxxxxxxxxxxxx*</name> <xpath>/WebLogic-web-app/context-root</xpath> <operation>replace</operation> </variable-assignment> |
| 7. | Save your changes. |

Updating an Application with a Deployment Plan:

Perform the following steps:

|  |  |
| --- | --- |
| 1. | Launch a Web browser and access your domain's administration console. The default port is 7001:<http://localhost:7001/console> |
| 2. | Log into the console using your domain's administrative username and password. |
| 3. | In the Change Center panel, click Lock & Edit: |
| 4. | In the Domain Structure panel, click Deployments: |
| 5. | Select the checkbox for the HRApp application, and click the Update button: |
| 6. | Click the Change Path button associated with the Deployment Plan Path field : |
| 7. | Select the radio button for your new plan.xml file, and click Next. If necessary, use the hyperlinks next to the Current Location field to browse to your <APP\_HOME> directory: |
| 8. | Click the Finish button. |
| 9. | In the Change Center panel, click the Activate Changes button: |
| 10. | Verify the new context path of the application. Direct your Web browser to the following [URL:http://localhost:7003/HR](url:http://localhost:7003/HR) |

V) wl(WebLogic)deploy Ant task:

Step1: Create  .xml file

Step2: write  script in it.

Step3: execute command  ant  deploy {if we save the file name as build.xml}

Ant  –f  filename.xml {if the file name is created with filename.xml}

VI) Two-Phase Deployment: The new two-phase deployment protocol helps to maintain domain consistency. In previous versions of WebLogic Server, when you deployed an application, the administration server sent a copy of the application file(s) to all the targeted servers, which then loaded the application. If deployment to any of those servers failed or partially failed, the entire deployment's state across its target servers became inconsistent.

The two-phase model makes inconsistent deployment states in clusters less likely by confirming the success of the prepare phase before deploying the application on any targeted servers. A deployment that fails during the prepare phase will not enter the activation phase.

Prepare Phase: The prepare phase of deployment, the first phase, distributes or copies files and prepares the application and its components for activation, validating them and performing error checks on them. The purpose of the prepare phase is to ensure that the application and its components are in a state in which they can be reliably deployed.

Activate Phase: The second phase, the activate phase, is the actual deployment, or activation, of the application and its component with the relevant server subsystem. After the activate phase, the application is made available to clients.

JDBC( Java Data Base connectivity)

JDBC: Java database connectivity (JDBC) is the JavaSoft specification of a standard application programming interface (API) that allows Java programs to access database management systems. The JDBC API consists a set of interfaces and classes written in the Java programming language.  Using these standard interfaces and classes, programmers can write applications that connect to databases, send queries written in structured query language (SQL), and process the results.

The JDBC API is consistent with the style of the core Java interfaces and classes, such as java.lang and java.awt. The following table describes the interfaces, classes, and exceptions (classes thrown as exceptions) that make up the JDBC API. In the table, interfaces belonging to the javax.sql package are extensions to the standard JDBC interfaces and are contained in the Java 2 SDK, Enterprise Edition.

JDBC Architecture:

JDBC Drivers: There are four types of drivers in JDBC.

1) Type-1 Driver:(JDBC-ODBC  bridge driver)

1.   This driver receives any JDBC calls and sends then to ODBC driver.

2.   ODBC driver understand these calls and communicates with the database library provide by the vendor.

3.  ODBC driver and vendor database library must present on the client machine.

2) Type-2 Driver: (Native API-partly java driver)

1. It converts JDBC calls into database specific calls with the help of vendor database library.

2. It communicates directly with the database server; it requires that some binary code to be present on the client machine.

3) Type-3 Driver: (Net protocol-pure java driver)

1.  It follows three-tiered approach where by the JDBC database requests are passed through the network to a middle-tier server.

2. The middle-tier server translates the request to the database specific library and then sends it to the database server.

3.  The database server then executes the request and gives back to the result.

4)Type-4 Driver:( Net protocol-pure java driver )

4. It converts JDBC calls into the vendor-specific DBMS protocol. So that client applications can communicate directly database server.

5.   This driver completely implemented in java to archive platform independent.

In JDBC contains two types of Architectures.

1) Two-Tier Arcitecture:  In two-tier java application communicates directly with the DBMS.

A JDBC driver is needed that can communicate directly with the DBMS. So This is Client/Server configuration(Architecture)

Multi-tier Architecture: In multi-tier commands are sent to a middle-tier of services which then send the commands. The DBMS process commands sends the results back to middle-tier, which then sends them to client.

Data Source: A Data Source object provides a way for a JDBC client to obtain a database connection from a connection pool. It is used to store the WLST JNDI tree, it can support transactions and it is also associated with the connection pool.

What is JNDI(Java Naming Directory Interface):

 The Java Naming and Directory Interface (JNDI) is an application programming interface (API) for accessing different kinds of naming and directory services. JNDI is not specific to a particular naming or directory service, it can be used to access many different kinds of systems including file systems, distributed objects systems like CORBA, Java RMI, and EJB; and directory services like LDAP, Novell NetWare, and NIS+.

Why JNDI: In WebLogic Server, JNDI serves as a repository and lookup service for J2EE objects including:

1.   EJB home stubs

2.   JDBC DataSources

3.   JMS connection factories, queues and topics

4.   RMI stubs

JNDI Root(Tree):

Connection pool: A connection pool is a collection of database connections that is maintained by memory. That can be reused. Once an application has finished its physical connection the connection is recycled rather than beginning destroyed.

Connection pools:

1.   Are created at WebLogic server.

2.  Can be administrated using the Administration console.

3.  Can be dynamically resized to accommodate increasing load.

Benefits of DataSource And Connection pool:

1.   Time and overhead are saved by using an existing database connection

2.   Connection information is managed in one location in the Administration Console

3.   The number of connections to a database can be controlled

4.   The DBMS can be changed without the application developer having to modify underlying code

5.   A connection pool allows an application to “borrow” a DBMS connection.

JDBC DtataSource Architecture:

How to use DataSource:

A client retrieves a Data Source through JNDI look up and uses it to obtain a database connection.

Process :

1. Client will look up for the JNDI , once jndi is identified it will return database.

2. Data source will get connection from connection pool.

3. It will access the database with that connection.

JDBC  - To run point base Database:

Step1:Copy database files from c:\student\course\work\database\ and paste in c:\bea\WebLogic91\common\eva1\pointbase\database\

and edit startpointbase c:\bea\WebLogic91\common\eva1\pointbase\tools\startpointbase.cmd

Step2:  Edit  Startpointbase

Remove "%SAPLES\_HOME%\domain\wl\_server\pointbase.ini" and paste “c:\bea\WebLogic91\common\eva1\pointbase\tools”

and then save it.

 Step3: Run startpointbase script.

configuredatasoure in admin console and test connection:

Create a DataSource with the following specifications:

Name:                       dizzyworldDS

JNDI Name:             dizzyworldDS

Database Type:         PointBase

Database Driver:       \*PointBase’sDriver(Type 4)versions:4.x,5.x

Database Name:        HRDATABASE

Host Name:               localhost

Port:                          9092

Database User Name: PBPUBLIC

Password:                   PBPUBLIC

Initial Capacity:            5

Maximum Capacity:     15

Capacity Increment:     5

Login Delay:                1

Target:                        dizzy1

deploy an application testds.war from lab14and test for database connectivity.

Steps for creating data source and connection pool in console:

     Step1: click on services

            2: click on JDBC

             3: click on Data source

            4: click on Lock And Edit

            5: click on New

            6: Enter JDBC datasource properties

                        Name   : Datasource Name

                        JNDI Name    : JNDI Name

                        Database Type : database name

                        Driver Type     : Select Type-4 driver Non-Xa

                                    à Next

            7: Select Transaction options

                  Select

              Support global transaction

               Select one-phase commit  
                                   à Next

            8: Create a New DataSource

                        Database Name           : Database name

                        HostName                   : Hostname

                        PortNumber                : Portnumber

                        DB User Name           : username

                        DB password  : password

                        Conformpassword      : password

                                                àNext

9: Display connection properties

            10: Test configuration

            11: Display connection properties

                                    àConnection successes

àNext

            12: Select Targets (Admin or cluster(ms1&ms2))

                                    àFinish

                                    àActivate changes  
            13. Test Data Source

Cluster JDBC (or) MultiDataSource: A multi data source is an abstraction around a group of data sources that provides load balancing or fail-over processing between the data sources associated with the multi data source. Multi data sources are bound to the JNDI tree or local application context just like data sources are bound to the JNDI tree.

Multi Data Source Algorithms:

1)Failover: Connections requests are sent to the first data source in the list, if the request fails the request is sent to the next data source, in the list and so forth, the process is represented untial a valid connection is obtained or until the end of the list is reached in which case an exception is thrown.

2) Load balancing: The multi data source distributes connection requests evenly to its number data sources, which algorithm the multidata source also provides failover processing. That is if a request fails the multi data source sends to the requests to the next data source in the list until a valid connection is obtain, or until the end of the list is reached, in which case an exception is thrown.

Diff b/w Xa and Non-Xa  Datasource:

|  |  |
| --- | --- |
| Xa datasource  1)  It allows global transaction that my be        multiple resources.  2) It involves a co-ordinating transaction      manager  with one or more databases in       a single global transaction.  3) It comes  from the X/Open group       specification on distributed, global transactions. | Non-Xa Datasource  1)  It allows single transaction that my      be single resources.  2) there is no transaction coordinator, and it is     a single resource is doing all its transaction     work itself.  3) It comes from  Servlet or EJB or plain old  JDBC in a Java application talking to  a single database. |

JMS(JavaMessagingService)

JMS: The Java Message Service (JMS) [API](http://en.wikipedia.org/wiki/Application_Programming_Interface) is a [Java](http://en.wikipedia.org/wiki/Java_(programming_language)) [Message Oriented Middleware](http://en.wikipedia.org/wiki/Message_Oriented_Middleware) (MOM) API for sending messages between two or more [clients](http://en.wikipedia.org/wiki/Client_(computing)). JMS is a part of the [Java Platform Enterprise Edition](http://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition). It is a messaging standard that allows application components based on the Java 2 Platform Enterprise Edition (J2EE) to create, send, receive, and read messages. It allows the communication between different components of a [*distributed application*](http://en.wikipedia.org/wiki/Distributed_computing) to be [loosely coupled](http://en.wikipedia.org/wiki/Loose_coupling), reliable, and asynchronous.

The following are JMS elements:

JMS provider: An implementation of the JMS interface for a [Message Oriented Middleware](http://en.wikipedia.org/wiki/Message_Oriented_Middleware) (MOM). Providers are implemented as either a Java JMS implementation or an adapter to a non-Java MOM.

JMS client: An application or process that produces and/or receives messages.

JMS producer/publisher: A JMS client that creates and sends messages.

JMS consumer/subscriber: A JMS client that receives messages.

JMS message: An object that contains the data being transferred between JMS clients.

JMS queue: A staging area that contains messages that have been sent and are waiting to be read. Note that, contrary to what the name *queue* suggests, messages have to be delivered in the order sent A JMS queue only guarantees that each message is processed only once.

JMS topic: A distribution mechanism for publishing messages that are delivered to multiple subscribers

The JMS API supports two models:

1.   Point-to-point

2.  [Publish and subscribe](http://en.wikipedia.org/wiki/Publish/subscribe)

1) Point-to-Point: In the point-to-point model, a *sender* posts messages to a particular queue and a *receiver* reads messages from the queue. Here, the sender knows the destination of the message and posts the message directly to the receiver's queue. This model is characterized by the following:

1.  Only one consumer gets the message.

2. The producer does not have to be running at the time the consumer consumes the message, nor does the consumer need to be running at the time the message is sent.

3. Every message successfully processed is acknowledged by the consumer.

2) The publish/subscribe model supports publishing messages to a particular message topic. *Subscribers* may register interest in receiving messages on a particular message topic. In this model, neither the *publisher* nor the subscriber knows about each other. A good analogy for this is an anonymous bulletin board. The following are characteristics of this model:

1. Multiple consumers (or none) will receive the message.

2.  There is a timing dependency between publishers and subscribers. The publisher has to create a message topic for clients to subscribe. The subscriber has to remain continuously active to receive messages, unless it has established a durable subscription. In that case, messages published while the subscriber is not connected will be redistributed whenever it reconnects.Using Java, JMS provides a way of separating the application from the transport layer of providing data. The same Java [classes](http://en.wikipedia.org/wiki/Class_(computer_science)) can be used to communicate with different JMS providers by using the [JNDI](http://en.wikipedia.org/wiki/Java_Naming_and_Directory_Interface) information for the desired provider. The classes first use a *connection factory* to connect to the queue or topic, and then use populate and send or publish the messages. On the receiving side, the clients then receive or subscribe to the messages.

Difference b/w Queue and Topic:

|  |  |
| --- | --- |
| Queue | Topic |
| 1) In queues, one message can be consumed    by only one client.  2) Queue represent Point-To-Point model.  3) queue is used to send one to one system.  4) In queue the messages are send      to FIFO(First in first out) order. | 1) In the topics, one message can be consumed     by many clients.  2) Topic represent Public and Subscribe  model.  3) topic is used to send more than one system      at a time.  4) In Topic  the messages are send to LIFO     (Last in first out) order. |

Distributed Queue : Many producers can serialize messages to multiple receivers in a queue.

Distributed Topic : Publishing and subscribing to a topic decouples producers from consumers.

Difference b/w Distributed Queue and Distributed Topic:

|  |  |
| --- | --- |
| Distributed Queue  1) A Distributed Queues are allow you to     retrieve a connection to any of the Queues     across a cluster by using the Global JNDI name.  2) It seems one of the main pieces of     functionality Distributed Queue gives you is    load balanced connections across      multiple managed servers.  3) The members of the unit are usually     distributed across multiple servers within     a cluster, with each queue member belonging     to a separate JMS server.  4)A distributed queue is a set of physical     JMS queue members. | Distributed Topic  1) A distributed topic can be used to     create a TopicPublisher and TopicSubscriber.  2) The topic members can be located anywhere but  must all be served either by a single WebLogic Server  or any number of servers in a cluster.  3) A distributed topic is a set of physical JMS  topic members. |

JMS Architecture:

Connection Factory: A ConnectionFactory object encapsulates a set of connection configuration parameters that has been defined by an administrator. A client uses it to create a connection with a JMS provider.

1.  It encapsulates connection configuration information.

2.  It is used to create pre-configured connections.

3.  It stored in JNDI.

4.  Can be targeted server or cluster.

5.   It  supports concurrent use.

à The default connection factory that is bounded in JNDI to WebLogic is            WebLogic.jms.ConnectionFactory

Threshold and a Quota: A threshold and a quota can be set for Server and Destination objects.

 A quota is a limit defined for JMS administered objects; it includes these values:

1.  The maximum number of bytes that can be stored

2.  The maximum number of messages that can be stored

 A threshold is a limit that triggers message paging, flow control and logged warnings using:

1. Upper and lower values for the number of bytes

2. Upper and lower values for the number of messages

Durable Subscribers and Subscriptions:

1. Durable subscribers register durable subscriptions to guarantee message delivery even if subscribers are inactive.

2.  A subscriber is considered active if the Java object that represents it exists.

3.   By default, subscribers are non-durable.

Persistent store: The persistent store provides a built-in, high-performance storage solution for WebLogic Server subsystems and services that require persistence. For example, it can store persistent JMS messages or temporarily store messages sent using the Store-and-Forward feature. The persistent store supports persistence to a file-based store or to a JDBC-enabled database.

There are two types of  persistent mechanisms: 1) persistent         2)non-persistent

    A persistent message is guaranteed to be delivered once-and-only-once. The message cannot be lost due to a JMS provider failure and it must not be delivered twice. It is not considered sent until it has been safely written to a file or database. WebLogic JMS writes persistent messages to a WebLogic persistent store (disk-base file or JDBC-accessible database) that is optionally targeted by each JMS server during configuration.

   A Non-persistent messages are not stored. They are guaranteed to be delivered at-most-once, unless there is a JMS provider failure, in which case messages may be lost, and must not be delivered twice. If a connection is closed or recovered, all non-persistent messages that have not yet been acknowledged will be redelivered. Once a non-persistent message is acknowledged, it will not be redelivered.

When to Use Persistent Messaging:  Persistent messaging permits messages in memory to be written out to a persistent store.                                                                                                                               Configure persistent messaging if:                                                                                                                 – Development requires durable subscriptions (use durablesubscribers in the application)                      – You require that in-progress messages persist across server restarts

How a Durable Subscription Works:

When the client becomes active again, its ID is used to retrieve and redeliver messages.

Configure a Durable Subscription:

To configure durable subscriptions, an administrator must:

– Create and configure a JMS store

– Configure connection factories or destinations as persistent

– Associate the JMS store with the JMS Server

 The JMS store can be configured to use either:

   – A file store

   – A JDBC store (a connection pool)

Configure JMS server through console:

The following steps are:

            Step1:  Click on JMS Server

            Step2:  Click on Lock And Edit

            Step 3:  Click on New

            Step 4:  Enter JMS server Properties

                                    Name:  JMS server Name

                                    Persistent Store: none

                                  à Next

Step 5: Select targets

                        Target :  server name(ms1)

à Finish

Configure JMS Module:

The following steps are:

            Step1:  Click on JMS Modules

            Step2: Click on Lock And Edit

            Step 3:  Click on New

            Step 4:  Enter the following properties

                        Name :  JMS Module Name

                        Descriptor File Name:Filename

                        Location Domain:

                                    ----à Next

            Step5: Select Target :Cluster

                                    -----àNext

            Step 6: Select would you like to add Resources of this JMS System

                                    ------àFinish

Configure JMSQueue:

The following steps are:

            Step1: Click on JMS Module

            Step 2: Click on JMS Module Name

            Step 3: Click on Lock And Edit

            Step 4: Click on New

            Step 5: Select Queue

            Step 6: Description Properties

                                    Name: QueueName

                                    JNDI Name: JNDI Name

                                    Template: None

                                                --à Next

            Step 7: Select Targets  JMS Server

                                    ----à Finish

Configure JMSTopic:

The following steps are:

            Step1: Click on JMS Module

            Step 2: Click on JMS Module Name

            Step 3: Click on Lock And Edit

            Step 4: Click on New

            Step 5: Select Topic

            Step 6: Description Properties

                                    Name:  TopicName

                                    JNDI Name: JNDI Name

                                    Template:  None

                                                --à Next

            Step 7: Select Targets  JMS Server

                                    ----à Finish

Delete struck messages from the queue:

The following steps are:

            Step1: Click on JMS Module

            Step 2: Click on JMS Module Name

            Step 3: Click on JMS Queue

            Step 4: Click on Monitoring

            Step 5: select the queue

Step 6: Click on show messages

            Step 7: Select message ID

            Step 8: Delete

            Step 9: Finish

Check the pending messages in a queue:

The following steps are:

            Step1: Click on JMS Server

            Step 2: Click on JMS Server Name

            Step 3: Click on monitoring

            Step 4: Click on Active destrination

                        -à To check the pending messages in queue.

SSL(Security Socket Layer):

I) Configure SSL in WebLogic:  
1. Generating the certificate:  
The following steps are:  
Step1: Open a command prompt and set the environment by running the setDomainEnv script.  
           ( C:\bea9\user\_projects\domains\ram\_domain\bin\setDomainEnv.cmd)  
Step2: Generate the private – public key pair. For demonstration we would use keytool java utility to do so.            However we can use other utilities like openssl etc.  
           keytool  -genkey  -alias mykey   -keyalg RSA   -keysize 2048   -keystore  identity.jks  
Step3: Generate a Certificate Signing Request (CSR) and send it to Certifying Authority.  
            keytool  -selfcert   -alias mykey  -keystore   identity.jks  
Step 4: Create a identity  keystore, this can be done my exporting  
            keytool   -export  -alias  mykey  -file  cert.cer  -keystore   identity.jks  
Step5: Create a trust keystore, this can be done my importing.  
            keytool   -import   -alias   mykey  -file  cert.cer   -keystore   trust.jks    -noprompt  
To verify the contents of the keystore, you can use the below command,  
            keytool  -list  -v  -keystore <keystore-name>  -storepass <keystore-password>

2)  Configuring the keystore on the WebLogic Server:  
Step 1: Log into the Admin Console, Click on servers  
Step 2: Click on Lock and Edit  
Step 3: select the server on which you want to configure the SSL    certificate.(Ex:ms1)  
Step 4: Click on keystores  
Step 5: select Custom identity and Custom trust  
Identiy:  
CustomIdentitykeystore:C:\bea9\user\_projects\domains\sai\_domain\identity.jks  
Custom Identity keystore type: jks  
Custom identity passphrase : Pavan@123  
Trust:  
Custom trust keystore: C:\bea9\user\_projects\domains\sai\_domain\trust.jks  
Custom trust keystore type: jks  
Custom trust passphrase : Pavan@123  
àsave     ---àAcivate changes

Step 6: Click on SSL  
Step 7: Enter identity  
            Private key alias: mykey  
            Privatekey passphrase : Pavan@123  
                        ---àsave     ---à Activate changes

àTo check SSL type browser  [https://localhost:5003(ms1](https://www.blogger.com/null) ssl portnumber)/messaging(deploying application)

                                    Apache Webserver

Install the apache web server in Linux:

Step 1: first unzip the file on zip file  
            Gunzip    httpd-2.0.55.gz  
Step 2: tar file is open. Un tar that file  
            Tar –xvf  httpd-2.o.55.tar

            The file will display   httpd-2.o.55

Step 3: cd  httpd-2.0.55

            ./configure—prefix= /home/apache2

            ./make

            ./make install

The install is completed.

Check Apache servers running processes:  
  
ps  -ef | grephttpd | grep  -v grep

1) To start apache - httpd –k start

2) To stop apache – httpd –k stop

3) To Restart apache – httpd –k restart

Program files / Apache software foundation / Apache 2.2/bin folder in

i) Main configuration file in Apache is  “httpd.conf”

      ii) Additional config files –“Extra” folder

4) Apache default port no – 80

5) Apache ssl default port number – 443

6) To check the syntax – httpd  -t

7) All html documents – “htdocs” folder

[The Apache Directory Structure](https://www.blogger.com/null): The Apache software is typically distributed into the following subdirectories:

|  |  |
| --- | --- |
| cgi-bin | This is where many, if not all, of the interactive programs that you write will reside. These will be programs written with Perl, Java, or other programming languages. |
| Conf | This directory will contain your configuration files. |
| htdocs | This directory will contain your actual hypertext documents. This directory will typically have many subdirectories. This directory is known as the DocumentRoot. |
| Icons | This directory contains the icons (small images) that Apache will use when displaying information or error messages. |
| images | This directory will contain the image files (GIF or JPG) that you will use on your web site. |
| Logs | This directory will contain your log files - the access\_log and error\_log files. |
| Sbin | Use nogroup |

Main Configuration file in apache:

1.                  The Apache software is configured by changing settings in several text files in the Apache conf (configuration) directory.

2.                  There are four configuration files used by Apache. The main configuration file is usually called httpd.conf.

|  |  |
| --- | --- |
| access.conf | This is The security configuration file. It Contains instructions about which users should be able to access.  And  what information. |
| httpd.conf | This is The server configuration file. It Typically contains directives that affect how the server runs, such as user and group ID's it should use when running, the location of other files, etc. |
| srm.conf | This is The resource configuration file. It Contains directives that define where documents are found, how to change addresses to filenames, etc. |
| mime.types | A configuration file that relates filename extensions to file types. |

Apache root: On an Apache HTTP Server, you can control where the server looks for live HTML documents; in other words, your web pages or content management system.

DocumentRoot: . The directory will typically have many subdirectories. This directory is known as the DocumentRoot. The documentRoot directory in apache is htdocs.

Server root: Server Root Directory                                                                                     Syntax: ServerRoot A<path>< tt="">                                                                                Example: ServerRoot /sw/pkg/apache     
               Since: Apache 1.0  </path><>

This directive sets the root directory in which the server resides. Typically, it contains the subdirectories conf/ and logs/. Relative paths for other configuration files are taken as relative to this directory. This directive can be also overridden from the command line via the -d option.

II) Integrate Apache with WebLogic Server:  Install apache in our machine and open httpd.conf file (C:\Program Files (x86)\Apache Software Foundation\Apache2.2\conf\httpd.conf) and to do the following steps are:  
Step 1: Make sure the apache server run on port number 80 or 8080  
            Listen localhost:80 or 8080                                                                                                                          Listen:80  
Step2:Copy the module Mod\_WL\_20 to C:\ProgramFiles(x86)\ApacheSoftware Foundation\Apache2.2\modules  
Step 3: copy the module on  LoadModule WebLogic\_module modules/mod\_wl\_22.so  to httpd.conf file.  
Step 4: copy the below content in to  httpd.conf  on main server.  
<Location />     
SetHandler WebLogic-handler                                                                                                 </Location>

<IfModule mod\_WebLogic.c>    
WebLogicCluster localhost:5002,localhost:5004                                                                                   Debug ON                                                                                                                                          WLLogFile c:/temp/wlproxy.log                                                                                                      WLTempDir c:/temp     
 </IfModule>  
 Step 4: Re-start the server and access the application.

[http://localhost:apacheportnumber/appname(http://localhost80:/messaging)](https://www.blogger.com/null)

 III) Integrate Apache-SSL with WebLogic server:  Install the "httpd/apache\_x.x.x-win32-x86-openssl-x.x.x.msi" s/w in our machine. Open httpd.conf file Then do the following steps

Step 1: Configure apache configure file httpd.conf, uncomment the following 2 lines,  
            LoadModule ssl\_module modules/mod\_ssl.so  
            Include conf/extra/httpd-ssl.conf

Step2: Execute the following step for windows

set OPENSSL\_CONF=C:\Program Files\Apache Software Foundation\Apache2.2\conf\openssl.cnf

Step 3: Generate certification using openssl,

openssl req -new -out server.csr  
openssl rsa -in privkey.pem -out server.key  
openssl x509 -in server.csr -out server.cert -req -signkey server.key -days 365

Step 4:  Copy the generation files to the directory defined by httpd-ssl.conf

            We have the Self-signed SSL certificates ready now. Now We need to MOVE the "server.cert" and        "server.key" file to the

     "C:\Program Files\Apache Software Foundation\Apache2.2\conf" location.

Step 5:  check httpd-ssl.conf

    Now we need to modify the "C:\Program Files\Apache Software Foundation\Apache2.2\conf\extra\httpd- ssl.conf".

 Let all the default options as it is but make sure to modify the following section according to your need:

<VirtualHost \_default\_:443>  
ServerAdmin some@email.com  
DocumentRoot "Your Root folder location"  
ServerName www.domain.com:443  
ServerAlias domain.com:443  
ErrorLog "logs/anyFile-error.log"  
CustomLog "logs/anyFile-access.log" common  
SSLEngine on

SSLCertificateFile "C:/Program Files/Apache Software Foundation/Apache2.2/conf/server.cert"

SSLCertificateKeyFile "C:/Program Files/Apache Software Foundation/Apache2.2/conf/server.key"  
</VirtualHost>

Step 6: Open an exception in Windows Firewall for TCP port 443.(set ssl port number:443 in our mechine)

 Step 7: Access the application using the below url

[https://localhost:443/](https://localhost/app-name)app-name or [https://localhost](https://localhost/)/app-name

Set  SSLport number:443 in our mechine:

Step 1: Click on start button

Step2 : Click on Control panel

Step 3: Click on windows firewall

Step 4: Click on Advanced Settings

Step 5: Click on Inbounded rules

Step 6: Click on new rule

Step 7: Select port

                        ----à click on next

Step 8:  select TCP

                Select specific location port: 443

                        ------à next

Step 9: Select Allow the connections

                        ----à next     ---ànext

Step 10:  SSL port is created

Diff  b/w one-way ssl and two-way ssl:

One Way SSL :- Only the client authenticates the server. This means that the public cert of the server needs to configured in the trust store of the client.  
Two Way SSL: - The client authenticates the server & the server also authenticates the client.  
This means that the public cert of the server needs to configured in the trust store of the client. And Also the public cert of the client needs to be configured on the server's trust store.

Virtual Hosting:

Diff b/w Name-based and Ip-based virtual hosting:

IP-based virtual hosts use the IP address of the connection to determine the correct virtual host to serve.  
Name-based virtual hosting, the server relies on the client to report the hostname as part of the HTTP headers & many different hosts can share the same IP address.

Configure virtual host in apache and WebLogic:

 Go to servers  -> click on admin server -> click protocol -> select the chanel tab ->click new option N/w channel name : 80protocol:http  and click next ->  listen Address: localhost listen port : 80 and click next -> click finish  
  Go to environment -> virtual hosts ->click new option -> create a new virtual host -> click ok.

JVM(Java Virtual Machine)

JVM Architecture: JVM is the heart of any Java based Application Server. We face most of the issues due to incorrect JVM tuning. It is very important to understand the Overall architecture of the JVM in order to trouble shoot different JVM tuning related issues. Here we are going to discuss the Architecture and the Major parts of a Java Process And the Java Heap Division.

The Following Diagram is just a basic overview of a Java Process in a 2 GB process Size Machine. Usually in 32 bit Windows Operating Systems the default process size will be 2 GB (In Unix based 64 bit operating Systems it can be 4GB or more). So i draw the following Diagram of Java Process to explain the Java Process partitions in a 2Gb process size machine.

Java Process Architecture Diagram

In the above diagram we will find different partitions of a Java Process. Please compare the above diagram with below descriptions.

.

1) Just for Example we can see that Process Size is 2048 MB (2GB)

2) The Java Heap Size is 1024MB (means 1GB)   -Xmx1024m

3) Native Space = ( ProcessSize – MaxHeapSize – MaxPermSize) It means around 768 MB of Native Space.

4) MaxPermSpace is around -XX:MaxPermSize=256m

5) Young Generation Space is around    40% of Maximum Java Heap.

Different parts of JVM:

1) Eden space: Eden Space is a Part of Java Heap where the JVM initially creates any objects, where most objects die and quickly are cleaned up by the minor Garbage Collectors (Note: Full Garbage Collection is different from Minor Garbage Collection). Usually any new objects created inside a Java Method go into Eden space and the objects space is reclaimed once the method execution completes. Where as the Instance Variables of a Class usually lives longer until the Object based on that class gets destroyed. When Eden fills up it causes a minor collection, in which some surviving objects are moved to an older generation.

2) Survivor Spaces: Eden Space has two Survivor spaces. One survivor space is empty at any given time. These Survivor Spaces serves as the destination of the next copying collection of any living objects in Eden and the other survivor space.

The parameter SurvivorRatio can be used to tune the size of the survivor spaces.

-XX:SurvivorRatio=6 sets the ratio between each survivor space and Eden to be 1:6

If survivor spaces are too small copying collection overflows directly into the tenured generation.

3) Young Generation: (-XX:MaxNewSize): Till JDK1.3 and 1.4 we used to set the Young Generation Size using -XX:MaxNewSize. But from JDK1.4 onwards we set the YoungGeneration size using (-Xmn) JVM option.

Young Generation size is controlled by NewRatio.  It means setting -XX:NewRatio=3 means that the ratio between the Old Generation and the Young Generation is  1:3. Similarly -XX:NewRatio=8 means that 8:1 ratio of tenured and young generation.

NewRatio: NewRatio is actually the ratio between the (YoungGenaration/Old Generations) has default values of 2 on Sparc , 12 on client Intel, and 8 everywhere else.

NOTE: After JDK 1.4 The Young Generation Size can be set using  (-Xmn) as well.

1) Virtual space-1:(MaxNewSize – NewSize): The First Virtual Space is actually shows the difference between the -XX:NewSize and -XX:MaxNewSize.  Or we can say that it is basically a difference between the Initial Young Size and the Maximum Young Size.

JavaHeapArea:( -Xmx and –Xms): Java Heap is a Memory area inside the Java Process which holds the java objects.  Java Heap is a combination of Young Generation Heap and Old Generation Heap. We can set the Initial Java Heap Size using -Xms JVM parameter similarly if we want to set the Maximum Heap Size then we can use -Xmx JVM parameter to define it.

Example:

-Xmx1024m —> Means Setting the Maximum limit of Heap as 1 GB

-Xms512m —> Means setting Java Heap Initial Size as 512m

.

NOTE-1): It is always recommended to set the Initial and the Maximum Heap size values as same for better performance.

NOTE-2): The Theoretical limitation of Maximum Heap size for a 32 bit JVM is upto 4GB. Because of the Memory Fragmentation, Kernel Space Addressing, Swap memory usages and the Virtual Machine Overheads are some factors JVM does not allow us to allocate whole 4GB memory for Heap in a 32 bit JVM. So usually on 32-bit Windows Operating Systems the Maximum can be from 1.4 GB to 1.6 GB.

If we want a larger memory allocation according to our application requirement then we must choose the 64-bit operating systems with 64 bit JVM. 64-bit JVM provides us a larger address space. So we can have much larger Java Heap  with  the increased number of Threads allocation area. Based on the Nature of your Operating system in a 64 bit JVM you can even set the Maximum Heap size upto 32GB.

Example:        *-Xms32g -Xmx32g -Xmn4g*

2) Virtual Space-2: (MaxHeapSize – InitialHeapSize): The Second Virtual Space is actually the Difference between the Maximum Heap size (-Xmx)and the Initial Heap Size(-Xms). This is called as virtual space because initially the JVM will allocate the Initial Heap Size and then according to the requirement the Heap size can grow till the MaxHeapSize.

PermGen Space: (-XX:MaxPermSize): PermGen is a non-heap memory area where the Class Loading happens and the JVM allocates spaces for classes, class meta data,  java methods and the reference Objects here. The PermGen is independent from the Heap Area. It can be resized according to the requirement using -XX:MaxPermSize and -XX:PermSize  JVM Options. The Garbage collection happens in this area of JVM Memory as well. The Garbage collection in this area is called as “Class GC”. We can disable the Class Garbage Collection using the JVM Option -noclassgc. if  ”-noclassgc” Java Option is added while starting the Server. In that case the Classes instances which are not required will not be Garbage collected.

Native Area: Native Memory is an area which is usually used by the JVM for it’s internal operations and to execute the JNI codes. The JVM Uses Native Memory for Code Optimization and for loading the classes and libraries along with the intermediate code generation.

The Size of the Native Memory depends on the Architecture of the Operating System and the amount of memory which is already commited to the Java Heap. Native memory is an Process Area where the JNI codes gets loaded or JVM Libraries gets loaded or the native Performance packs and the Proxy Modules gets loaded.

There is no JVM Option available to size the Native Area. but we can calculate it approximately using the following formula:

NativeMemory = (ProcessSize – MaxHeapSize – MaxPermSize)

Garbage collection: It’s always best to enable the Garbage collection Logging in our production environment as well because it does not cause any resource overhead or any side effect on WebLogic server or another application server’s performance.  GC log helps us in investigating man issues. Apart from issues it helps us to find out if some tuning is required based on the statistics of the Garbage collection.                          Garbage collection logging can be enable and collected in a separate log file by using the following JAVA\_OPTIONS:

-Xloggc:D:/gcLogs/GCLogs.log         -XX:+PrintGCDetails        -XX:+PrintGCTimeStamps

As soon as you add these JAVA\_OPTIONS which are JVM specific (above will work for Sun and Open JDKs fine) the JVM will start generating the garbage collection logging in the GCLog.log file. Now if you will open this file then you can

see something like following:

|  |  |
| --- | --- |
| *01* | *4.636: [GC [PSYoungGen: 230400K->19135K(268800K)] 230400K->19135K(2058752K), 0.0635710 secs] [Times: user=0.08 sys=0.01, real=0.06 secs]* |
| *02* | *7.302: [GC [PSYoungGen: 249535K->38396K(268800K)] 249535K->51158K(2058752K), 0.0777300 secs] [Times: user=0.21 sys=0.04, real=0.07 secs]* |

|  |  |
| --- | --- |
| *03* | *7.521: [GC [PSYoungGen: 49735K->38388K(268800K)] 62496K->51933K(2058752K), 0.0741680 secs] [Times: user=0.15 sys=0.04, real=0.07 secs]* |
| *04* | *7.595: [Full GC (System) [PSYoungGen: 38388K->0K(268800K)] [PSOldGen: 13545K->51794K(1789952K)] 51933K->51794K(2058752K) [PSPermGen: 19868K->19868K(39936K)], 0.3066610 secs] [Times: user=0.28 sys=0.02, real=0.31 secs]* |

|  |  |
| --- | --- |
| *05* | *9.752: [GC [PSYoungGen: 230400K->26206K(268800K)] 282194K->78000K(2058752K), 0.0728380 secs] [Times: user=0.15 sys=0.00, real=0.08 secs]* |
| *06* | *11.906: [GC [PSYoungGen: 256606K->38393K(268800K)] 308400K->94759K(2058752K), 0.1058920 secs] [Times: user=0.19 sys=0.00, real=0.10 secs]* |

|  |  |
| --- | --- |
| *07* | *13.480: [GC [PSYoungGen: 268793K->38394K(268800K)] 325159K->109054K(2058752K), 0.0762360 secs] [Times: user=0.20 sys=0.03, real=0.08 secs]* |
| *08* | *18.115: [GC [PSYoungGen: 268794K->38384K(268800K)] 339454K->179238K(2058752K), 0.1351350 secs] [Times: user=0.42 sys=0.10, real=0.14 secs]* |

|  |  |
| --- | --- |
| *09* | *20.860: [GC [PSYoungGen: 268784K->38394K(268800K)] 409638K->200343K(2058752K), 0.1063430 secs] [Times: user=0.29 sys=0.03, real=0.11 secs]* |
| *10* | *22.148: [GC [PSYoungGen: 268794K->38399K(268800K)] 430743K->221395K(2058752K), 0.1173980 secs] [Times: user=0.24 sys=0.02, real=0.12 secs]* |

|  |  |
| --- | --- |
| *11* | *23.357: [GC [PSYoungGen: 268799K->26775K(268800K)] 451795K->231618K(2058752K), 0.0714130 secs] [Times: user=0.15 sys=0.03, real=0.08 secs]* |
| *12* | *24.449: [GC [PSYoungGen: 257175K->29170K(268800K)] 462018K->239909K(2058752K), 0.0312400 secs] [Times: user=0.06 sys=0.01, real=0.04 secs]* |

You can notice something in the above output:

Point1: [Full GC (System) [PSYoungGen: 38388K->0K(268800K)]    It means a Full GC is happening on the complete Heap Area including all the Areas of the Java Heap Space.

Point2: [GC [PSYoungGen: 230400K->19135K(268800K)]   Indicates some small GCs which keep on happening in the young generation very frequently, This garbage collection cleans the Young Generation short living Objects.

Point3:  Meaning of the [GC [PSYoungGen: 230400K->19135K(268800K)]   line is around 256MB (268800K) is the Young Generation Size, Before Garbage Collection in young generation the heap utilization in Young Generation area was around  255MB (230400K)  and after garbage collection it reduced up to 18MB (19135K)

Point4:  Same thing we can see for Full Garbage collection as well….How effective the Garbage collection was…[Full GC (System) [PSYoungGen: 38388K->0K(268800K)] [PSOldGen: 13545K->51794K(1789952K)]  Here it says that around

[(old)1789952K +  young (268800K) ]  memory space means  OldGeneration is consuming 1.75GB space and Young Generation is consuming around 255 MB space  So it means total Heap size is around 2GB.

But analyzing the Garbage collection log like above technique Line by Line is very bad…so here we have an alternative was to analyze the Garbage Collection log in few Seconds to see how much time the Full Garbage collection is taking as an average and other reports…etc.

Step1): Download the “garbagecat-1.0.0.jar   (881 KB) ”  tool from the follwing link:<http://garbagecat.eclipselabs.org.codespot.com/files/garbagecat-1.0.0.jar>

Step2): Open a command prompt and then make sure that JAVA is set in the Path so that we can use “jar” utility of JDK to run the “garbagecat-1.0.0.jar”  tool.

Step3): Put the “garbagecat-1.0.0.jar”  file and the “GCLog.log” file in the same directory. then run the following command:

java      -jar      garbagecat-1.0.0.jar      GCLog.log

Step4): As soon as our run the above command you will see that in your current directory following files are created:

garbagecat-1.0.0.jar

GCLog.log

gcdb.lck

gcdb.log

gcdb.properties

report.txt

Step5): Now open the “report.txt” file to see the Overall report of the Garbage Collection something like following:

========================================

SUMMARY:

========================================

# GC Events: 12

GC Event Types: PARALLEL\_SCAVENGE, PARALLEL\_SERIAL\_OLD

Max Heap Space: 2058752K

Max Heap Occupancy: 462018K

Max Perm Space: 39936K

Max Perm Occupancy: 19868K

Throughput: 95%

Max Pause: 306 ms

Total Pause: 1233 ms

First Timestamp: 4636 ms

Last Timestamp: 24449 ms

========================================

.

If you see that the Garbage Collection Max Pause time is very high like more than 5-7 Seconds for a 2 GB heap then you need to worry about it.

NOTE: Garbage collection  is a best utility to generate the Garbage Collection Report for Sun JDK and Open JDK for other JDKs you should use other tools for accurate results.

Garbage collection algorithms:

1) Mark-and-sweep collector:  This type of collector first traverses the object graph and marks reachable objects. It then scans the heap for unmarked objects and adds their memory to a list of available memory segments. This collector typically uses a single thread to do its work and is a stop-the-world collector.

2) Mark-and-compact collector:  A mark-and-compact collector, sometimes known as a mark-sweep-compact collector, uses the same marking phase as a mark-and-sweep collector. During the second phase, it compacts the heap by copying marked objects to a new area of the heap. These collectors are also stop-the-world collectors.

3) Copying collector:  This type of collector divides the heap into two areas, commonly known as semi-spaces. It uses only one semi-space at a time; the JVM creates all new objects in one semi-space. When the garbage collector runs, it copies any reachable objects it finds to the other semi-space as it finds them, thus compacting the heap as it copies live objects. All dead objects are left behind. This algorithm works well for short-lived objects, but the expense of continually copying long-lived objects makes it less efficient. Again, this is a stop-the-world collector.

4) Incremental collector:  Incremental collectors basically divide the heap into multiple areas and collect garbage from only one area at a time. This can create much smaller, though more frequent, pauses in your application. There are numerous approaches defining how the actual collection is handled from traditional mark-and-sweep to algorithms designed explicitly for use with multiple smaller areas like the train algorithm. See “Incremental Mature Garbage Collection Using the Train Algorithm” by Jacob Seligmann and Steffen Grarup (<http://www.daimi.aau.dk/~beta/Papers/Train/train.html>) for more information.

5) Generational collector:  This type of collector divides the heap into two or more areas that it uses to store objects with different lifetimes. The JVM generally creates all new objects in one of these areas. Over time, the objects that continue to exist get tenure and move into another area for longer-lived objects. Generational collectors often use different algorithms for the different areas to optimize performance.

6) Concurrent collectors:  Concurrent collectors run concurrently with the application, typically as one or more background threads. These collectors typically have to stop-the-world at some point to complete certain tasks, but the amount of time they halt all processing is significantly reduced because of their other background work.

7) Parallel collectors:  Parallel collectors typically use one of the traditional algorithms but use multiple threads to parallelize their work on multiprocessor machines. Using multiple threads on multi-CPU machines can dramatically improve the scalability of a Java application on multiprocessor machines.

Creating And Analyze log for Garbage Collection:

Creating GC log: Set memory argument in startWebLogic.cmd.

-Xms256m –Xmx512m –XX:CompailThreshold=8000 –XX:permSize=48m –XX:MaxPermSize=128m

analyze GC logs:

Srep1:Go to admin console

Step2: Click the Adminserver

Step3: Select the monitors tab

Step4: Click the performance

Step5: Select the garbage collector

Security Realms: A security realm is a container for the mechanisms including users, groups, security roals, security palaces and providers. That are used to protect WebLogic resources, we can have multiple security reasons in a WebLogic servers domain. But only one can be set as the default realm.

            This security realms page, lists is security realms that has been configured in this WebLogic server domain. Click the name of the realms to explore and configure that realm.

Creating users and groups in WLS:

            Step1:  Goto security realm

            Step2: Click on Lock And Edit

            Step3: Select Myrealm

            Step4: Select user and group tab

            Step5: Click on New

                        Name               :  Ram

                        Designation     :  WebLogic\_Admin

                        Provider          :  Select provider

                        Password         :  Ram123

                        ConformPwd  :  Ram123

                                    ------->    Ok

            Step6: Click on Ram

            Step7: Click on group Tab

            Step8: Select Role

            Step9: Click on Ok

Step10: Click on Save

Roals:

1) Admin Channel Users: Admin channel users can access the admin channel.

2) Administrator: Administrator can view and modify all the resource attributes and start, stop servers.

3) App Testing: App testing group.

4) Cross Domain Connecters: Cross Domain Connecters can make inter-domain  calls from foreign domains.

5) Deployers: Deployer can view all the resource attributes and deploy applications.

6) Monitors: Monitors can view and modify all resource attributes and perform operations not restricted by roles.

7) Operators: Operators can view and modify all response attributes and perform server life cycle operations.

Difference between unicast and multicast:

|  |  |
| --- | --- |
| Unicast  1)  It is just one-to-one communication that      takes place between the client and the server.  2)  In unicast, one packet is transmitted to only     one destination at a time. So it recive only one           reciver.  3) In unicast, the Internet protocol methods,     such as, TCP(Transmission Control     Protocol) and UDP(User Datagram Protocol)     are used.  4) When a user uses the Windows Media    Player, he or she has direct contact with   the server. Each of the users using   the unicast [system](http://www.differencebetween.net/miscellaneous/difference-between-caste-system-and-class-system/) utilises additional bandwidth. | Multicast  1) It is a multi-communication level and mainly      multicasts enabled routers used for broadcasting.  2) In multicast sends packets to multiple      destinations which is represented by a group address.       So it receives multiple receivers. 3) When the former is more practical as only       a small section of the Internet is multicast enabled.  4)  In multicast, there is no direct link between      the user and the server. When using Windows     Media Player, the user does not have any direct      link with the server. Instead, once the user starts to     use Windows Media Player, an .nsc or Next      how channel is generated which is then delivered      to the user from the server. |

Trouble shooting issues:

Deployment:   We will send error log to the application team for modification.

Caused By: WebLogic.utils.ErrorCollectionException:

There are 1 nested errors:

WebLogic.j2ee.dd.xml.AnnotationProcessException: Duplicate ejb name 'BDAccountEjbBean' found: annotation 'Stateless' on bean failed due to connection pool issue: we will fix connection pool issues and then redeploy the application

Out of memory issue during the deployment:

error: java.lang.outofmemory.permgenspace

this error occured due to space in perm area.

setDomainEnv.sh

xx:permsize 128m

xx:maxpermsize 128m

we have set intialpermsize=maxpermsize then restarted the servers, redeployed the application

If one or two application failed when we are triggering through scipt.we will fix that issue and do a deployment using console

JDBC:

1) DB down (raise a ticket to db team)

2) In correct hostname or port number ( raise a ticket to network team)

3) Data base connection lost ( telnetipaddress port )

4) Data base user\_acc lock ( raise a ticket to db team for unlocking user\_acc)

5) Invalid pakage error (raise a ticket to db team)

6) TNS listener error (raise a ticket to db team)

7) Schema does not exist (raise a ticket to db team)

8) Cannot allocate resource error

Intialcapacity : 5

max      : 15

increase max to 25

9) Connection leaks ( send error to application team)

10) Connection time out ( raise a tickect to db team for long running quries)

JMS:

stuck message issues

Check whether dest queue is available, check message format, check queue name.

rolling message issues (messages will run continuously in the loop)

delete those messages in the queue.

Disk Space:

If the disk space usage is 95%-100% then we will delete old log files

[root@localhost ~]# df -kh

FilesystemSize  Used Avail Use% Mounted on

/dev/sda2             3.8G  1.9G  1.8G  52% /

/dev/sda1              46M  9.2M   35M  22% /boot

tmpfs                 506M     0  506M   0% /dev/shm

/dev/sda3              14G  1.8G   12G  100% /home

du -kh (disk usage)

s

[root@localhost ~]# du -sh /home

1.8G    /home

[root@localhost bea10.3]# du -sh \*

181M    jdk160\_05

28K     logs

211M    jrockit\_160\_05

100M    modules

24K     registry.dat

8.0K    registry.xml

19M     user\_projects

556K    utils

429M    wlserver\_10.3

delete old log files

/home/bea10.3/user\_projects/domains/sherkhan/servers/AdminServer/logs

rm -rf Adminserver.log00001  Adminserver.log00002 Adminserver.log00003

rm -rf Adminserver.out00001 Adminserver.out00002 Adminserver.out00003

rm -rf access.log00001 access.log00002 access.log00003

/home/bea10.3/user\_projects/domains/sherkhan/servers/ms1/logs

rm -rf ms1.log00001

rm -rf ms1.out00001

or zip the log files

/home/bea10.3/user\_projects/domains/sherkhan/servers/AdminServer/logs

gzip -r \*

/home/bea10.3/user\_projects/domains/sherkhan/servers/AdminServer

gzip -r logs

High CPU utilization:

top (linux)

prstat (solaris)

top - 07:45:22 up  3:03,  3 users,  load average: 0.16, 0.33, 0.17

Tasks: 113 total,   2 running, 109 sleeping,   0 stopped,   2 zombie

Cpu(s):  0.0%us,  0.7%sy,  0.0%ni, 99.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st

Mem:   1035400k total,  1020348k used,    15052k free,    77688k buffers

Swap:  2040212k total,        0k used,  2040212k free,   483724k cached

                                        %cpu %Mem

9523 root      22   0  637m 239m 3660 S 98.7 23.7   0:12.79 java

If you find any zombie process count >50 raise a ticket to solaris admins

If any java processes are occupying 95-100% cpu usage then check the log files for any continuous looping messages or jdbc transaction time outs.

fix the problem and kill manged  server using kill -9 pid and restart the service instance.

404 error:

page can't be displayed.

10.4.5 404 Not Found

The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent.

1) check whether they are using correct url

2) check whether apache server is running ( ps -ef | grephttpd) ( ps -ef | grep -i apache)

3) check the diskspace of Apache server if it is full then delete the log files (df -kh)

goto Apache2.2/logs

delete old logs

4) Check whether the deployed application is in active state

5) If the deployed application is failed then fix the issue and redeploy the application

Users are getting 404 error some times and they are able to access the application sometimes.

1) check whether all managed servers are in running state.

if one of the managed server is in shutdown state then bring up the server.

check the http requests in access.log file for all managed server

if you are getting 404 error in one of the managed server log. then check server log for any errors

i got the in log file:

port already in use

netstat -anp | grep 8002

if the port is listened on any other instance. restat managed server.

if the issue still persists then raise a reqest to network team..

500 error:

Service unavailable

this error is due to server down

check apache or WebLogic service instance is the server is down then start the server.

Slow response:

check All WebLogic server status. bring the servers up if they are down

check network handshake requests in application logs. If you found any issues related to n/w then raise request to n/w team.

check for stuck thread issues in WebLogic. If you found any stuck thread issues then take thread dump and analyze.

checkcpu usage for java processes.

check heap size of WebLogic server gc log or in console.

If the heap size is more than 80% then take heap dump send it to l3 support.

check no of users logged in to the application.

check for long running quiries from data base side.

check for latency in data base side.

check memory leaks in gc logs.

check connection leaks in the WebLogic server side.

check space in WebLogicunix machine.

check apache server space.

OOM(OutOfMemory):

-     Login to the Corresponding Server through Putty

                -     Then Check the Status of the Server instances

                -     Check the Server logs and Out logs for OutOfMemory Error

 -     Take the Access logs at the time of OOM and it will be good if we take thread dump

If Server(s) is/are in Running State.

                -     Analysis the Thread dump for the Cause of OutOfMemory Error (Due to App/Server)

                -     Then Depending on the Server Status (if not in Running State) Restart the Server.

i) OutOfMemory during deployment:

1.                  If the application is huge(contains more than 100 JSPs), we might encounter this problem with default JVM settings.

2.                  The reason for this is, the MaxPermSpace getting filled up.

3.                  This space is used by JVM to store its internal datastructures as well

4.                  as class definitions. JSP generated class definitions are also stored in here.

5.                  MaxPermSpace is outside java heap and cannot expand dynamically.

6.                  So fix is to increase it by passing the argument in startup script of the server: –XX:MaxPermSize=128m (default is 64m)

ii) Memory related issues can be caused by many problems including:

• Java Heap is undersized for the environment

• There is not enough native memory available for the java process

• Web server is overloaded

Below are some general guidelines on how to address memory-related issues. In addition, you should search the “My Oracle Support” knowledge base for the specific “OutOfMemory” error message that you see in your PIA\_WebLogic.log.

Also, you can collect further details on WebLogic memory usage by using monitoring tools referenced in section “Monitoring WebLogic Memory Usage”

General Guidelines on Fixing OutOfMemory Issues:

1. First, you need to determine if WebLogic is running out of “native heap” memory or “java heap” memory. Typically you are able to tell this by checking the “OutOfMemory” error message in the PIA\_WebLogic.log:

a. If the error message refers to “native” or to a “thread” related error, it is an issue with “nativememory”. Examples of errors due to running out of native memory are:

• “Unable to create new native thread”

• “Error starting thread: Not enough storage is available”

Native memory errors are more likely to occur on PeopleTools 8.50 and lower versions where you are running a 32-bit java process (which has address space limitations)

b. Any other error messages, are usually due to running out of java heap memory

2. If you are running out of java heap, you may want to start by increasing the java heap settings (if you are unable to increase java heap setting, then the other option is to add more WebLogic PIA’s to the environment).

Increase the java heap setting as follows:

a. First check the current heap setting by searching string “-Xmx” in your WebLogic log (for Unix, search PIA\_stdout.log. For Windows, search NTservice-<DOMAIN\_NAME>-PIA.log).

The “-Xmx” value shows you the current heap setting. For example, these settings show that the minimum heap (-Xms) and maximum heap (-Xmx) are set to 512mg:

Java command line=java -server –Xms512m –Xmx512m

b. For PeopleTools 8.50, try increasing the heap (at 256mg increments) up to 1.5gb. For PeopleTools 8.51 or 8.52, you can increase the heap even higher, provided the server has enough memory available.

For Unix, you can change the heap setting in file setEnv.sh. Example:

JAVA\_OPTIONS\_LINUX="-jrockit -XnoOpt -XXnoJITInline –Xms768m –Xmx768m

• For Windows, you will either need to change the setting in the Windows registry, or else change setting in setEnv.cmd and then rebuild the Windows Service

Refer to the following document for more details on changing the java heap setting:

Doc#638298.1: How To Increase/Decrease JVM Heap Size for WebLogic

3. If you are running out of native memory heap, then you may want to consider doing the following:

a. Lower the java heap setting (ieXmx/Xms settings) in order to allow more of the java process’ memory to be used for “Native Memory”. (see step 2b above, for instructions on changing java heap setting)

If the java heap is already being fully utilized, and you are unable to lower it, then you may want to consider adding additional PIA’s to your environment

b. Lower the thread stack size. Note that the threads use native memory, so if you lower the thread stack size, then the threads will not consume as much memory. The thread stack size is specified using parameter “-Xss”. Refer to the following document for details

651285.1: WebLogic Error: "java.lang.OutOfMemoryError: unable to create new native thread"

Log files not rotating:

check the diskspace if it is full then delete old logs

check whether log4j properties file set in classpath

1.  Check the Status of the Server

2. ./startWebLogic.sh

3. ./startManagedWebLogic.sh <manageservername>

4.   [0R]

5. Check through console.

6.  Check the disk Space(if full, Delete the logs and then need to restart the Server)

7.   du –kh (folder)

8.     df –kh (filesystem)

9.  Avail space is 90%

If full , mv <source path><destination path>

Delete, rm –rf<filename: adminserver.log>

Stuck thread:

"[STUCK]”

When a transaction is running more than 5 minutes, a message (example below) is logged to the PIA\_WebLogic.log.

<Apr 18, 2011 12:47:04 PM PDT><Error><WebLogicServer><BEA-000337><[STUCK] ExecuteThread: '4' for queue: 'WebLogic.kernel.Default (self-tuning)' has been busy for "675" seconds working on the request …..

Note that the message shows that the thread is ‘STUCK’. But in fact, the thread may not be stuck, but is just taking a long time to complete. These threads often successfully complete, if given enough time.

If you see a lot of long-running threads, at the time users are experiencing problem, then this indicates that the web server is having issues processing threads, which may cause the web server to hang.

Long running threads can be caused by different issues. The problem often occurs due to issues on the app server or database subsequently causing the threads to queue up and wait on the web server. So if you see a lot of stuck threads, you may want to troubleshoot further by doing the following:

1. Get a thread dump as described in section "Creating/Analyzing Thread Dumps". The thread dump may help you determine whether the threads are getting hung up on the app server or database.

2. Have your DBA check for long running SQL's and/or DB locks

3. You can also look at the 'Stuck' thread messages in the PIA\_WebLogic.log to see what user(s) are running the transactions and the specific component they are running. This may help you determine if there is a specific user and/or transaction that is causing the problems.

Port Conflict Issue:

While configuring a new WebLogic instance and starting it, that might be get an issue like : "Port already in use". There could be many reasons for this one.

1. on the same machine multiple standalone instances might be running one of the instance already used that port which you have given for new configuration.

2. apache might be running with the same port.

3. middleware might be running on the same machine with same port

On Solaris Operating environment we have 2 options:

1. usingpfiles command

netstat –na|grep --> identify port in use

pfiles |grep -isockname |grep port --> look for every java process is initialized by startWebLogic.sh or startManagedWebLogic.sh

2. Another way costly one (Third party package) to find the process that is using particular port is :

lsof -itcp:

3. Best way is perl script using a method it will check only standard ports which are used by the system.

getservbyport(intport\_number, const char \*protocol\_name)

#!/usr/bin/perl

($name, $aliases, $port\_number, $protocol\_name) = getservbyport(7001, "tcp");

print "Name = $name\n";

print "Aliases = $aliases\n";

print "Port Number = $port\_number\n";

print "Protocol Name = $protocol\_name\n";

JVM memory arguments:

-XX:-PrintGCDetails outputs detailed  information at each collection

-XX:-PrintGCTimeStamps outputs a time stamp at the start of each collection

-xloggc=<filename>           outputs gc information to the specified file

-XX:-DisableExplicitGC disable calls to system .gc( )

- -XX:NewSize=2m default size of new generation

-XX:MaxNewSize=size maximum size of the new generation

-XX:PermSize=64m default size of permanent generation

-XX:MaxPermSize=64m maximum size of the permanent generation

- -Xms256m Initial heap size

--Xmx512m  maximum heap size

-xx:survivor Ratio=<value>   Ratio of survivors spaces to young generation

-XX:-UseParallelGC   Use parallel garbage collection for scavenges.

THREAD DUMP

Tread dump:- Thread dump provides a snapshot of the current active live threads. It provides the stack trace of all the java threads in the JVM. It is used when the server is hung and we want to see the threads executing and take their dump.

There are different ways to take thread dump.

In unix: kill -3 <pid>

In windows:  ctrl+break

WebLogic.Admin utility:  javaWebLogic.Admin -url t3://localhost:7001 -username WebLogic -password WebLogic THRED\_DUMP

WLST Scripting:

connect('WebLogic','WebLogic','t3://localhost:7001')

cd('server')

cd('AdminServer')

TreadDump()

disconnect()

exit()

Admin console:

Step1: login to the admin console

Step2: Click on server

Step3: Navigate to servers

Step4: Click monitor tab

Step5: Click on tread

Step6: Click on the dumpthread stack.

Locating the Thread Dump:  The thread dump is placed in the WebLogic log file. The log file location varies depending on the OS platform:

 For UNIX: the output is sent to:

<PS\_HOME>/webserv/<DOMAIN\_NAME>/servers/logs/PIA\_stdout.log

For Linux: the output is sent to:

<PS\_HOME>/webserv/<DOMAIN\_NAME>/servers/logs/PIA\_stderr.log

For Windows: the output is sent to:

<PS\_HOME>\webserv\<DOMAIN\_NAME>\servers\PIA\logs\NTservice-<DOMAIN\_NAME>-PIA.log

Analyzing a Thread Dump: The thread dump can be a bit challenging to analyze, and you may need assistance from an Oracle Support Engineer. Below are some tips on how to analyze the thread dump. This information is broken out into the following sections:

1. General Information about the thread dump

2. Overview of types of threads commonly seen in thread dump

3. Examples of different issues you may observe in the thread dump

1) General Information about the Thread Dump:  Note that the thread dump always begins with this line:

===== FULL THREAD DUMP ===============

And ends with this line:

===== END OF THREAD DUMP ===============

The first line of the thread dump shows when the thread dump was created, followed by the exact java version you are using.

Example:

Mon Apr 18 12:46:56 2011

Oracle JRockit(R) R28.0.0-679-130297-1.6.0\_17-20100312-2123-windows-ia32

2) Overview of Types of Threads commonly seen in Thread Dump:

i) Threads waiting for Requests:  You will always see some threads that are just waiting for work, as WebLogic always allocates some threads to be available and ready to process any incoming requests. These threads can easily be identified because you’ll see “ExecuteThread.waitForRequest” in the call stack. These threads will be in ‘ACTIVE’ or ‘STANDBY’ mode. These threads do not have much significance when troubleshooting. However, if you see a lot of these threads waiting for requests (20 or more), it most likely indicates that the environment is just recovering from a very heavy load, when the thread dump was taken (and as the load diminishes, WebLogic will remove many of these extra threads that are waiting for requests)

Ex: at WebLogic/work/ExecuteThread.waitForRequest(ExecuteThread.java:157)

ii) Socket Muxer Threads: You will also see approximately two to five socket muxer threads. These threads' main responsibility is to read the request off the socket and pass the work to the appropriate thread. WebLogic allocates a percentage of execute threads from the self-tuning thread pool to be Muxer threads. Usually you will see three or four of these threads:

"ExecuteThread: '0' for queue: 'WebLogic.socket.Muxer'" id=25 idx=0x60 tid=2068 prio=5 alive, in native

iii) ListenThreads: You will also see approximately six “listen threads”, usually three for SSL and three for non-SSL. The purpose of these threads is to wait for connections to arrive. All browser requests enter the WebLogic server through these threads.

"DynamicListenThread[Default]" id=39 idx=0x90 tid=2812 prio=9 alive, in native

"DynamicSSLListenThread[DefaultSecure]" id=40 idx=0x94 tid=3148 prio=9 alive, in native

iv) Jolt Connection Threads: WebLogic Server and the Tuxedo Application Server use Jolt to communicate with each other. PIA creates two threads inside the WebLogic’s JVM per Jolt connection. For each Jolt connection made between WebLogic and the Tuxedo Application Servers, you will see a LLENwReader and a LLENwWriter thread in the thread dump:

"LLENwReader" id=52 idx=0xc4 tid=4408 prio=5 alive, in native, daemon

"LLENwWriter" id=53 idx=0xc8 tid=7828 prio=5 alive, waiting, native\_blocked, daemon

v) Threads waiting on Application Server: If the web server is waiting on the app server to process a request, you will see the following thread (below)

at bea/jolt/IOBuf.waitOnBuf(IOBuf.java:119)

3) Examples of Different Issues you may Observe in Thread Dump: Below are examples of different issues and the thread stacks you may observe.

Many threads waiting on App Server: If you see a lot of threads such as the one below, then this means that many of the WebLogic threads are waiting on the application server to finish processing the request:

at bea/jolt/IOBuf.waitOnBuf(IOBuf.java:119)

i) Many threads processing the same call stack: If you see many threads all processing the same call stack, then you may need to review contents of the call stack in order to troubleshoot the issue. For example, in one case, the web server hung and the thread dump showed hundreds of threads like the one below. This was caused by an issue with a proxy server configuration, causing all threads to get hung up at logout:

com.sun.net.ssl.internal.ssl.SSLSocketImpl.readRecord(SSLSocketImpl.java:798)

psft.pt8.psp.logoutAccessedPIAs(Unknown Source)

ii) All threads busy and waiting on one thread: By design, the PIA does not allow more than one request per HTTP session, to be submitted to the application server. If the PIA receives multiple requests from the same HTTP session, it will queue up all subsequent requests and process just one at a time. Typically, there should not be situations where the PIA receives multiples requests from the same HTTP session. However, this can occur in the following situations:

1. You are using a proxy server that is re-submitting requests to the web server if a response is not received within a certain time.

-OR-

2. A user submits a long-running request, and while waiting for the request to finish, the user continuously attempts to submit more requests.

When one of the above scenarios occurs, in the thread-dump you see one request waiting on Jolt to get response from the App-Server and many other threads waiting for the lock on the session to be released. Below are excerpts from a thread dump, showing this situation:

a) There are many threads like this that are “blocked”, and all the threads are waiting on the same lock #.

-- Blocked trying to get lock: java/lang/String@0x27D36AC0[thin lock]

b) The thread that is holding the lock on “0x27D36AC0” (that all blocked threads are waiting on), is usually processing a jolt request (ie it is waiting on the application server):

at bea/jolt/IOBuf.waitOnBuf(IOBuf.java:119)

^-- Holding lock: java/lang/String@0x27D36AC0[thin lock]

c) At the end of the thread dump, you may see a list of “blocked locked chains”. In this list, you’ll notice that all threads are waiting on one thread: “Thread #0” in this example. Which happens to be a jolt request (ie it is waiting on application server)

Blocked lock chains

===================

Chain 2: "[ACTIVE] ExecuteThread: '2' for queue: 'WebLogic.kernel.Default (self-tuning)'" id=35 idx=0x80 tid=3964 waiting for java/lang/String@0x27D36AC0  held by:

"[ACTIVE] ExecuteThread: '0' for queue: 'WebLogic.kernel.Default (self-tuning)'" id=16 idx=0x48 tid=180 in chain 1

Chain 3: "[ACTIVE] ExecuteThread: '3' for queue: 'WebLogic.kernel.Default (self-tuning)'" id=44 idx=0xa4 tid=4620 waiting for java/lang/String@0x27D36AC0  held by:

"[ACTIVE] ExecuteThread: '0' for queue: 'WebLogic.kernel.Default (self-tuning)'" id=16 idx=0x48 tid=180 in chain 1

Chain 4: "[ACTIVE] ExecuteThread: '4' for queue: 'WebLogic.kernel.Default (self-tuning)'" id=49 idx=0xb8 tid=1120 waiting for java/lang/String@0x27D36AC0

held by:

"[ACTIVE] ExecuteThread: '0' for queue: 'WebLogic.kernel.Default (self-tuning)'" id=16 idx=0x48 tid=180 in chain 1

Analysing ThreadDump by using Summari tool:

Download: The binary is available for download at                                                   <http://yusuke.homeip.net/samurai/en/samurai.jar>

How to launch samurai: You can simply double-click to launch Samurai on your desktop or type as following in your command prompt.

            $java  -jar  samurai.jar

Automatic update is not available with this way. Please check and download latest version manually.

Step1: Drag and drop the ThreadDump into summary tool

Step2: When Samurai detects a thread dump in your log, a tab named "Thread Dump" will appear.

Step3: You can just click "Thread dumps" tab to see the analysis result. Samurai colors idle threads in gray, blocked threads in red and running threads in green.

There are two resultant views and Samurai shows "Table view" by default.  
In many cases, you are just interested in the table view and the sequence view. Use the table view to decide which thread needs be inspected, the sequence view to understand the thread's behavior.

Result1:

Result2:

HeapDump: A Heapdump is a snapshot of JVM memory – it shows the live objects on the heap along with references between objects. It is used to determine memory usage patterns and memory leak suspects. It is useful to analyse OOM(OutOfMemory) situations.

To take Heap dump:

Eclipse Memory Ananlyser is a very useful tool to analyze heap dumps. It has a lot of features such as Memory Leak detection where it runs an automated test to determine the suspected leaks.

Step 1) Start the WebLogic Server, with the application in active state which causes memory leak.

Step 2) Get the process id of the server using jps

Step 3) Access the application that causes memory leak

Step 4) Take heap dump at regular interval using jmap.

jmap -dump:format=b,file=dump1.bin [processId]

Analyzer HeapDump by using Eclipse MAT:   Analyzer  Open the Heap Dump in Eclipse Memory Analyzer (U can download it from<http://www.eclipse.org/mat/downloads.php>) Approximate size is 42 MB. Just u need to extract this Zip then u can directly start (no Installation needed)

Step1:

Step2: Observe the heap usage of Objects in the heap dumps. If the object instance keeps on increasing in the subequent heap dumps, force a garbage collection from the WebLogic Server console.

Step3: Take heap dumps again and open in the Eclipse Memory Analyzer. If the number of instances still don’t go down for those objects, you can expect to see this  
<Jul 16, 2010 10:49:15 AM IST> <Critical> <Health> <BEA-310003> <Free memory in  
the server is 47,856 bytes. There is danger of OutOfMemoryError>  
Exception in thread “Thread-12″ java.lang.OutOfMemoryError: Java heap space  
at demo.MemoryLeakTest.runTest(MemoryLeakTest.java:14)  
at jsp\_servlet.\_\_memoryleak$1.run(\_\_memoryleak.java:86)  
at java.lang.Thread.run(Thread.java:619)

If the leak is happening due to a WebLogic Class, it can be a known issue or an undiscovered BUG. You need to get in touch with Oracle Support. If it’s an Application Class, you need to contact the developers. Out of Memory can also happen dude to third party codes such as database drivers.

How to install Eclips MAT? How to Analyze Heapdump using Eclips MAT and Jhat tools:

Step1: download the Eclips MAT

Step2: Extract Memory Analyser zip file and open MemoryAnalyzer.ini

Step3:  Double-click on MemoryAnalyzer.exe to start Memory Analyser Tool

Step4: Select Search for new features to install and click Next

Step5:  Accept the license agreements and click Next

Step6: Click Finish to install the extensions

Step7: Click Install All to ignore the warning

Step8: Restart Memory Analyser to reflect changes.

JPROFILE

Introduction: JProfiler eases developers in creating more efficient applications by improving their performance. It is oriented towards testing and exploring different aspects of the performance of a Java program, concerning the operation of the JVM making use of available platform resources. JProfile is mainly using to find out the memory leaks of JVM.

JProfiler provides the following functionality:

1.                  Memory consumption measurement

2.                  Memory stack frames tracing

3.                  CPU loads profiling

4.                  Momentary heap allocation information

5.                  Thread state progress statistics

6.                  Visual representation of JVM work loads by different parameters

7.                  Source code reference

8.                  Garbage collection during profiling

9.                  Remote profiling

10.              Profiling only of a particular stage or a combination of several stages of the program life cycle

11.              Using different target environments

12.              Saving profiler information for further examination

13.              Exporting profiler information in text format

 JProfiler is a commercially licensed [Java](http://en.wikipedia.org/wiki/Java_(programming_language)) [profiling](http://en.wikipedia.org/wiki/Profiling_(computer_programming)) tool developed by ej-technologies GmbH, targeted at [Java EE](http://en.wikipedia.org/wiki/Java_EE) and [Java SE](http://en.wikipedia.org/wiki/Java_SE)applications.

1.                  JProfiler works both as a stand-alone application and as a plug-in for the Eclipse software development environment.

2.                  JProfiler supports local profiling (analysis of applications running on the same machine as the JProfiler software) and remote profiling (analysis of Java applications running on remote machines)

3.                  Enables both memory profile to assess memory usage and dynamic allocation leaks and CPU profiling to assess thread conflicts.

4.                  Provides visual representation for the virtual machine load in terms of active and total bytes, instances, threads, classes, Garbage Collector activity.

Download Jprofiler(<http://www.ej-technologies.com/download/jprofiler/files.html>)

You will be asked to provide your name and e-mail id.An Evaluation Key will be mailed to you.

At the time of installation, you will be prompted for the installation key, copy it from your mail and paste it as shown in the screenshots.

NOTE: It is not recommended to use JProfiler in Production Environments .as it consumes more resources.which may not be desired in Production Envs.

Patch: A patch is a piece of software designed to fix problems[[1]](http://en.wikipedia.org/wiki/Patch_(computing)#cite_note-news.com-1) with, or update a [computer program](http://en.wikipedia.org/wiki/Computer_program) or its supporting data. This includes fixing [security vulnerabilities](http://en.wikipedia.org/wiki/Security_vulnerability)[[1]](http://en.wikipedia.org/wiki/Patch_(computing)#cite_note-news.com-1) and other [bugs](http://en.wikipedia.org/wiki/Computer_bug), and improving the [usability](http://en.wikipedia.org/wiki/Usability) or [performance](http://en.wikipedia.org/wiki/Performance). Though meant to fix problems, poorly designed patches can sometimes introduce new problems (see [software regressions](http://en.wikipedia.org/wiki/Software_regression)). In some special cases updates may knowingly break the functionality, for instance, by removing components for which the update provider is no longer licensed or disabling a device.

Patch management is the process of using a strategy and plan of what patches should be applied to which systems at a specified time.

Patch installation steps:

            Step1: Take the back up of bea home directory and config.xml file

Step2: Copy all patches along with the patch-catalog.xml file to the Linux box.at                                      \*/oracle/utils/bsu/cache\_dir

            Step3: Stop all the servers including admin.

            Step4: On the Linux machine go to the bsu folder. ( \*/oracle/utils/bsu/) and run the                                    following command

               Ex: ./bsu.sh -prod\_dir=<WebLogic home> -patchlist=<patch name> -verbose -install

            Step5: Use the below command to check the output

              \*\oracle\utils\bsu> ./bsu.sh -view -prod\_dir=/usr/local/oracle/wls1033/wlserver\_10.3                                                        -status=applied

            Step6: Start the all servers.

            Step7: Check the application health.

Commands:

      Patch installation: ./bsu.sh -prod\_dir=/usr/local/oracle/wls1033/wlserver\_10.3 -                                                               patchlist=4EWM -verbose   -install

      Patch Uninstallation: ./bsu.sh -remove -patchlist=1FKM -                                                                                                    prod\_dir=/usr/local/oracle/wls1033/wlserver\_10.3 –verbose

      Check for what are all patches installed: ./bsu.sh -view -                                                                                             prod\_dir=/usr/local/oracle/wls1033/wlserver\_10.3 -status=applied

Creating patch logs:  ./bsu.sh -report -log=test.log -log\_priority=trace

Screen shots

Applying patches on WebLogic Server using Oracle Smart Update(BSU):  Oracle provides the Smart Update utility to apply patches and upgrade the WebLogic Server installations. Oracle’s WebLogic Server is now a critical component  of Fusion Middleware and every other component of Fusion Middleware requires WebLogic Server to be installed as  a pre-requisite. Applying patches and upgrading WebLogic Server is quite straight forward using the Oracle’s Smart Update utility,the documentation for Oracle Smart Update Utility can be found [here.](http://download.oracle.com/docs/cd/E12839_01/doc.1111/e14143/toc.htm)

Step1: Shutdown and take a complete backup of the WLS environment.

The Startup/Shutdown scripts are placed in   $WLS\_HOME/user\_projects/domains/<domain\_name>/bin

Step2: The Oracle Smart Update Tool is located at  “$WLS\_HOME/utils/bsu“

Step3: Launch the the Oracle Smart Update Tool :

Step4: Once logged in, you will be presented with Oracle Smart Update Dialog.

Step5: You can choose to “Register for security updates”, this is usually helpful to keep yourself updated with the latest security updates and product expiration.

Step7: On the left pane you would see   WebLogic Servers installed  and on the right pane you will see two tabs. “Get Patches” and “Manage Patches” and a section to show the downloaded patches.

Step8: Now select the patches and hit the “Download Selected” button, you will be prompted if you wish to to validate and resolve conflicts.

Step9: The Validation completes with the following message:

Step10: Click “OK” to proceed downloading the patches.

Step11: Once the patches are downloaded and click the “Manage Patches” tab to proceed with the patch application. In the “Downloaded Patches” section you will notice the patches downloaded, click the “up” arrow to apply the patch

Step12: You will be prompted with couple of prompts for you to take action:

Click “OK” to proceed

Step13: Once more the validation is done, click “OK” to proceed                                       Step14:  One more “Are you sure?” prompt, annoying I know. Click “Proceed” to apply the patch

Step15:  Once the patch is applied you’d see the patch in the Applied Patches “Default” tab

That  the patch is now applied. If you face any issues its worth investigating the server logs.

Log File Location: The log file location is:

            C:/bea/user\_projects/domains/ram\_domain/servers/admin server/log

1) Access log:

2) Serveer log: The server log records information about events such as the startup and shutdown of servers, the deployment of new applications, or the failure of one or more subsystems. The messages include information about the time and date of the event as well as the ID of the user who initiated the event.

3) Domain log: This will have about domain information. (domainname.log)

4) AdminServer log: This will have about the AdminServer information. (AdminServer.log)

5) Out logs: This will have about the JVM output. (Adminserver.out)

6) Application logs: This will have information about each and every application which we deployed in server.

7) Node Manager logs: This will have information about Node Manager. (node manager.log)

            (C:/bea/WebLogic91/common/nodemanager/nodemanager.log)

Diff b/w WebLogic 8,9,10 & 11 versions:

|  |  |  |  |
| --- | --- | --- | --- |
| Features | WLS8.1 | WLS9x | WLS10.3 and 11G |
| JDBC Connection Pool-Max Capacity | PM-25(AdminServer)  DM-15(AdminServer)  PM-15(managed Server) | PM and DM-15  (AdminServer and Managed Server) | PM and DM-15  (AdminServer and Managed Server) |
| Execute ThreadDefaultThreadCount | PM-25(AdminServer)  DM-15(AdminServer)  PM-15(managed Server) | PM and DM-15  (AdminServer and Managed Server) | PM and DM-15  (AdminServer and  Managed Server) |
| JMS Services | Queue and Topics can beCreated under JMS Server | Queue and Topic Services Can be Created only by JMS Module | Queue and Topic Services  Can be Created only by JMS Module |
| JMS Server Starting and Stopping | Not Available | Not Available | Particular JMS Instances Can be Stopped |
| JMS Advanced features | Quota,SAF(Store and Forward Agents) is not Available | Quota,SAF(Store and Forward Agents)is Available | Quota,SAF  (Store and Forward Agents)  is Available |

|  |  |  |  |
| --- | --- | --- | --- |
| JMS Configuration repository | NO Config file for JMS | Separate Configuration File for JMS inside the  WebLogic Domain | Separate Configuration File for JMS inside the  WebLogic Domain |
| JMS transaction | No JMS transaction Logs | No JMS transaction Logs | JMS transaction Logs |
| Cluster—Unicast  Address | No Unicast Address | No Unicast address | Unicast Address is available |
| JMS Destination KeyCustom Key TypeFacility | No Custom Key Type | No Custom Key Type | No Custom Key Type |
| Garbage Collector Process Scheduled GC | No GC | NO GC | Scheduled Garbage Collection |
| XML—Xpath and XLang—WebService | Not Supported | Supported | Supported |
| EJB 3.0 | Not Supported | Supported | Supported |
| Advanced WEbservice Support by SOA | Not Supported | NOT Supported | Supported |
| Oracle Fusion and Ebusiness Suite Integration | Not Supported | Not Supported | Supported |
| Log File(Default Transaction Log) | Available | Not Available | Not Available |
| JDBC log | Available | Not Available | Not Available |
| Jolt Connection Pools | Not Available | Available | Not Available |
| Config folder | Not Available | Available | Available |
| Prepare, Active states | No Prepare state for application. Only active state | Prepare state for application, This optimises memory utilization. | Prepare state for application, This optimises memory utilization. |
| Deployment fails | Server dosent come up if deployment fails | Server boots in ADMIN mode if deployment failes | Server boots in ADMIN mode if deployment failes |
| configuration information | All configuration information is in one config.xml | Seperate xml files for domain config and jms modules are added | Seperate xml files for domain config and jms modules are added |
| Side by side deployment | Side by side deployment is not possible | Side by side deployment is possible | Side by side deployment is possible |
| Lock and Edit | Not Available | Available | Available |
| Connection pools | We have connection pools and datasources | We have datasources and connection pools are inside datasources. | We have datasources and connection pools are inside datasources. |
| app inf lib and classes | Not Available | Available | Available |
| Deployment updates | We need to delete and redeploy from admin console | We can update the application using admin console | We can update the application using admin console |
| Queues | We have exclude queues. | We have work managers | We have work managers |
| WebLogic Scripting Tool(WLST) | Not Available | Available | Available |
| license.bea | genericlicense.bea file for each version that you candownload from Oracle. | genericlicense.bea file for each version that you candownload from Oracle. | In WLS 10 MP2 and above, the downloadedsoftware comes with a fully blown license so no need to evendownload a new one from Oracle. |

Ticketing Tools

1) BMC Remedy ticketing tool:

IITL(Information Technology Infrastructure Library)Process:

1.                  Change Management.

2.                  Incident Management.

3.                  Problem Management.

4.                  Release Management.

Different Types of Tickets:

1) Incident ticket which identity by INC: Something happen accidently the ticket which raises manually or automatically.

Ex: WebLogic server failed to startup  ticket will be raised automatically.

2) Change ticket by CRQ: If somebody wants to do change or creating a new during that time the change management ticket uses.

3) Problem ticket which identified by PBC: It is used to managed problem investigations known errors and solutions DB(Data Base)entries. Problem management can practically prevent the occurency of incidents errors and addition management.

States of Tickets:

1) New: Displays when creating a new record or ticket.

2) Assigned: Auto set to assigned when you create a new incident assigned to some one.

3) In progress: Actively working on that incident also must select at assigning a record to yourself.

4) Pending: can’t work on that incident must fill in the reason failed or pending. It means keeping the ticket on hold for some time.

5) Resolved: A solution or work around has been found, must fill in the status reason failed.

6) Closed: The system will auto-close in five business days or if user wants close the ticket we can close immediately or manually.

7) Canceled: If record was an accident or the issue doesn’t need resolution customer or support staff may task incident as cancelled.

Urgency or priority:

1) Critical: It will impact business.

2) High: It will import only for that server or only for that particular batch systems.

3) Medium: It is not that much critical but still we need take task on that job.

4) Low: It does not require to consider this point.

            Based on how many systems, process and business units are afforded as well as if the work around is possible or not.

2)  Amdocs clarify CRM Support:

Handling Change Request: If any config  changes have been done. The change request will be raised.

Status of Tickets:

1) Requested: The ticket is in requested state. These are different levels of approvals needed for implementing any change request.

Level1:  Line manager ready.

Level2:  Production co-ordinater ready.

Level3:  Change co-ordinater ready.

Level4:  Skill group implementation(Request implementer) ready.

2) Ready: While getting the approvals the ticket will be in ready state.

3) Scheduled: After getting the approvals the ticket will come in to scheduled state.

4) Accepted: For implementing any change request ticket should be in scheduled state. The request has to be accepted.

5) Resoled:

6) Failed

7) Rejected